Quality of Transaction

Von der Fakultät für Informatik, Elektrotechnik und Informationstechnik
der Universität Stuttgart zur Erlangung der Würde
eines Doktor-Ingenieurs (Dr.-Ing.) genehmigte Abhandlung

vorgelegt von

Matthias Kaschub

geb. in Ravensburg

Hauptberichter:	Prof. Dr.-Ing. Andreas Kirstädter
Mitberichter:	Prof. Dr. Tobias Hoßfeld
Tag der mündlichen Prüfung:	25. Oktober 2017

Institut für Kommunikationsnetze und Rechnersysteme
der Universität Stuttgart

2017

Abstract

Today, for many Internet users, the quality of experience with a network is limited by the properties of the Internet network being accessed. In most cases, the network accessed is the bottleneck in the transmission of data between the Internet user and a corresponding node. An contributing factor for the user's experience is the achievable bitrate. At a higher bitrate, for example, it is possible to view video streams at higher image quality. And more important, at a higher bitrate, a user will need to wait less time for data transmissions of a given size.

However, the average utilization of Internet access networks is very low, and due to this, the maximum bitrate is only utilized for short periods. This is especially true for Internet access technologies of a dedicated medium for each user, such as DSL, where the bitrate is available only for a single user. Even though the available bitrate is only rarely fully utilized, for a good user experience, it is important to provide a high bitrate on the network being accessed. One main reason is the parallel use of multiple applications that affect each other noticeably. Examples for such impairments are: a video stream cannot be played with the desired quality, the latency in online games become worse, or waiting times in web applications increase to a point where it seriously affects the workflow. Users often explicitly close file-sharing applications, pause large downloads, or disable smartphone application updates.

Classical approaches for improving the quality of service or user experience (QoS/QoE) can address such situations; but they did not gain widespread support, as these approaches require the collaboration of components beyond that of the access network. A classification of such approaches and further reasons that hinder their success are given in the first chapter.

The objective of this work is to design and evaluate a new system that improves the user experience when the access network is the main bottleneck. Simple to be deployed, this approach does not involve parties beyond the user and the headend of the access network. The key features required to avoid the problems of previous QoS/QoE approaches, allowing easy implementation in applications and on networking equipment are that the system uses easy-to-understand metrics from a user's perspective for the user application integration, and that it allows its corresponding metrics to be efficiently measured by networking equipment in the access network.

To achieve these goals, this approach defines transactions that represent user-observable results.

The corresponding network view of such a transaction is simply all the transmitted data that lead to this result.

When a user requests a web page, for example, this approach uses the total amount of time until the network transmission of the transaction on application layer is complete as its metric. From a user perspective, this metric represents the time until the web page is fully loaded and can be displayed. From a network perspective, this corresponds to the time for transmitting a certain amount of data over a set of transport layer connections. In case of HTTP, these connections can be specified with their IP 5-tuple.

This approach classifies a wide range of networking applications into multiple abstract application classes that use three different types of transactions and their respective metrics. The user's experience of each transaction is formulated as a utility function, which is a function derived from the metrics of the corresponding transaction type.

In a system using this approach, applications formulate their requirements for their transactions of utility functions, based on the metrics of the transaction type. The headend of the access network identifies the transaction, and applies these requirements to the data transmissions. In other words, this approach allows the user to configure the prioritization rules for its transmission on the headend of the access network in a dynamic way.

This work introduces such a system and evaluates it using event-based network simulation. The evaluation shows that the user experience in a given access network can be considerably improved in comparison to the currently prevailing best-effort approach. Alternatively, the approach can be used to increase the number of users served with a given access network and the same user experience. This approach can realize these improvements and gains, in all evaluated scenarios and network topologies. For example, in access networks with dedicated resources per user, as well as in access networks where resources are shared among multiple users. The system is robust against errors and inaccurate formulation of the requirements. Even when the metrics of the requirements used for prioritization deviate from their real value on average by one order of magnitude, the system can still achieve significant improvements.

In Chapter 2, the general Quality of Transaction (QoT) approach is introduced, and an architecture for implementing it is developed in Chapter 3. The methodology for the evaluations is described in Chapter 4. These evaluations also require a specific model for Internet traffic in the access network, which is introduced and described in Chapter 5. In Chapter 6 evaluates and compares the properties of this approach to alternative approaches.

Kurzfassung

Für viele Internet-Benutzer wird die empfundene Qualität von Datenübertragungen über das Internet derzeit durch die Eigenschaften des jeweiligen Internet-Zugangsnetzes limitiert. Das Zugangsnetz ist in diesen Fällen meist der Flaschenhals der Übertragungsstrecke zwischen dem Benutzer und der jeweiligen Gegenstelle im Internet. Ein Faktor, der die empfundene Qualität von Datenübertragungen in vielen Fällen maßgeblich beeinflusst, ist die erreichte Bitrate der Übertragung. Eine höhere Bitrate ermöglicht beispielsweise eine Video-Übertragung mit höherer Qualität, vor allem aber muss der Benutzer bei Datenübertragungen mit gegebener Größe kürzer warten, da diese schneller fertig sind.

Die durchschnittliche Auslastung von Internet-Zugangsnetzen ist in der Regel sehr gering, da die maximale Bitrate nur selten benötigt wird. Dies gilt insbesondere bei Internet-Zugangs-Technologien mit einem dedizierten Medium für jeden einzelnen Benutzer, wie zum Beispiel DSL, wo die verfügbare Bitrate jeweils nur diesem einen Benutzer zur Verfügung steht. Damit die Qualität des Internetzugangs als gut empfunden wird, muss also eine hohe Bitrate zur Verfügung gestellt werden, die aber nur selten ausgenutzt werden kann. Wird dieser Flaschenhals von mehreren Diensten, Anwendungen oder Benutzern gleichzeitig benutzt, so kann die mittlere Auslastung gesteigert werden. Sofern nur jeweils eine einzelne Übertragung die maximale Bitrate benötigt, wird die empfundene Qualität nicht dadurch beeinträchtigt. In der Praxis beeinflussen sich mehrere gleichzeitig aktive Anwendungen eines Anwenders oft merklich. Beispielsweise kann dann Video-Streaming nicht in der gewünschten Auflösung abgespielt werden, Online-Spiele haben zu hohe Verzögerungen und die Wartezeiten im Web-Browser werden zu groß. Dies behindert dann beispielsweise den Arbeitsfluss in webbasierten Anwendungen. Viele Anwender beenden daher oft explizit und manuell Filesharing-Anwendungen, halten große Downloads an und pausieren App-Updates von Smartphone und Tablett.

Klassische Verfahren zur Verbesserung der Dienstgüte (QoS/QoE) adressieren zwar dieses Problem, konnten aber bisher keine allgemeine Verbreitung erlangen, da es hierfür notwendig wäre, dass sich auch Netzwerk-Komponenten außerhalb des Zugangsnetzes daran beteiligen. Im ersten Kapitel dieser Arbeit ist eine Klassifikation solcher Ansätze gegeben, und es werden weitere Ursachen aufgeführt, die eine weitere Verbreitung behindern.

Das Ziel dieser Arbeit ist es, ein neues Verfahren zu entwerfen und zu bewerten, das die Dienstgüte der Nutzer verbessern kann, wenn vorwiegend das Zugangsnetz den Flaschenhals darstellt. Um die Verbreitung zu erleichtern, bezieht dieses Verfahren neben den Geräten des Benutzers und der Kopfstelle des Zugangsnetzes keine weiteren Komponenten mit ein. Die Hauptmerkmale, die zur Vermeidung der Probleme bisheriger QoS/QoE Ansätze notwendig sind, sind einfache Implementationen in den Anwendungen und den Netzwerk-Komponenten. Das Verfahren definiert daher aus Benutzersicht einfach zu verstehende Metriken für die Integration in Anwendungen. Um dies zu erreichen, definiert das Verfahren Transaktionen, die vom Benutzer beobachtbare Ergebnisse darstellen. Die zugehörige Netzwerk-Sicht einer Transaktion ist die Menge an übertragenen Daten, die zu einem solchen Ergebnis führt. Dieser Ansatz ermöglicht den Netzwerk-Komponenten entsprechende Metriken effizient im Zugangsnetz zu bestimmen.

Das Verfahren verwendet daher beispielsweise für Übertragungen von Webseiten als Metrik nicht die Verzögerung jedes einzelnen Paketes, sondern nutzt die gesamte Zeit für die Übertragung einer Transaktion der Anwendungsschicht. Aus Anwendersicht entspricht dies der Zeit bis zur Darstellung der Webseite. Aus Netzwerksicht entspricht dies einer bestimmten Datenmenge, die über eine oder mehrere Verbindungen übertragen wird. Im Falle von HTTP werden diese Übertragungen zum Beispiel durch IP-5-Tupel spezifiziert.

In dieser Arbeit wird eine Vielzahl von Netzwerk-Anwendungen in abstrakte Anwendungsklassen unterteilt. Diese Anwendungsklassen verwenden drei verschiedene Arten von Transaktionen mit entsprechenden Metriken. Die vom Benutzer empfundene Qualität wird als Nutzen-Funktion formuliert, die von den Metriken des entsprechenden Transaktions-Typs abgeleitet werden.

In einem System, das diesen Ansatz implementiert, formulieren die Anwendungen ihre Anforderungen an die Übertragung im Zugangsnetz in Form von Nutzen-Funktionen. Diese basieren auf den Metriken des jeweiligen Transaktions-Typs. Die Kopfstelle des Zugangsnetzes identifiziert die Transaktion und wendet die Anforderungen an, um die Übertragungen möglichst gewinnbringend zu steuern. Im Endeffekt ermöglicht dieses Verfahren also dem Benutzer Vorrangs-Regeln für seine Übertragungen auf der Kopfstelle des Zugangsnetzes zu konfigurieren.

In dieser Arbeit wird ein solches System eingeführt und mit eventbasierten Netzwerksimulationen untersucht. Die Bewertung des Verfahrens zeigt, dass gegenüber einem Best-Effort-Ansatz die empfundene Qualität mit einem gegebenen Internet-Zugangsnetz deutlich verbessert werden kann. Alternativ können mit einem gegebenen Internet-Zugangsnetz bei gleichbleibender empfundener Qualität wesentlich mehr Benutzer bedient werden. Das Verfahren kann diese Verbesserungen oder Gewinne in allen untersuchten Szenarien und Netz-Topologien realisieren. Beispielsweise wird dies sowohl in Zugangsnetzen mit dedizierten Ressourcen pro Benutzer erreicht, als auch in Zugangsnetzen, in denen sich viele Benutzer die Ressourcen teilen. Das Verfahren ist robust gegen Ungenauigkeiten bei der Bestimmung von Anforderungen der Anwendungen. Selbst wenn die für die Priorisierung verwendeten Anforderungen im Durchschnitt eine Größenordnung von den richtigen Wert abweicht, kann das Verfahren noch deutliche Vorteile erzielen.

In Kapitel 2 werden die Grundlagen des Verfahrens eingeführt. Eine Architektur für die Umsetzung des Verfahrens wird in Kapitel 3 entwickelt. In Kapitel 6 werden die Eigenschaften des Verfahrens untersucht und das Verfahren gegenüber alternativen Ansätzen bewertet. Die Methodik zur Bewertung wird in Kapitel 4 beschrieben. Für die Bewertung wird außerdem ein spezielles Verkehrsmodell benötigt, das in Kapitel 5 eingeführt und beschrieben ist.

Contents

List of Figures

List of Tables

Abbreviations and Symbols

3GPP	3rd Generation Partnership Project
AJAX	Asynchronous JavaScript and XML
ALTO	Application–Layer Traffic Optimization (IETF working group standardizing a protocol [APY12] for the requirements given in [KPS$^+$12])
AP	Access Point
API	Application Programming Interface
ARQ	Automatic Repeat Request
ATM	Asynchronous Transfer Mode
ASN.1	Abstract Syntax Notation One (standardized in [Int02])
BRAS	Broadband Remote Access Server
BER	Bit Error Rate
CaTV	Cable TV
CBR	Constant Bit Rate (Categorizes the traffic characteristic of a data stream. Typical for audio codecs and some video codecs.)
CCTV	Closed Circuit Television (commonly used for video surveillance)
CDF	Cumulative Distribution Function
CDN	Content Delivery Network
ConEx	Congestion Exposure (A concept described in [Bri08, MB12], and an IETF working group.)
CMTS	Cable Modem Termination System

CPE	Customer Premises Equipment
CSS	Cascading Style Sheets
DCCP	Datagram Congestion Control Protocol (as standardized in [KHF06])
DHCP	Dynamic Host Configuration Protocol (as standardized in [Dro97])
DOCSIS	Data Over Cable Service Interface Specification
DPI	Deep Packet Inspection
DTE	Data Terminal Equipment (Throughout this work, the term DTE is used for any device that is directly or indirectly connected to any Internet access network and is operated by a customer or user. This term is used independently from the access technology for all devices and comprises desktop computers, hand-held devices, mobile User Equipments (UEs) and appliances such as printers, Internet Protocol Television (IPTV) appliances, smart meters etc...)
DiffServ	Differentiated Services (as standardized in [NBBB98] and [BBC$^+$98])
DSL	Digital Subscriber Line
EBNF	Extended Backus–Naur Form (standardized in [ebn96]
ECN	Explicit Congestion Notification (As defined in [RFB01])
FIFO	First In – First Out
eNodeB	Evolved-UTRAN Node B (This denotes the new type of base stations in LTE networks)
ER	Entity-Relationship (model and diagram notation for describing data structures)
ETSI	European Telecommunications Standards Institute
FEC	Forward Error Correction
FT	Finish-time Transaction (see Section 2.4.1)
FTP	File Transfer Protocol (as standardized in [PR85])
FTTB	Fiber to the Building
FTTC	Fiber to the Curb

FTTH	Fiber to the Home
GPS	Generalized Processor Sharing
GUI	Graphical User Interface
HTB	Hierarchical Token Bucket (Packet queuing / Scheduling strategy in Linux)
HTML	Hypertext Markup Language
HTML5	Hypertext Markup Language version 5 (Ongoing standardization effort by W3C, HTML5 will standardize many things beyond the markup: scripting languages, scripting libraries, drawing, math markup, storage, document manipulation, networking, and more.)
HTTP	Hypertext Transfer Protocol (as standardized in [FGM$^+$99])
ICMP	Internet Control Message Protocol (As standardized in [Pos81a])
IAT	Inter-Arrival Time
IETF	Internet Engineering Task Force
IMAP	Internet Message Access Protocol (as standardized in [Cri03])
IMS	IP Multimedia Subsystem (3GPP standard [3GP08d])
IntServ	Integrated Services (Architecture for QoS, uses reservation with RSVP)
iOS	Apple iPhone OS
IP	Internet Protocol (version 4 (IPv4) standardized in [Pos81b], version 6 (IPv6) standardized in [DH98])
IPv6	Internet Protocol Version 6 (standardized in standardized in [DH98])
Netfilter	Linux firewalling framework
IPTV	Internet Protocol Television
ITU	International Telecommunication Union
Java	Java programming language
Jini	Java network architecture for distributed system (Originally developed by Sun Microsystems, currently managed by the Apache Foundation[1])

[1] URL:http://river.apache.org/

LAN	Local area network
LEDBAT	Low Extra Delay Background Transport (Transport layer protocol modification introduces by BitTorrent, described and standardized in [SHIK12a])
LTE	3GPP Long Term Evolution
M2M	Machine-to-Machine
MAC	Media Access Control
MCS	Modulation and Coding Scheme
MOS	Mean Opinion Score
NAT	Network Address Translation
NSC	Network Simulation Cradle (See [JM06])
NSIS	Next Steps In Signaling (IETF working group and QoS framework defined in [HKLdB05])
NSLP	NSIS Signaling Layer Protocol (Protocol for NSIS, defined in [STAD10] and [MKM10])
OFDM	Orthogonal Frequency Division Multiplexing
OS	Operating System
P2P	Peer-to-Peer
PC	Personal Computer
PGW	Public Data Network Gateway (in LTE-SAE)
PON	Passive Optical Network
PS	Processor Sharing (a scheduling discipline)
Pt2Pt	Point-to-Point
QoE	Quality of Experience
QoS	Quality of Service
QoT	Quality of Transaction

RCP	Rate Control Protocol (Proposal for alternative congestion control described and evaluated in [Duk07])
RLC	Radio Link Control (usually in the context of wireless radio, e.g. in 3GPP LTE defined in [3GP08b])
RR	Round-Robin scheduling
RRC	Radio Resource Control (usually in the context of wireless radio, e.g. in 3GPP LTE defined in [3GP08c])
RSVP	Resource Reservation Protocol (As described in [BZB^{+}97])
RT	Real-time Transaction (see Section 2.4.2)
RTT	Round-Trip Time
SAE	LTE System Architecture Evolution
SCTP	Stream Control Transmission Protocol (as standardized in [SXM^{+}00] and [Ste07])
SINR	Signal to Interference plus Noise Ratio
SNR	Signal to Noise Ratio
SIP	Session Initiation Protocol (IETF protocol standardized in [RSC^{+}02])
SLP	Service Location Protocol (as standardized in [GPVD99] and [Gut02])
SSH	Secure Shell (As defined in [LL06, YL06c, YL06a, YL06d, YL06b, CF06, GR06])
ST	Streaming Transaction (see section 2.4.3)
STB	Set-Top Box
TCP	Transport Control Protocol
TDM	Time Division Multiplex
TDMA	Time Division Multiple Access
TFO	TCP Fast Open (as explained in [RCC^{+}11])
TISPAN	Telecommunications and Internet converged Services and Protocols for Advanced Networking (Standardized by ETSI in the documents referenced in [ETS06].

TM	Transaction Manager
TOS	Type of Service
UPnP	Universal Plug and Play (Protocol for service discovery, configuration, and control. Introduced by Microsoft.)
UDP	User Datagram Protocol
UE	User Equipment (This term is coined by the 3GPP and refers to DTEs owned by the user and connected to the base station. In this work, usually the technology independent term Data Terminal Equipment (DTE) is used.)
UMTS	Universal Mobile Telecommunications System
UNC	University of North Carolina at Chapel Hill
VBR	Variable Bit Rate (Categorizes the traffic characteristic of a data stream. Typical for video codecs and some audio codecs.)
VCI	Virtual Channel Identifier
VLAN	Virtual Local Area Network (in this document, it refers to IEEE 802.1q tagged VLANs)
VM	Virtual Machine
VoD	Video on Demand
VoIP	Voice over IP
W3C	World Wide Web Consortium
WDM	Wavelength Division Multiplexing
WLAN	Wireless LAN (IEEE 802.11)
XCP	Explicit Congestion Control Protocol (Proposal for congestion control described in [KHR02])
XML	Extensible Markup Language
b_T	total “rebuffering” time of transaction T
B_T	number of “rebuffering” events of transaction T
t_B	duration of buffered data, before a stream resumes playback

C_T	number of traffic chunks in of transaction T
c_L	shape parameter for utility depending on loss frequency
c_v	shape parameter for utility depending on loss event auto covanriance
d_T	duration of transaction T
f_T	finish-time of transaction T
h_U	Imbalance of positive and negative values of utility functions. (For details see Section 5.2.2.)
l_T	relative frequency of packet losses (packets not meeting the latency requirement of transaction T)
Ψ_{lT}	autocorrelation of packet losses of transaction T
R	available data-rate of the access medium (in bits per second)
R_{UL}	available data-rate of the access medium in uplink direction (bits per second)
R_{DL}	available data-rate of the access medium in downlink direction (bits per second)
R_{exp}	expected data-rate of the access medium (in bits per second)
$R_{\mathrm{exp,UL}}$	expected data-rate of the access medium in uplink direction (bits per second)
$R_{\mathrm{exp,DL}}$	expected data-rate of the access medium in downlink direction (bits per second)
R_{Core}	Available data-rate on the core network
s_T	total size of transaction T in bits
$s_{T,\mathrm{DL}}$	downlink size of transaction T in bits
$s_{T,\mathrm{UL}}$	uplink size of transaction T in bits
s^r_T	remaining size of transaction T in bits
$s^r_{T,\mathrm{DL}}$	remaining downlink size of transaction T in bits
$s^r_{T,\mathrm{UL}}$	remaining uplink size of transaction T in bits
$t_{T,\mathrm{drop}}$	drop time of transaction T

$t_{C,\mathrm{drop}}$	deadline of traffic chunk C
$t_{p,\mathrm{drop}}$	proportional share of drop time
$t_{C,0}$	constant share of drop time
$t_{c,\mathrm{drop}}$	constant share of drop time
$t_{T,\mathrm{exp}}$	expected finish-time of transaction T
$t_{p,\mathrm{exp}}$	proportional share of expected finish-time
$t_{c,\mathrm{exp}}$	constant share of expected finish-time
Δt_{DL}	one-way delay of the access medium in downlink direction (bits per second)
Δt_{UL}	one-way delay of the access medium in uplink direction
Δt_{Core}	one-way delay of the core network
U_T	utility value of finish-time transaction T
$U_{T,\mathrm{drop}}$	minimal utility value for transaction T
$U_{T,\mathrm{ideal}}$	maximum utility value of transaction T for an ideal transmission
$U_{p,\mathrm{exp}}$	proportional share of utility value at expected finish-time
$U_{c,\mathrm{exp}}$	constant share of utility value at expected finish-time
$U_{p,\mathrm{ideal}}$	proportional share of the ideal (maximum) utility
$U_{c,\mathrm{ideal}}$	constant share of the ideal (maximum) utility
U_{b_t}	utility penalty for duration of rebuffering events
U_{b_N}	utility penalty for number of rebuffering events
$U_{p,\mathrm{drop}}$	proportional share of utility value at drop time
$U_{c,\mathrm{drop}}$	constant share of utility value at drop time
σ_t^d	σ parameter of distribution for the multiplicative knowledge error for time values
σ_s^d	σ parameter of distribution for the multiplicative knowledge error for size values

σ_U^d	σ parameter of distribution for the multiplicative knowledge error for utility values
σ_t^v	σ parameter of distribution for the multiplicative variation of time values
σ_U^v	σ parameter of distribution for the multiplicative variation of utility values
$e_{T,t}^d$	multiplicative knowledge error for time values of transaction T
$e_{T,s}^d$	multiplicative knowledge error for size values of transaction T
$e_{T,U}^d$	multiplicative knowledge error for utility value of transaction T
$v_{T,t}^d$	multiplicative variation factor for time values of transaction T
$v_{T,U}^d$	multiplicative variation factor for utility values of transaction T

List of Definitions

1 Introduction

Currently, broadband subscribers use the total access bitrate available to them only to a very small degree. Yet broadband subscribers still complain about the speed of their access, especially when multiple applications must share the available bitrate.

This work proposes and evaluates a system architecture for Internet access, which improves the user experience in situations, where the available bitrate is apparently not enough. The proposed system does not require changes to Internet protocols or devices beyond the broadband Internet access components, and is therefore easier to deploy than previous approaches.

This introduction is structured as follows: An overview of the current state of broadband access and its traffic characteristics is given in Section 1.1. A classification of Quality of Service (QoS) approaches in Section 1.2 summarizes the properties and characteristics of existing approaches. An overview of common problems in these approaches in Section 1.3 summarizes why none of these approaches have been widely adopted yet. Based on these common problems, Section 1.4 lists criteria and goals for the QoT approach presented in this work.

1.1 Motivation – Reasons for Quality of Service

The German Federal Network Agency (Bundesnetzargentur) [Bun11] reported in 2011 that there are 27 million broadband subscribers in Germany. The average bitrate of a broadband subscriber is around 6 Mbit/s, and, on average, each broadband subscriber transmits 12 Gbyte of data per month. This means that the utilization of the broadband access (predominantly Digital Subscriber Line (DSL) in Germany) is far below 1%.

Users are requesting these high bitrates because it makes a noticeable difference for them in many services or applications. For example, downloading a large file will finish earlier when the available bitrate is higher. The user can select higher quality versions of videos if the available bitrate is high enough. This means that in these cases, the user experience depends on the maximum bitrate. Another reason for the demand for high bitrates is that, without any QoS mechanism, the traffic of many applications interferes with each other. Interactive applications suffer when they have to share the available bitrate with instances of large downloads. For example, the latency (”lag”) in online games increases when there is a file-sharing program running at the same time. The voice quality of Voice over IP (VoIP) telephony calls becomes worse when there is an upload or download at the same time; or a streaming video starts to ”rebuffer” when there is a download at the same time.

1.1.1 Access Technologies

While DSL is a technology where there is a dedicated line to each subscriber, there are also access technologies, where the access line or the medium is shared among multiple subscribers. Examples of these technologies are Passive Optical Network (PON), Internet over Cable TV (CaTV), WLAN and cellular radio access. These technologies are currently gaining market share at the cost of DSL.[1,2] They offer a high peak bitrate, but this rate is shared among all connected subscribers. When one subscriber utilizes the peak bitrate, it is not available to another subscriber.

1.1.2 Multiplexing Gain

Multiplexing or aggregating the traffic of multiple subscribers onto a common line or medium “smooths” the traffic variability — it reduces the variance of the traffic, because the average bitrate increases strongly, while the peaks increase only slightly. Multiplexing, therefore, allows a higher utilization of this medium, which is called the *multiplexing gain*. The number of combined sources is called the *multiplexing factor*. It is well understood how multiplexing subscribers onto a shared link will help to increase the resource utilization [AMS82]. The bitrate of a single traffic source exceeds a given bitrate-threshold more often than the combined traffic exceeds the combined threshold.

It has long been known that the traffic of Internet access subscribers is bursty [LTWW94, JD05]. This means that there are long range and short-range correlations in this traffic. This burstiness leads to a lower multiplexing gain in comparison to uncorrelated traffic. For the low multiplexing factors (below one hundred) in Internet access, the subscriber’s service quality is already degraded at low average resource utilizations.

[1] [Bun11, page 34] shows that DSL had an annual growth of 3.5% in Germany from 2008 (20.9 million subscribers) to 2011 (23.2 million subscribers) while Fiber to the Home (FTTH), Fiber to the Building (FTTB), CaTV, Powerline and satellite together had an annual growth of 25% (1.8 million subscribers in 2008 and 3.5 million subscribers in 2011.) CaTV, Powerline and satellite are shared medium technologies, FTTH and FTTB can refer to Point-to-Point (Pt2Pt) Ethernet or more commonly to PON, which is also a shared medium technology.

[2] Mobile Internet in Germany had an average growth of data volume of 164% between the year 2007 (3.54 PB) and 2010 (65.41 PB), according to [Bun11, page 56]

1.1.3 Traffic Characteristics

However, the traffic requirements of Internet access customers are also heterogeneous. Some of the traffic is very delay sensitive; for example, VoIP, life IPTV, online gaming, and many other interactive applications. Other traffic requires a high peak data-rate, for example, web browsing, and many cloud computing applications. However, there are also applications that transmit a huge amount of traffic which is not delay sensitive. Examples are online backups, software updates, background data synchronization, Peer-to-Peer (P2P) file sharing and other P2P applications. Beyond that, receiving some of the transmitted data is not even required at all. Modern web browsers, for example, are able to prefetch some linked content without the user clicking on it, because the likelihood is high that this data could be needed later.

This heterogeneous traffic offers a new degree of freedom, because it allows an access network to differentiate between latency and utilization. When the access network would be aware of the type of traffic, it could improve its subscribers' service quality by prioritizing short and urgent data over high volume and less urgent data [EM92, STK99, YRK+07]. Some of the background applications mentioned transmitting high data volume could bear a delay of multiple hours without causing a significant disadvantage.

While such a system can improve the service quality for a user in a given situation, the main advantage for an operator is that it would be possible to increase the utilization of his network without degrading the users' service quality; i.e. allowing more paying customers in the same network.

For many years, it has been a research topic as to how to do this. As a result, there are many systems and approaches to improve the service quality. The following Section 1.2 gives an overview of these approaches.

1.2 Classification of QoS Approaches

The International Telecommunication Union (ITU) defines the term Quality of Service (QoS) in [ITU08] as

> "Totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service."

However, the term usually refers to systems that aim to improve those characteristics. For example, in [BDH+03, p. 121], the term QoS is defined to refer to all approaches that aim to "provide a better service quality than best effort" to at least one service or traffic class.

There are many different QoS systems and approaches. Each approach has to make several assumptions:

- about the type of the services
- about the type and quantity of degradations these services experience
- about the underlying network architecture
- about capabilities of the nodes involved

Briscoe et al. give an extensive overview on QoS techniques and their merits in [BBP$^+$14], with the focus on techniques that can reduce latency.

Many QoS approaches are often related to one or multiple QoS metrics. Examples for typical QoS metrics are

- the average bitrate
- the peak bitrate or a quantile of the bitrate distribution
- the average delay of packets
- the maximum delay or a quantile of the delay of packets
- the number of dropped packets

One has to distinguish whether a QoS approach measures such a metric, and uses it as its information source, whether the QoS approach influences this directly, or whether the QoS is designed to improve the metric by other means.

The following criteria are used to explore the space of possibilities for a system that aims to improve QoS or Quality of Experience (QoE):

- Which types of actions influence the network (Section 1.2.1)
- Which information sources does it use (Section 1.2.2)
- Which entities in or around the network are involved (Section 1.2.3)
- How small or big are the parts of the traffic that the approach recognizes to be different parts of the whole traffic on the network (Section 1.2.4)

The approaches to classify QoS have more objective criteria, which are the type of action that is performed, the type of information each approach uses for its decisions, and the entities that are involved.

1.2.1 Types of Action

QoS approaches differ in the type of action they perform. They can affect the order in which data is transmitted, and influence the transmission rate, the data volume a service or application sends or receives, and the path where the data is transmitted.

Reordering

Most QoS approaches act through schedulers. Schedulers prioritize queued packets, i.e. they modify the order of these packets. Typically, schedulers have multiple queues, packets are classified into these queues, and the scheduler serves the queue with the highest priority packets first. Such a prioritization of data can be performed on different layers. On the link layer, many Ethernet switches can perform prioritization as defined in IEEE 802.1d [80204], or they can prioritize based on other criteria. Wireless LAN (IEEE 802.11) (WLAN) stations can perform a distributed statistical prioritization using IEEE 802.11e [Soc05]. The 3rd Generation Partnership Project (3GPP) standards for 3GPP Long Term Evolution (LTE) define mechanisms for the prioritization of data packets in the Evolved-UTRAN Node B (eNodeB), the Radio Link Control (RLC) layer [3GP08b], the Media Access Control (MAC) layer [3GP08a], and in the Radio Resource Control (RRC) layer [3GP08c].

Layer-3 switches and routers can perform various types of queuing disciplines and prioritization. For example, Linux based routers can choose from a set of mechanisms and queuing disciplines built directly into the Linux kernel [HGM$^+$02].

In higher layers, gateways and application layer proxies could prioritize whole connections or application layer elements, for example, Hypertext Transfer Protocol (HTTP) objects. However, this is rarely applied; at least it is not publicly known that this has been used by now. Further, such reordering on the application layer can also be done by the applications on the end systems. In the early times of P2P file sharing, some P2P programs offered a configuration option to select other programs, such as VoIP telephony programs, that should not be affected by the data transmission. The P2P file-sharing program paused its data transmission on detecting an ongoing transmission in one of the other programs. However, such configuration options are rare in today's P2P file sharing programs. Instead, some applications make use of lower-than-best-effort transport protocols [WR11], especially the Low Extra Delay Background Transport (LEDBAT), which is implemented in the BitTorrent Delivery Network Accelerator (DNA)[3] and μTorrent.[4] LEDBAT and similar approaches are important research topics [RTV10, AG10]. LEDBAT has been standardized by the Internet Engineering Task Force (IETF) in [SHIK12a]. LEDBAT tries to detect if the capacity of a bottleneck is fully utilized by other traffic, and, by lowering its transmission rate, tries not to compete with it. The LEDBAT socket application, therefore, only sends when the bottleneck is empty. This effectively reorders the transmissions of application data, thus allowing the other application to transmit its data first.

Schedulers not only reorder packets, as explained above, often some packets are even dropped. Even simple First In – First Out (FIFO) schedulers drop arriving packets when they do not fit into the available buffer space. For connections using a transport layer protocol offering reliable

[3] `http://www.bittorrent.com/dna`

[4] `http://github.com/bittorrent/libutp`

transport using Automatic Repeat Request (ARQ), a thus dropped packet will not reduce the total data volume, since the ARQ mechanism will resend the packet later. Further to postponing this dropped packet, transport layer protocols such as Transport Control Protocol (TCP) react to dropped packets by reducing their sending rate, which further delays the transmission.

Reducing Transmission Rate or Volume

Other QoS mechanisms work by influencing the data volume or the data-rate. A common way to change the transmission rate is that automatic codec selection adapts to the current conditions in the network, such as load and latency. This adaptation can be performed in one or both of the end systems, or in the network. [Orl08] presents an approach where the adaptation to the network state is supported by the network. Popular applications that make use of such adaptation, measure the network characteristics instead of utilizing network support. Examples of such adaptation in the application are the automatic codec selection in Skype [DCMP08, BMM^{+}07], YouTube, and some Session Initiation Protocol (SIP) phones and networking devices [Cis07].

Multi-layer videos and scalable video coding allow a congested bottleneck to drop packets in one layer, with the receiver still able to decode the video at lower quality or resolution [SMW07]. Many video compression codecs use Intra-frames (I-frames) with independent picture information, as well as B-Frames and P-frames with information based on the I-frames [Uni07]. Compared with multi-layer videos, a simpler approach for improving the service quality of videos would provide for a congested bottleneck to drop P-frames and B-frames, making the impact on the video quality less severe than if I-frames were missing.

These approaches allow the network to reduce the transmission rate and total data volume of a video transmission. Dropping one layer of a multi-layer video, however, requires several contributing components, which are not widely deployed right now. Therefore, this technique is rarely used currently.

Another way to reduce the traffic volume is to deny service for new requests, called “admission control”. In radio access networks, admission control can deny some or all services to a user, to avoid overloading the radio cell or network [NS96]. QoS approaches using reservation, such as Integrated Services (IntServ) or NSIS NSLP, aim to avoid services that would cause overload [HKLdB05, STAD10].

In some cases, reducing the data-rate of elastic traffic will eventually reduce the data volume. For example, in P2P file-sharing applications, peers will download additional pieces from a host as upon completing the current piece. Moreover, the peers will download further pieces from someone else if this host is limiting its data-rate, which effectively reduces the data volume this peer is uploading.

Changing the Transmission Path

Traditionally, in radio access networks, handover is only used to select the cell with the best signal quality. A congested cell can also handover some of its UEs to another cell to reduce its load [VZ02], [3GP12, Chapter 5]. This changes the path of this UE's connections.

Content Delivery Networks (CDNs) can direct requests to another server, or to another location,to prevent a server or a data center from becoming overloaded [BCD$^+$02]. For example, the CDN delivering YouTube videos applies such mechanisms [TFK$^+$11]. The Application–Layer Traffic Optimization (ALTO) working group is designing a framework [KPS$^+$12, APY12] that allows clients to ask the network for the best peer, or communication partner, especially for P2P applications. Strictly, these approaches do not only change the path of the transmission, but also the corresponding node.

In general, controlling the path of a transmission is only beneficial in some specific scenarios, for example, when a mobile subscriber of a radio access network is in reach of multiple base stations. Otherwise, for fixed access lines, there is no alternative path.

1.2.2 Types of Information Sources

QoS systems need to know which action to perform for which traffic. This decision has to be based on information about the traffic. The information can be provided manually, either by the applications or services, or by the network. The entirety of this information and all surrounding circumstances is generally called "context". Only a small portion of this context can be obtained and used by computers.

Manual Configuration

In most cases, the information for QoS systems is provided manually, for example, by configuring network devices such as Ethernet switches to prioritizing packets from one switch port over packets from other ports. Such configuration can be done by the user, by an operator, or by the service provider.

There are also complex configurations, for which a network operator prioritizes traffic, depending on certain higher layers characteristics, such an IP protocol number and transport layer port number. Even classification of traffic with Deep Packet Inspection (DPI) is such a manual configuration, as the rules are not influenced by application or network state. Such classification can be performed with any packet firewall, such as Linux IPTables [HGM$^+$02], with DPI systems such as l7filter[5], and with application gateways. These systems and configuration options are widespread in many devices for home users. Especially, many residential gateways, such

[5] `http://l7-filter.clearfoundation.com/`

as the popular *AVM FritzBox*,[6] offer this.[7] A large number of commercial appliances also combine some or all of these approaches. These appliances are usually sold as *WAN optimizer* or *Network Booster*.

It is also common to manually signal information provided through the network; for example, when Type of Service (TOS) bits are set on packets that enter the network of an operator, and are used inside that network.

Application Knowledge

Manual configuration is often time-consuming and complex. It will generally not adapt automatically to new applications that do not match already configured patterns. Therefore, it is desirable to use the information that is provided by the applications. Here, applications also include the applications running in the domain of a service provider that act as a server in a client-server type service.

Since QoS is not widely deployed in the Internet, only a small number of applications can provide information usable for QoS. Linux and other operating systems offer the socket options *IP_TOS* and *SO_PRIORITY* [Lin11, Lin12] to configure the IP TOS bits in outgoing packets on this socket. Some applications, such as the NetKit implementation [Lin99] of *telnet*, [PR83] mark their packets or allow configuring the value of the TOS bits. A few applications, such as *OpenSSH*,[8] mark the packets, depending on whether the corresponding packet carries interactive or bulk data. Applications for protocols, such as Secure Shell (SSH), but also many computer games, have an inherent understanding of the context for each data transmission. The application terminating a SSH session establishes one or multiple channels, and it can explicitly mark each of the channels – among other parameters – as interactive. These parameters are exchanged between SSH client and server. Client and server, therefore, share this knowledge, and can mark the packets in both directions. For some protocols, e.g. SIP, application layer gateways are able to derive this knowledge from the protocol.

Protocols and architectures where only one endpoint of the connection has knowledge about the context, can only attribute traffic originating from the side having this knowledge. For example, a generic web server does not know in which context its communication partner (a web browser) performs a request. HTTP could be extended as to signal such information to a web server, but no such proposal has currently been widely deployed.

[6] http://www.avm.de/de/Produkte/FRITZBox/

[7] http://www.avm.de/de/Service/Service-Portale/Service-Portal/Praxis_und_Tipps/gaming_priorisierung.php

[8] http://www.openssh.org/specs.html

Network State

The network can also provide additional information for QoS systems. Relevant values are information about the connection status and information about the network load. The connection status can be the available bitrate, the link quality or packet error rate, and similar measures. Values describing the network load are, for example, the bitrate used, the queue length, or the number of active connections. This information is obtained either directly or indirectly. Depending on where the information is obtained and where it is used, it has to be signaled to the device or entity using it.

Information about the connection status is especially relevant for radio access networks. There, the Signal to Interference plus Noise Ratio (SINR), a metric for the link quality, is used to adapt to the current channel by choosing the best Modulation and Coding Scheme (MCS) for the current channel situation. It is also used for opportunistic scheduling algorithms that improve the overall throughput by preferring UEs with good channel conditions [GV97, KK12].

Using and signaling the network load directly is a common concept. An early approach was to send Internet Control Message Protocol (ICMP) *Source Quench* messages to hosts that contribute to congestion in any part of the network [Pos81a], [Bra89, Section 3.2.2.3]. Several variants and improvements for TCP also signal the network load. Explicit Congestion Notification (ECN) allows the network to inform the sender of a TCP connection that this connection is causing congestion [RFB01]. Explicit Congestion Control Protocol (XCP) [KHR02] and Rate Control Protocol (RCP) [DM06] are proposals to use explicit information about the network load to control congestion.

Deducing the network load indirectly is an even a more widespread concept, since the congestion control of TCP does just that. TCP congestion control indirectly informs the sender of a connection about congestion on the path to the receiver by reporting lost packets [JBB92]. Congestion Exposure (ConEx) is an ongoing effort by the IETF to further signal this congestion information to the sender and all intermediate nodes.

1.2.3 Involved Entities

Another criterion for classifying QoS approaches is to distinguish which entities assume an active role to improve the service quality.

These entities can take several roles:

- Measurement of information
- Performing an action
- Deciding on which action to perform
- Forwarding or aggregation of measurements or decisions

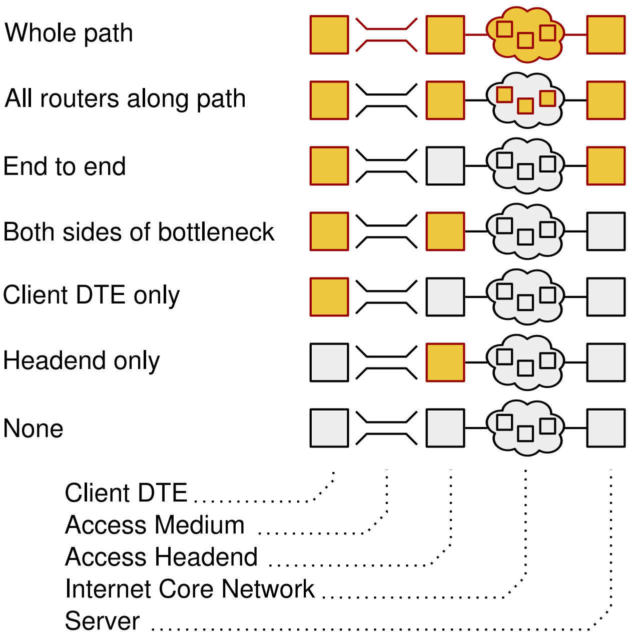


Figure 1.1: Possible combinations of the entities involved

For a typical connection of an access network subscriber, the following entities are involved:

- Client Data Terminal Equipment (DTE)
- Access network consisting of access medium and access headend
- Core Network consisting of routers, switches, and other forwarding devices
- Server, as the communication partner

Each of these entities could contribute to a system that improves the subscribers' service experience (see Figure 1.1). In the following, the different types of possible combinations of involved entities are explained generally, with examples of corresponding QoS approaches.

None When there is no QoS system deployed, there is no entity involved. All packets are handled according to "best-effort". End-to-end transport layer protocols, such as TCP, implicitly offer some service quality by reacting to cross traffic. TCP is defined so that all TCP connections on a congested link achieve the same long-term average data-rate. In [HFPW03], this fairness criterion is generalized onto other protocols.

End-to-End Using and adjusting the end-to-end transport protocols can also be regarded as a QoS approach.

The LEDBAT congestion control algorithm [SHIK12b] for TCP is called a "scavenger" service, as it tries to utilize a link only when it does not detect cross traffic congesting this link. In this protocol, only the end systems take an active role. In the IETF ConEx working group, there are ideas to generalize this concept and adjust the congestion control algorithm of each connection to reflect the service requirements for this connection [MB12]. Low priority services are handled by using the "scavenger" service; other services are handled by using one of the common congestion control algorithms, and high priority traffic might use an even more aggressive algorithm. The ConEx concept is not strictly end-to-end, as it also requires policer devices at the edge of the network that enforces compliance of the end systems. [BWC12, MB12].

It is obvious that all end-to-end approaches are slow in reacting to changes in the network, such as a change in the data-rate of the cross traffic. The cross traffic interferes with the packets of a connection, which has influence on the queuing delay of these packets and may change the drop probability. The receiver of the connection does not notice the change until the packets arrive, which were interfering with the cross traffic. The receiver then has to forward the information back to the sender. Because it is usually not possible to derive a change in the network by observing a single packet, it usually takes multiple Round-Trip Times (RTTs) until the connection has adapted to the change.

One end system only Under the assumption that most of the data is transmitted using adaptive transport layer protocols such as TCP, it is also possible that one side of the connection can influence the sending rate of its communication partner, and therefore indirectly influence both directions of the transmission. This can be done by influencing the TCP behavior on one of the end systems, or by limiting the data-rate or packet rate on an intermediate node, for example, the subscriber's residential gateway.

A *Congestion manager*, described in [BS01], is an approach that influences the TCP stack on the end system. It is extended, substantiated, and evaluated in [Egg04]. It proposes to share information between multiple TCP stacks on an end system to improve sharing of the available data-rate.

To improve the service quality, this kind of approach has to be deployed on the client DTE. On the server, such an approach can help to handle overload on the server, or the server's connection to the Internet. However, it cannot improve the service quality, as it is unaware of the status of the user's bottleneck.

On intermediate nodes, such as the residential gateway, scripts such as the Wondershaper[9] utilize the Linux traffic control with rules to prioritize among data streams, and limit the data-rate of certain streams [HGM+02]. Usually, such an approach requires the user to configure the priorities manually for different devices and services, for example by specifying TCP and UDP port ranges that shall be prioritized.

Headend only Similar to traffic shaping on residential gateways, rules to prioritize traffic can also be applied on the headend. Without the knowledge of the user about the requirements

[9] http://lartc.org/wondershaper/

of the traffic, it is usually necessary to determine the type of transmission by analyzing the content of the packets. This is usually referred to as Deep Packet Inspection (DPI).

Such an inspection and prioritization can generally only be done at the edge of the network, since routers in the core network carry too much traffic, and analyzing the content of packets would be too complex.

Besides the practical challenges and problems of DPI, it is also controversial when it is done outside the subscriber's domain or control – first, because of privacy concerns when the operator analyzes the content of the transmitted packets. Second, because the prioritization criteria of the operator might differ from the user's requirements or criteria. This is an important argument in the current debate on *Net neutrality*.

Whole path or all Routers along the Path For the well-known approaches for QoS improvements, IntServ, and Differentiated Services (DiffServ), generally all nodes on the path between the communication partners are involved.

Approaches based on the reservation of resources, such as IntServ, generally require that a reserved path is set up from one end system to the other end system, including all components in the middle. Usually, the reservation is mapped to lower layer mechanisms, or the lower layer properties are considered in the reservation. Thus it can be said that not only the IP routers are involved, but also lower layer components, such as switches.

For DiffServ, all routers along the path could use TOS marks for scheduling. Also, lower layer components, such as Ethernet switches, can use these values. For DiffServ, it is not necessary that all components along the path need to use TOS values for scheduling. It is, however, necessary that the TOS values are forwarded unaltered.

Both sides of bottleneck The new approach introduced in this work involves components on both sides of the bottleneck.

1.2.4 Partitioning of Traffic for Differentiation

In all QoS approaches, the transmitted data has to be classified in order to perform traffic differentiation. This classification can be performed on small chunks of data, (e.g. for each single packet) or on large chunks of data (e.g. all data of one user).

How the transmitted data is partitioned for the classification, is especially relevant when these partitions are attributed with some of the classification output. Such an attribution is necessary to signal the classification outcome from one network entity to another entity. Traffic granularities[10] for QoS approaches on the Internet access are:

[10] This granularity should not be confused with QoS granularity, which refers to the granularity of differentiation parameters, for example, how many different service classes are available. This definition is, for example, used in [YYP04, SM03]. This QoS granularity is not relevant in this section. QoT uses fine grained differentiation

- Subscriber: As of today, it is still a common case that all packets of one subscriber, of one server, or of one switch port fall into the same QoS class. Most switching hardware implements this approach and manuals explain this as the first approach to configure QoS, as for example, in Cisco switches in [HM03, Chapter 13].
- In radio access networks, QoS parameters are configured for each radio bearer, see [KN04, Chapter 7]. Typically, there is only one radio bearer per subscriber. However, it is sometimes possible to set up multiple bearers for one subscriber. Each bearer has its own QoS parameters, and applications can choose to open their connections on one of them. Such bearer management is implemented, for example, in Nokia's Qt library [Nok10]. A similar setup is used for DSL "triple play"[RFW06]: three different IEEE 802.1q [Soc11] Virtual Local Area Networks (VLANs) are set up for voice, TV/video, and Internet. The VLANs are then statically prioritized, for example, in products of Thompson [Tho08].
- Transport layer connection: For today's Internet traffic, differentiating transport layer connections would be an intuitive approach, because there is an obvious relation between one TCP connection and one application or one user interaction in an application. With the Resource Reservation Protocol (RSVP) for IntServ, for example, it is possible to identify packets using their transport layer connection (network address, transport layer protocol, and transport layer port), see [BZB$^+$97].
- Packet: Used in DiffServ, for example, and most systems that classify the traffic using DPI, performing service differentiation on individual packets. Even when the DPI classification requires stateful matching of packets to transport layer connections, these packets are individually flagged and prioritized by a packet scheduler unaware of these connections.[11]

How the transmitted data is partitioned in a QoS approach is often determined by technological constraints. For example, for Ethernet switches without any capability to analyze the transported packets or data, the originating switch port is the only feasible criterion for differentiation. This information is also often signaled to other Ethernet switches using 802.1q VLANs. Approaches for QoS improvement on general IP networks have to deal with the properties of IP: Packets can be routed along different paths, a single IP router might, therefore, only see a fraction of the packets that belong to a connection. For two further reasons, IP routers should generally only regard the IP header of each packet, and treat each packet separately: not to break the end-to-end paradigm [SRC84], and to keep the computational complexity of the router, and

attribution, which will be discussed in Section 2.5.

[11] The usage example for the Open Source L7-filter DPI solution for the Linux kernel uses Netfilter firewall marking on the flows. However, it uses these markings to filter packets into different queues, and to use a Hierarchical Token Bucket (HTB) scheduler setup to schedule the packets from these queues: `http://l7-filter.sourceforge.net/L7-userspace-example`

therefore its cost, low.

1.3 Problems of QoS

As a conclusion for the previous Section 1.2, it can be said that a number of approaches, standards, and products aim to improve the service quality. However, none of these approaches, standards, or products is available for the majority of Internet access subscribers.

On one hand, the Internet access brings the right preconditions for QoS approaches, thus promising high performance gains:

- The available data-rate is limited, and it is expensive to increase it
- Each transmission's requirements are defined by the subscriber's needs. Therefore, the knowledge about the transmissions is most likely available on the subscribers' devices.
- The traffic requirements are heterogeneous. Therefore, there is a degree of freedom that can be exploited.
- The number of subscribers sharing the medium, and therefore the number of parallel transmissions, is low enough to differentiate between transmissions. Differentiating transmissions on an Internet core router, for example, would require an unreasonable amount of memory.

On the other hand, several aspects that might be a reason why none of the QoS approaches have been established for the majority of Internet access subscribers. The individual reasons why an operator has not been established are usually not publicized. The main reason might be the cost for introducing such a system. In addition, problems that arise from the involvement of multiple parties and political reasons are important.

1.3.1 Cost

In comparison to the total cost of a network, the additional cost for a QoS system are low. However, such a system might not provide enough benefit to compensate these cost.

In core networks with a high level of traffic aggregation, and thus a lower level of burstiness, the network can be utilized to a higher percentage. In this case, the achievable gains from a QoS system are comparatively low. In addition, core networks are usually optimized to handle a huge amount of packets by minimizing the effort to process a single packet. Therefore, service differentiation is expensive in comparison to the low processing cost in core networks. Especially mechanisms that require to keep the state for each flow are disproportionately expensive in the core network.

In a testimony about *Net neutrality* to the Commerce Committee of the Unites States Senate

[Bac06], Gary Bachula stated that it is “more cost effective to simply provide enough bandwidth”, than it is to provide the same QoE using service differentiation. He further states, that in the Internet2 project,[12] none of the evaluated QoS approaches is in use.

On end systems and in applications, the cost of a QoS system in terms of computational complexity are negligible, since the transmitted data is usually processed, analyzed, stored, or displayed to the user. These operations are usually computationally more complex than the QoS system. For an application, the cost of a QoS system are mainly the cost of implementing it. Typically, application developers try to avoid additional complexity. Therefore only very few programs offer support for a QoS system.

Examples for approaches where the application complexity hinders further adoption is the QoS framework IP Multimedia Subsystem (IMS) [3GP08d] for 3GPP networks, and generally, the Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN) [ETS06]. Further to the complexity of the system itself and the number of required components in the operator’s domain, it is also complex to implement these approaches into applications. Often, the application would even need to modify its protocol, for example, to switch to SIP as the underlying protocol. Although there are publicly available libraries[13] to use IMS in applications on smartphones, none of the popular apps uses this framework. From the current situation, it can be assumed that IMS might be deployed and used only for regular voice telephony, and perhaps for further services provided by the corresponding operator, such as live TV.

1.3.2 Multiple Parties

Another hindrance for QoS systems is that, in most cases, multiple parties would need to cooperate to achieve a significant gain. For example, core network operators do not trust the QoS differentiation of packets specified by their customers. Instead, IP transit networks will often even delete the TOS field in the IP header. Frequently, instead of forwarding these fields, core network or transit network operators use these to differentiate between different customers.

This probably explains why many corporate networks deploy a QoS system internally, but not externally. Internally, there is only a single party controlling the relevant applications, all devices, and all network components.

For Internet access, each party is waiting for all other parties to deploy a QoS system first: the application developer does not implement QoS systems because the network does not support it. The access network operator does not deploy it because the core network does not support it, and because neither the access network operator nor the subscribers directly benefit from it.

[12] `http://www.internet2.edu/`

[13] `http://code.google.com/p/imsdroid/`

The transit network operator sees no need for QoS systems, because the transit network is rarely overloaded due to its high aggregation level and the robust dimensioning.

It is therefore important that all involved parties need to have some immediate benefit from deploying the QoS system. For most of the existing QoS approaches, multiple parties need to cooperate.

1.3.3 Political Reservations

There are several reasons, why it might not be desirable to deploy a QoS system in the Internet.

In [Bac06] Gary Bachula states that one of the major hindrances for QoS approaches is that the network would have to be "re-architectured" to support new or improved services.

It is also questionable, whether IP packets should have different priorities, or whether a discrimination against certain services would be desirable. This question is a central issue in the debate about *Net neutrality*.

1.4 Objective of this Work

The objective of this work is to design and evaluate a new approach for improving the scheduling in the Internet access. Existing QoS approaches, standards, and products are not widespread in use. This new approach, called Quality of Transaction (QoT), tries to avoid the problems these QoS approaches have (cf. Section 1.3).

This new approach has to use metrics that fulfill three classes of criteria:

- Conceptional criteria:
  - It has to be possible to measure and quantify the metrics in Internet protocols and networks.
  - It has to be possible to create a system that can actually use this metric in scheduling decisions.
  - The architecture of such a system has to be in such that its adoption is not likely hindered by one of the problems listed above.
- Technical criteria:
  - A system has to be able to improve this metric significantly.
  - This improvement has to be robust to uncertainties in the measurement and the quantification of the metric.
  - It has to be robust against changes in traffic characteristics.
  - It has to work in a wide range of network topologies and network properties.

- Psychological criteria:
  - The metrics have to represent the user's view. They have to reflect the user's perception of the quality of the network and services properly.
  - Improving these metrics has to improve the user's valuation of the network's quality.

QoT defines transactions. From a user's perspective, a transaction represents a user-observable result. From a network perspective, a transaction is all transmitted data that leads to this result. This means, transactions are made to be identifiable for network equipment. An example of such a transaction is displaying a complete web page in a web browser. Transactions are defined and explained in Section 2.4.

Based on these transactions and from a user's perspective, metrics are defined, which represent important factors in the user's perception. In the example of displaying a web page, the relevant metric is the finish time, thus the time until the web page is fully loaded and displayed. These metrics are combined with utility values that can be used in packet networks for scheduling. In contrast to metrics such as Mean Opinion Score (MOS), these utility values have both a meaning from a user's perspective and are measurable by networking equipment.

Both types of criteria are independent. Since the metrics for QoT are chosen from a user's perspective (see Section 2.5), it is likely that the psychological criteria are met. This work focuses on the conceptional and technical criteria.

It is unlikely that questions concerning the psychological criteria can be answered without a running prototype of such a system, because there is an inherent feedback loop between the user's actions and the system's behavior.

Therefore, it makes sense to first design a system that fulfills the conceptional and technical criteria, and validate the psychological criteria afterwards. By psychophysical studies, the metrics and definition of utilities can be tuned to the system's metrics matching the user's perception as closely as possible.

With this order, it is crucial to add another technical criterion: The system has to be robust against the different parameterization of the metrics and utilities.

1.5 Structure of this Work

The remainder of this dissertation is structured as follows:

Chapter 2 introduces the base concepts of the new QoS approach named QoT. It focuses on meeting the conceptual criteria listed above, thus avoiding the problems of existing approaches.

Chapter 3 introduces an architecture based on these concepts. The components in this architecture, their design goals, and the principles of operation are also presented in this chapter. QoT is examined according to the criteria given in Section 1.3. This mainly includes the realization aspects that are outside the scope of performance evaluation.

The technical criteria are evaluated using performance evaluations. These performance evaluations are conducted using event-based network simulation. In Chapter 4, the simulation environment and in Chapter 5 the simulation model to test the QoT approach are explained. The most important parts of the simulation model are the realistic modeling of Internet traffic and the users' valuation or utility, which are introduced and explained in Section 5.2 and 5.3.

In Chapter 6, the technical criteria of QoT listed above are evaluated. The performance and robustness of QoT are thus evaluated in several situations, and in addition, the main factors influencing its performance are identified. Chapter 7 contains summary, conclusion and outlook of this work.

2 Concepts for Quality of Transaction

As introduced in Chapter 1, a new approach for a better QoS is needed, considering that all existing approaches either lack some features, or did not gain general acceptance for one of the reasons given in Section 1.3. The concepts for the new approach Quality of Transaction (QoT) are developed and explained in this chapter.

This contains, first, the assumptions and requirements for QoT. The fundamental idea of QoT is described based on these assumptions. The two key concepts of QoT are transactions, which are explained in Section 2.4, and utility functions, which are explained in Section 2.5.

2.1 QoT Concept Overview

Figure 2.1 shows an overview of the entities involved, and the signaling required for the QoT approach.

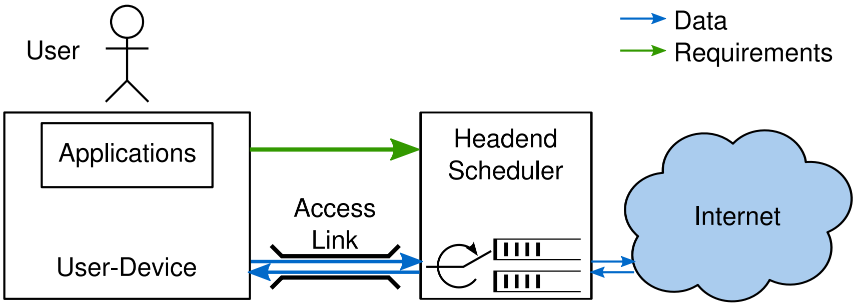


Figure 2.1: QoT system overview

QoT is deployed on both sides of the access link. On one side of the access link, there are the user's devices. These devices, called Customer Premises Equipment (CPE), comprise the DTEs that terminate data transmissions and provide the results of the services to the user. The CPEs also include optional forwarding devices, such as the user's residential gateway.

On the other side of the access link are the operator's devices. QoT has to be deployed on the first device handling the user's packets that controls the resource allocation on the access link. In the following, this node will be called "headend", independently of the access technology. Examples for headends are the Cable Modem Termination System (CMTS) for Data Over Cable

Service Interface Specification (DOCSIS), the Broadband Remote Access Server (BRAS) for DSL, and the Public Data Network Gateway (PGW) for LTE.

Definition 2.1: ***Headend***
The headend is the first hop from a subscriber to the Internet that controls the resource allocation on the access link.

Scheduling in the downlink direction is performed by the headend. Packets in the downlink direction are queued in the headend. For the uplink, packets are queued on one of the user's devices before being sent over the access link, for example, on the residential gateway. It depends on the access technology, whether the headend performs the scheduling of the queued packets, or the headend only performs arbitration between different users.

As the users provide all the information used by the scheduler for scheduling, this information needs to be signaled from the user to the headend. The information is sent from the user to the headend, while the headend refers to the packets or transmissions that are sent from the Internet over the headend to the user. The headend, therefore, has to match these packets or transmissions to the information from the user. This matching is performed using transactions, which are thus responsible for the name Quality of Transaction (QoT).

Transactions are defined from a user's perspective and from a network's perspective. From the user's perspective, a transaction is all transmitted data that lead to a result. Such a result is, for example, a downloaded file that allows displaying a complete website, or playing a video. From the network's perspective, a transaction is a set of sections of transport layer connections. The description of these sections of transport layer connections is formulated and signaled to the headend, together with the service requirements for this transaction. The headend uses this formulation to match the incoming packets to transactions. For scheduling, the headend uses the signaled requirements to decide which of the currently buffered transactions has to be served first. The transactions and the different transaction types are explained in Section 2.4.

In addition to transactions, the formulation of the user's requirements for transactions is the second important concept of QoT.

The base metric of these requirements is the value of a transaction for the user. The requirement is not just a scalar specifying the value in optimal network condition, but instead a function that specifies the value depending on how and when the corresponding data is transmitted. This definition of *utility* is explained in Section 2.5. The scheduler then has to maximize these values.

2.2 QoT Properties

QoT focuses on improving the scheduling on the access link. This limits the number of parties involved to two, but still promises good results because the access link is usually the bottleneck.

To have the system to cover both sides of the bottleneck, only the operator of the access network and the DTE need to be involved. For the system to be economically successful, these two parties need to profit from it. Since the subscriber has a contractual relationship with the operator of the access network, the business process for deploying the system is simple. When the operator decides to deploy QoT, he can provide his customers with a QoT enabled device, firmware images, or applications.

An operator of an access network profits by using a shared medium for multiple subscribers, because such a system promises a higher utilization of the shared medium, without reducing the service quality for the customers. A subscriber profits because his important services are preferred over his less important services, and therefore, the requirements of his services are more likely to be met.

QoT lets the user decide which services are to be prioritized. Instead of improving low level networking metrics, the goal of QoT is to improve the users' experience. This is only possible, when the requirements for the transmissions are specified by the user. This is also important in the context of the current debate on *Net neutrality*. In general, the user as a person is not able to formulate the requirements for all transmissions, other than some configuration options. Instead, it is necessary for the user's devices, operating system, and applications to provide this information.

Since the scheduling for the downlink direction is performed at the operator's side of the access link, the users' requirements need to be signaled to this scheduler, and the scheduler performs its scheduling accordingly. Practically, the approach of QoT can be summarized as a framework that allows the subscriber of an access network to upload prioritization rules for his connections to the access network headend. The access network scheduler uses these rules to prioritize among the connections of this subscriber. The access network can also use these rules to prioritize among connections of different users.

QoT does not require changing the application logic or the behavior of applications. The applications should provide information about the requirements of their transmissions, which does not require changes in the applications behavior. There are no states or events of QoT that applications need to react on. In addition, QoT has to handle new and future services. Therefore, QoT is not tailored to today's services and traffic patterns, but is instead generic and flexible enough to handle current and future traffic patterns.

Classification of QoT according to the criteria in Section 1.2:

- Types of action: Transmissions are reordered. In the uplink direction, the client DTE or residential gateway sends the most important transmissions first, and buffers the less important ones until there are enough free resources. In the downlink direction, this prioritization and buffering is done in the access network.

- Types of information: QoT can benefit from all types of information that contribute to the knowledge about a transaction. It is, however, important that all information is collected on the subscriber's side of the access network. Therefore, all information that the user, his devices, and his applications can provide is relevant. The most important source are the applications. Examples for such application knowledge are the estimated or previously known size of a transaction, or whether a transaction is performed in the background, or the result is immediately visible to the user.
- Involved Entities: Only the client DTE and the headend are involved

2.3 Assumptions and Requirements

It is assumed that, in the near future, the access network will continue to be the main bottleneck for private Internet customers. QoT is therefore designed to improve the service quality for end users by improving the access. For improving the service quality, QoT relies on the help of the DTEs and the access network.

2.3.1 Network Structure and Bottleneck

Other approaches to QoS often regard sender, receiver and all intermediate routers and switches to be relevant, see Section 1.2. QoT only regards the nodes on both sides of the access link to be relevant for the system. Figure 2.2 shows the typical nodes involved on the network layer, when an Internet subscriber connects to a server of a service provider.

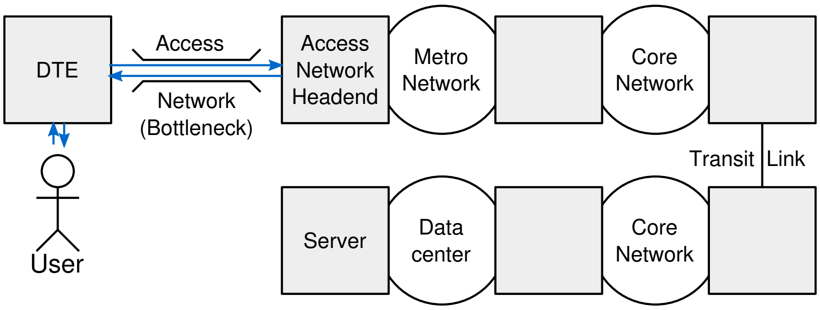


Figure 2.2: Typical Internet architecture on IP layer.

For fixed access, there are costs for the last mile of each potential customer. Therefore, there is an extreme cost pressure to use the cheapest possible option and components. For the backhaul or core network, the higher aggregation allows for higher resource utilization, and therefore, the cost per subscriber is lower. There, reliability becomes an important matter.

Wireless access networks have base stations, or access points, shared between many subscribers. However, the wireless resources are scarce and expensive, and all promising methods to increase the data-rate are utilized [Ver00]. In general, the cost per bit on the last mile is far more expensive compared to backhaul, core network, or links to data centers.

There are publications that claim that a large proportion of traffic is not limited by the access link, but by the application [SUKBEN05, SCUKB07]. The measurement methodology can only consider long-lived TCP connections, which are often from P2P applications [HLM+04]. However, delay sensitive traffic is usually transmitted using short-lived connections. Application layer throttling of the data-rate is a workaround for the missing QoS mechanism in the access that is often used today. It tries to reduce the interference between the corresponding traffic and the delay sensitive traffic.

The outcome is that the access link will continue to be the main bottleneck for Internet customers in the foreseeable future.

2.3.2 Access Technologies

QoT is designed to be independent of the access technology. There are two major types of access technologies:

- Dedicated access medium, where there is a dedicated line for each subscriber. In most cases, the available data-rate of the access line is constant over time. The subscriber can utilize the full data-rate of the access line. Since today's applications generate bursty traffic, the average utilization of such access links is low.
- Shared access lines connect multiple subscribers at once. The available data-rate is shared among the subscribers. One subscriber may only use the full data-rate when no other subscriber is active at the same time. This aggregation leads to a higher utilization of the access medium, but when multiple subscribers transmit data at the same time, the data-rate for one subscriber is significantly reduced.

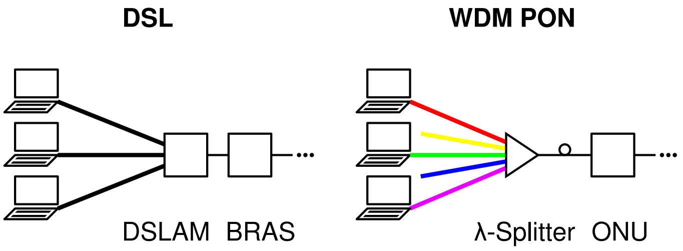


Figure 2.3: Examples of access technologies with dedicated access medium.

Figures 2.4 and 2.3 depict examples of access technologies with dedicated and shared access medium, and the corresponding headends for these technologies.

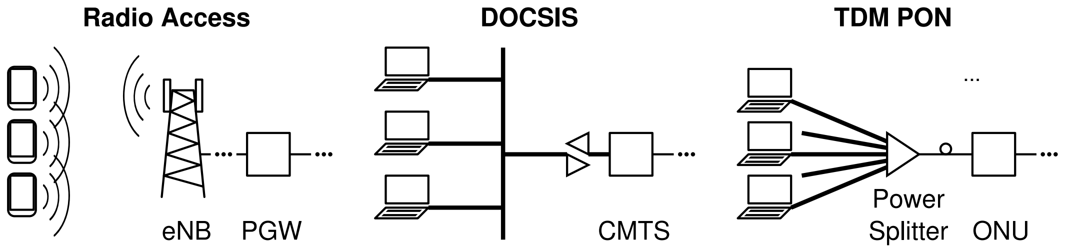


Figure 2.4: Examples of access technologies with shared access medium.

Dedicated access medium

Internet subscribers with a dedicated line usually need a high bitrate to satisfy the required peak bitrate of all applications. In 2013, the dominating technology for dedicated lines was DSL and it is common currency that one subscriber requires at least 6 Mbit/s for a good service quality, mainly because of some applications that require a high peak data-rate. The average data-rate of one broadband subscriber is more than two orders of magnitudes smaller.[1] Due to the resulting low average utilization, the applications of one subscriber will only rarely require more bitrate than available. The potential gain for any QoS approach is therefore limited in such cases.

For Internet subscribers with a dedicated line with insufficient capacity, no QoS approach can provide the peak data-rate for the applications that require it. However, for such subscribers, there are situations, where QoS systems can help. A common situation for Internet subscribers with a dedicated line of a capacity of 1 Mbit/s or less is that they have to stop downloads, file sharing applications, and other background tasks, to be able to watch videos, live TV or make a VoIP call without delays or disruption. This could also be done automatically. In fact, one of the design goals of QoT is to solve these situations automatically.

Shared access medium

Sharing the access medium among multiple subscribers is a cost effective way to provide a high peak data-rate, and at the same time, increase the utilization of the available data-rate. Access technologies sharing the access medium, such as Internet over CaTV (DOCSIS), Time Division Multiplex (TDM)-PON, and radio access, are currently increasing their market share [Bun11]. These technologies, however, cannot guarantee the peak data-rate for each user. With fair (equal-rate) scheduling, the obtained data-rate is the available data-rate, divided by the number of active subscribers. In mobile access networks, this bitrate also depends on the current channel condition. In fixed access networks, this bitrate is a “guaranteed data-rate”. However, this guaranteed data-rate is usually not sufficient for many applications.

[1] [Bun11] specifies the monthly volume of transmitted data of a German broadband subscriber as 12 Gbyte. This is equal to 36 kbit/s average data-rate.

In most cases, the applications competing for the available data-rate have different service quality requirements. Some of the applications might even run in the background, and are not impaired when their transmissions are delayed. In these cases, a QoS system can delay the transmission of background applications until an application requiring a high peak data-rate has finished its transmission.

When the access medium is shared between higher numbers of subscribers, the higher level of aggregation makes the combined traffic less bursty. As a result, the number of times the correspondingly dimensioned data-rate does not suffice for the application becomes less. The evaluations in Chapter 6 show that this effect starts at several hundreds of subscribers, which is more than the typical access technologies can handle.

Access technologies for QoT

QoT is designed to improve the service quality in access networks in all cases where multiple applications compete for the available data-rate, and the access network is not able to satisfy all at the same time. The potential gain for QoS systems is limited to dedicated access media with high bitrate and low utilization. The potential gain for a QoS system is also limited when the access medium is shared among a large number (several hundreds) of customers.

2.3.3 Usage Scenario

QoT assumes that the user is using a DTE, which is connected to an access network where QoT is deployed. The user interacts with applications on the DTE, which receive or transmit data over the access network. The DTE is connected either directly to the access network, or as one of a group of DTEs over a residential gateway. Such a scenario is depicted in Figure 2.5

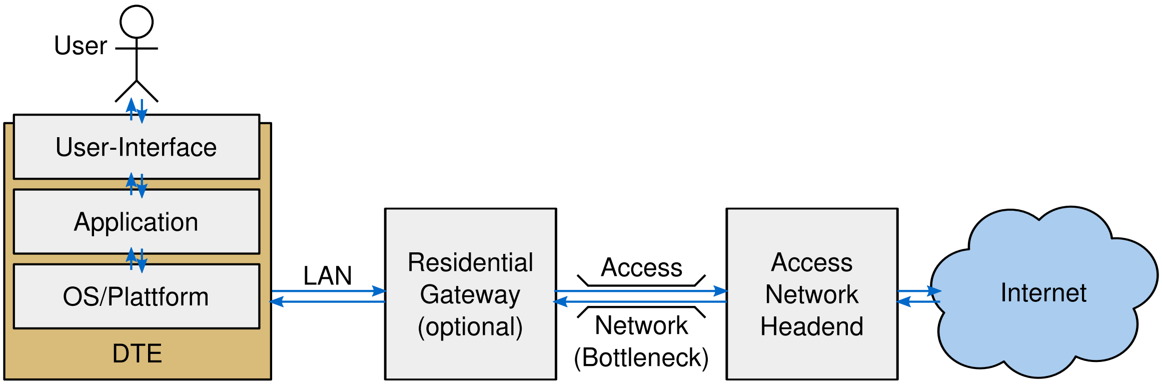


Figure 2.5: Scenario of the user's side of the access network for QoT.

The applications on the DTE are able to know the requirements for their network transmissions. For background transmissions, the requirements fully depend on the internal state of the application. The requirement is therefore known and can be used. For foreground transmissions, where the result is visible to the user, the real requirement depends on the user's expectation and his subjective judgment. The application can access a variety of context information that is relevant to predict the user's judgment or valuation. Section 3.5 lists the possible information sources that allow the application to know the requirements for its data transmissions.

An example:

- The application is a web browser.
- The DTE is a desktop computer.
- The access network is Internet over CaTV, where multiple subscribers share the same resources.
- The residential gateway is the IP router bundled with the cable modem, usually advertised as "Cable Modem Router".
- The headend is the CMTS.
- The backhaul of the CMTS is connected to the Internet.
- The web browser communicates with web-servers that are also connected to the Internet.

2.3.4 Signaling Direction

One important concept of QoT is that the signaling is unidirectional from the customer to the headend. The main reason for this is that the subscriber is the main source of the transaction information, and that the operator or the access network cannot contribute to the information that a user expects or the way he will perceive the service.

There are also reasons why it could be beneficial to signal in the opposite direction. The headend could signal information about the network load to the DTE. The applications could then adapt to the current load situation. The idea of ConEx tries to establish such feedback of the current load situation throughout the whole Internet, not just on the access network. In general, it is more flexible for the application to adapt, because the application has more influencing options. While the headend in the access network can only delay or drop a transaction, the application can also modify the transaction itself, for example, switch a video codec.

However, one of the principles of QoT is not to force applications to behave in a special way. So the application may keep its behavior and the application may implement its adaptation to the networks properties. While it might be beneficial for some applications to adapt to network properties, it is unrealistic to expect all applications to be able to adapt to all different kinds of access network types, different load situations, different transient load behaviors, and different load predictions. Therefore, the generic approach is to allow applications to perform such an adaptation, but not to expect any application to do it. For QoT, the applications only need to signal their requirements, and the access network headend has to respond accordingly.

It is possible to combine QoT with an approach, which signals load or congestion information to adapting applications. Therefore, there is no need for the QoT itself to include this.

To reduce obstacles for the implementation, modifications to existing code and systems of third parties should be minimized, as stated in Section 1.3. It is possible for Operating Systems (OSs) and platforms such as *Android* to support publishing the application's requirements, or even assume sensible defaults (see Section 3.5.3). It is more difficult on this level to provide support for adaptation in the applications.

Furthermore, for the downlink, an adapting application will not be able to react as fast as the scheduler will, because the application has to signal its communication partner to stop sending the data it is currently sending and start sending other data. On the contrary, the scheduler of the headend can immediately send the more important packet.

The communication partner in the Internet, for example, the web server that is currently serving a web page to the user, could provide additional information on the requirement of its data. For many protocols, such a server has some knowledge about the data it will be sending out. However, the server might have different priorities or preferences than those of the user. A service provider is not interested in maximizing the user's utility, but in making its service better than its competitors. Moreover, if content or service providers can take influence on the user's access network, this might violate *Net neutrality*, see Section 1.3.

2.4 Transactions

The basic idea of QoT is to bring the information relevant for the user's service experience to the schedulers relevant for the user's service experience. Those schedulers can use the information to improve the service quality, or increase the network utilization, for example, by prioritizing small and urgent data over large bulk transmits.

The user's overall service experience is the combination of the service quality of several separate services, results, and events. The QoT approach defines these – from the user's perspective, separate – items as "transactions", and tries to improve their quality.

Definition 2.2: ***Transaction***
A transaction is a user-observable result and comprises all transmitted data that leads to this result.

This definition of transactions is from the user's perspective. Therefore, a translation of such user-centric transactions has to be defined for a network centric view. This is explained in Section 3.2.1.

There are many different instances of transactions, and there are constantly new types of transactions, because the usage of communication networks is continuously extended to new fields.

It is therefore impossible to give an overview over all existing applications, and the way they use the network.

Instead, there are a few common types of results with a corresponding behavior on the network. Three types of transactions are defined, depending on the type of their results:

1. Finish-time Transaction: a result that is finished at a point in time
2. Real-time Transaction: a result that occurs over a period of time and cannot be delayed
3. Streaming Transaction: a result that happens over a period of time and can be delayed

These three transaction types are described in detail in the following sections.

2.4.1 Finish-time Transaction

The most common and intuitive type of result is when there is a single or a limited number of discrete events at which the result occurs. A simple example is an explicit download happening at one point in time. Transactions with such a result are Finish-time Transactions (FTs). The properties of the network have no influence over how the result looks like the network properties can only influence the point in time when the result is available.

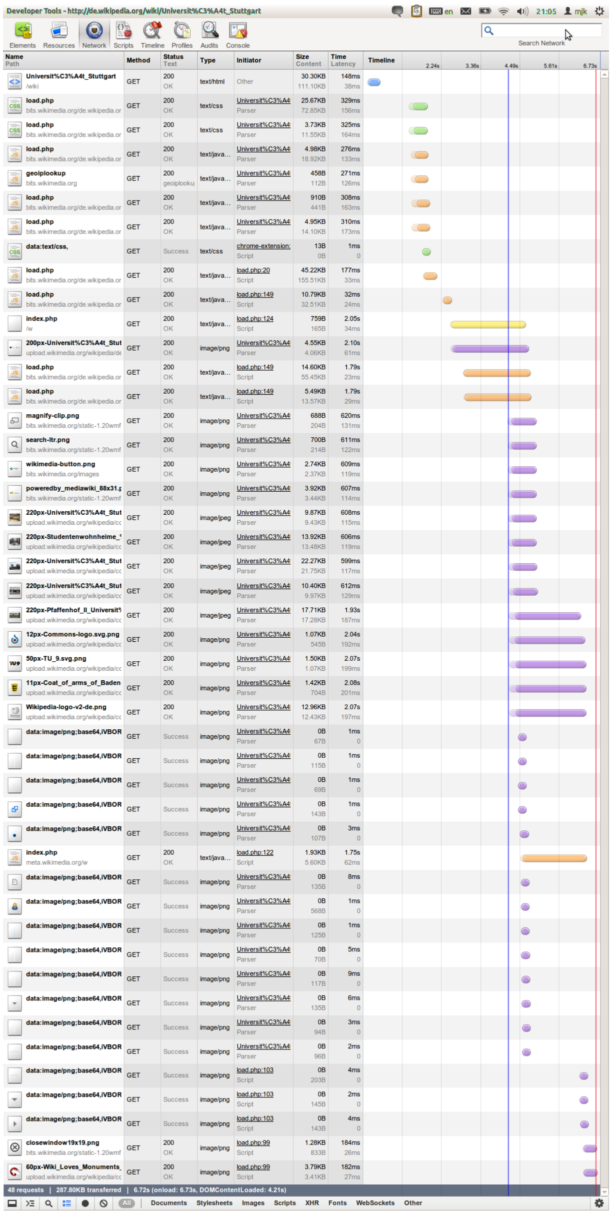
More examples of FTs are:

- Downloading or uploading a file with a web browser, a download manager, or a P2P program.
- Entering a level in an online game
- Sending an email, retrieving an email from the server, looking for new emails
- Doing an online backup
- Synchronizing a folder with a cloud storage folder
- Upgrading a program to a new version, e.g., a security-update

Often, one FT corresponds to one transport layer connection. For example, a download or upload of one file using HTTP or File Transfer Protocol (FTP) usually uses exactly one TCP connection. However, there are cases, when

1. one FT consists of multiple transport layer connections
2. one FT consists only of a part of one transport layer connection

Cases where one FT consists of multiple transport layer connections are quite common. Examples are web pages, which consist of a many elements that are retrieved over multiple TCP connections. An example of the elements contained in a simple web page is shown in Figure 2.6. Each line represents one HTTP element. More complex web pages, such as web pages of news sites, currently consist of several hundred elements. These elements are often downloaded over



Name Path	Method	Status Text	Type	Initiator	Size Content	Time Latency	Timeline
Universit%C3%A4t_Stuttgart /wiki	GET	200 OK	text/html	Other	30.30KB 111.10KB	148ms 38ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/css	Universit%C3%A4 Parser	25.67KB 72.85KB	329ms 156ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/css	Universit%C3%A4 Parser	3.73KB 11.55KB	325ms 164ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	Universit%C3%A4 Parser	4.98KB 18.92KB	276ms 133ms	
geoiplookup bits.wikimedia.org	GET	200 geoiplooku	text/java...	Universit%C3%A4 Parser	458B 112B	271ms 126ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	Universit%C3%A4 Parser	910B 441B	308ms 163ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	Universit%C3%A4 Parser	4.95KB 14.10KB	310ms 173ms	
data:text/css,	GET	Success	text/css	chrome-extension: Script	13B 0B	1ms 0	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	load.php:20 Script	45.22KB 155.51KB	177ms 33ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	load.php:149 Script	10.79KB 32.51KB	32ms 24ms	
index.php /w	GET	200 OK	text/java...	load.php:124 Script	759B 165B	2.05s 34ms	
200px-Universit%C3%A4t_Stut upload.wikimedia.org/wikipedia/de	GET	200 OK	image/png	Universit%C3%A4 Parser	4.55KB 4.06KB	2.10s 61ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	load.php:149 Script	14.60KB 55.45KB	1.79s 23ms	
load.php bits.wikimedia.org/de.wikipedia.or	GET	200 OK	text/java...	load.php:149 Script	5.49KB 13.57KB	1.79s 29ms	
magnify-clip.png bits.wikimedia.org/static-1.20wmf	GET	200 OK	image/png	Universit%C3%A4 Parser	688B 204B	620ms 131ms	
search-ltr.png bits.wikimedia.org/static-1.20wmf	GET	200 OK	image/png	Universit%C3%A4 Parser	700B 214B	611ms 122ms	
wikimedia-button.png bits.wikimedia.org/images	GET	200 OK	image/png	Universit%C3%A4 Parser	2.74KB 2.37KB	609ms 119ms	
poweredby_mediawiki_88x31.p bits.wikimedia.org/static-1.20wmf	GET	200 OK	image/png	Universit%C3%A4 Parser	3.92KB 3.44KB	607ms 114ms	
220px-Universit%C3%A4t_Stut upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/jpeg	Universit%C3%A4 Parser	9.87KB 9.43KB	608ms 115ms	
220px-Studentenwohnheime_ upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/jpeg	Universit%C3%A4 Parser	13.92KB 13.48KB	606ms 119ms	
220px-Universit%C3%A4t_Stut upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/jpeg	Universit%C3%A4 Parser	22.27KB 21.75KB	599ms 117ms	
220px-Universit%C3%A4t_Stut upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/jpeg	Universit%C3%A4 Parser	10.40KB 9.97KB	612ms 129ms	
220px-Pfaffenhof_II_Universit% upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/jpeg	Universit%C3%A4 Parser	17.71KB 17.28KB	1.93s 187ms	
12px-Commons-logo.svg.png upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/png	Universit%C3%A4 Parser	1.07KB 545B	2.04s 192ms	
50px-TU_9.svg.png upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/png	Universit%C3%A4 Parser	1.50KB 1.07KB	2.07s 199ms	
11px-Coat_of_arms_of_Baden- upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/png	Universit%C3%A4 Parser	1.42KB 704B	2.08s 201ms	
Wikipedia-logo-v2-de.png upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/png	Universit%C3%A4 Parser	12.96KB 12.43KB	2.07s 197ms	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 67B	1ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 115B	1ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 69B	1ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 143B	1ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 107B	3ms 0	
index.php meta.wikimedia.org/w	GET	200 OK	text/java...	load.php:122 Script	1.93KB 5.60KB	1.75s 62ms	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 135B	8ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 568B	1ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 125B	1ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 70B	5ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 117B	9ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 135B	6ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 94B	3ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	Universit%C3%A4 Parser	0B 96B	2ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	load.php:103 Script	0B 203B	4ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	load.php:103 Script	0B 145B	2ms 0	
data:image/png;base64,iVBOR	GET	Success	image/png	load.php:103 Script	0B 143B	4ms 0	
closewindow19x19.png bits.wikimedia.org/static-1.20wmf	GET	200 OK	image/png	load.php:99 Script	1.28KB 833B	184ms 26ms	
60px-Wiki_Loves_Monuments_ upload.wikimedia.org/wikipedia/cc	GET	200 OK	image/png	load.php:99 Script	3.79KB 3.41KB	182ms 27ms	

48 requests | 287.80KB transferred | 6.72s (onload: 6.73s, DOMContentLoaded: 4.21s)

Figure 2.6: Screenshot of Chromium web browsers developer Tools for `http://de.wikipedia.org/wiki/Universit%C3%A4t_Stuttgart`.

many different TCP connections [Cha10]. Since some of the elements originate from the same server, and browsers limit the number of concurrent TCP connections to a server,[2] the number of connections may be smaller than the number of elements. Evaluations in [Cha10] for popular web pages in 2010 cite that it is common to open more than ten TCP connections in a single web page.

HTTP is also an example for the second case, where a transaction is only a part of a transport

[2] On `http://meronymy.blogspot.de/2011/09/overview-of-browser-concurrent.html`, a list of links to the documented limits for concurrent connections shows that in 2011, browsers use 2 to 15 concurrent TCP connections.

layer connection. HTTP version 1.1 allows to keep the TCP connection open after transmitting one element. The next element may therefore already belong to another transaction. Another example is downloading an email over an already established Internet Message Access Protocol (IMAP) connection.

2.4.2 Real-time Transaction

The second and third types of transactions are results that are not finished at a discrete point in time. Instead, the result of these transactions is a process over time. A simple example of such a process is a VoIP telephony call.

In this case (RT), the network properties have an influence on the result at any time; however, this influence might be limited to a certain part of the result, while later parts are not affected. For example, a short overload on the network, causing some packets of a VoIP call to be dropped cause degradation of the voice quality only at this time. A few seconds later, the voice quality is normal again, because most audio codecs are able to recover from lost packets.

In general, each packet of a RT has to arrive at the receiver within a certain time span (the deadline). If it does not arrive in time, the value of this packet shrinks. Eventually the packet becomes useless. The user's valuation for the transaction depends on the number of packets that did not arrive within their deadlines. The network properties, therefore, have an influence on the properties of the result, the user experiences. The acceptable delay or deadline is defined by the application. When the application adapts to the current network properties, the deadline is one of the parameters that can be adapted. Without such an adaptation, and during the time in between adaptations, the deadline is fixed, and the influence of the network is limited to the number of packets that violate the deadline.

More examples for this type of transaction include

- Video telephony
- Live TV or radio broadcasts, which are, in contrast to voice or video telephony, unidirectional
- Remote desktop appliances transmitting all keyboard and mouse events to an application server and receiving screen image updates
- Sensor data: e.g. Closed Circuit Television (CCTV), temperature sensors, power meters, etc.
- Multi-player online games

Typically, RTs have a deadline in the range of milliseconds to seconds. In the Internet, RTs typically use User Datagram Protocol (UDP) or Datagram Congestion Control Protocol (DCCP) transport layer protocol, or they do not use any transport layer protocol at all, but instead use raw IP packets or Ethernet frames.

2.4.3 Streaming Transaction

Similar to RTs, Streaming Transactions (STs) are results that consist of a process over a period of time. The difference is how the application reacts on delayed packets. For STs, the application will not treat delayed packets as lost, but will instead wait for these packets. That means, for STs, the network properties can cause interruptions, and therefore influence the delay of the stream, but they do not influence the content.

In current Internet applications, RTs typically use UDP as transport layer protocol, while STs typically use TCP. Because of the high delays caused by head-of-line blocking in TCP, RTs are usually not possible with TCP [SK06]. Streaming applications that require all data to be received would need to implement some sort of reliability on top of UDP. Hybrids such as Stream Control Transmission Protocol (SCTP) currently have negligible share of traffic in measurements of Internet access data such as [San12a].

The applications usually have a buffer of at least multiple seconds, which means that the deadline for packets of these transactions are also typically in the order of multiple seconds [KA06]. However, for these transactions, it is indispensable that all packets arrive. Lost packets need to be retransmitted. If one packet is delayed, the receiver cannot use all of the following packets until the missing packet arrives.

Examples for STs are all kinds of streaming applications for non-live content, such as Video on Demand (VoD). A prominent example is YouTube.

Some cases can be seen as combinations of FTs and STs. An example for such a case is a stream that has large chunks of data, which are only usable as a whole. In this case, the playout curve would not be a straight line, but instead a function with steps. Or another combination: a FT that already has some value for the user, even if though is not yet fully received. A well-known example for such a case is a web page that is already usable when images, Cascading Style Sheets (CSS) style files, and scripts are missing, and only the Hypertext Markup Language (HTML) text has been received. QoT does not define a transaction type for these cases; instead, the application can decide whether the corresponding case is better described as a ST, or as multiple FTs, or as a combination of both.

2.5 Utility of Transactions

The formulation of the users' requirements as utility functions is the second key concept of QoT. The outputs of these functions are utility values that reflect the value of a transaction for the user. In general, utility in the context of communication networks and scheduling defines the valuation of one party for a hypothetical outcome in a network. Often, utility is formulated in relation to an instantaneous or mean data-rate, for example in [MW00]. Sometimes, utility is also assigned to single packets, [Kel01]. The QoT principle is to regard utilities as solely relating to transactions:

Definition 2.3: ***Utility***
For QoT, utility means the value a transaction has for the user.

This definition is similar to the concept of QoE, as its focus is on the user's experience of a network transmission. There are several different definitions for the term QoE, which are listed in [LC12]. The unique point of QoT is, however, the focus on transactions.

The requirements of QoT are not just scalar utility values specifying the value of this transaction in optimal network condition, but instead functions that specify the value depending on the scheduling.

The user evaluates the quality of services, results, or events based on parameters that he observes and notices. The inputs for these utility functions are metrics that can be measured from the data transmissions or streams. Each type of transaction requires its own type metrics and utility functions.

2.5.1 Network View on Transactions

One reason why it is difficult to map network metrics to application layer metrics and vice versa is because they are defined on different granularity levels, see Section 1.2.4.

When the traffic granularity of the QoS system is finer than the granularity of the user's perception, the perception depends on the combination of many QoS parameters. Each single parameter only has a small influence.

An example for this combination would be, downloading a large file. The user perception depends on the time the download is finished, which corresponds to the time when the last packet has arrived. Assuming the application will already specify the deadline of all packets to end at the same point in time, then for the user there is no difference whether one packet is late, or 100 packets are late. A scheduler, however, will try to fulfill the requirement of each packet, it will therefore count 100 times as bad when 100 packets are delayed. Further, when a single packet cannot be transmitted for any reason, the effort for all other packets of this download is wasted.

When the traffic granularity of the QoS system is coarser than the granularity of the user's perception, it is obviously impossible to differentiate among the multiple applications or services.

An important design principle for QoT is, therefore, to make the traffic granularity of the service differentiation attribution as close as possible to the granularity of the user's perception.

However, in many cases, for one transaction there is no direct corresponding network level entity. None of the typical granularities listed in Section 1.2.4 corresponds to the transactions from a user's view. In some cases, a transaction may correspond to a transport layer connection, but there are also many cases where a transaction consists of more than one transport layer connection, or where a transport layer connection belongs to multiple transactions.

Since QoT requires the headend to be able to recognize transactions, the QoT architecture, therefore, needs mechanisms to identify transactions, which is explained in Section 3.2.1.

2.5.2 Application Layer Metrics

Traditional QoS approaches regard low-level network parameters, such as bitrate, latency, packet loss, and jitter. The user's experience, however, depends on metrics on the application layer, such as the time until a result is displayed, or the number and duration of times a streaming video stalls. Mapping of the network metrics to application layer metrics is difficult; examples of studies on this topic are [Int05, NUN10].

Relevant application layer metrics are:

- The time until the user receives a result. Such a result may be
  - A full display of a web page
  - A finished download
  - A game level has loaded and starts
  - An email is displayed
- The audio quality of a VoIP call. The audio quality is usually assessed by the subjective MOS [Rec96], which is, by definition, to be measured by survey. Several algorithms or models exist to estimate the MOS value from technical parameters, for example, the e-model [ITU11]. With a given audio hardware, audio codec, and delay chosen by the application, the only influence of the access network is how many packets are lost or do not meet the latency requirement.
- The audio or video quality of other real-time transmissions, such as video telephony, live TV broadcast, etc. Again, the only relevant variable for an access network is the number of packets that are lost or do not meet the latency requirement.
- Often, voice or video is transmitted with lower latency requirements (several seconds and more), and with automatic repetition of lost packets. A prominent example for such an application is YouTube. In case the buffer runs empty, the user has to wait until new data is available, which is colloquially called "rebuffering". The user experience depends on the number and duration of rebuffering procedures.

2.5.3 Finish-time Transaction (FT)

Since the network only influences the time when the result appears, the user's experience, or the user's valuation for this transaction, can only depend on this finish-time. That means, the value of a FT is a function of the finish-time, as sketched as an example in Figure 2.7. This figure shows the basic form of the user's valuation for two different transactions, depending on the time when these transactions are finished. The user's valuation is a vague metric, which will be defined in Section 2.5 as utility values. It is obvious that these functions are monotonically decreasing, because the value for the user is higher, the earlier the transaction is finished. The blue curve is an example for the user's valuation of a web page. The user expects to wait about a second, but if the web page loads even faster, the user may not even notice the difference. After some seconds, the user becomes increasingly impatient, and finally closes the browser tab, or opens another web page. At that time, the user's valuation cannot become worse, which results in a horizontal shape. The green curve is an example for a download. Even after waiting some time, the user will not cancel the download, because it is running in a download manager and the user can continue with other work. However, the user will still become increasingly unsatisfied with the service the longer it takes.

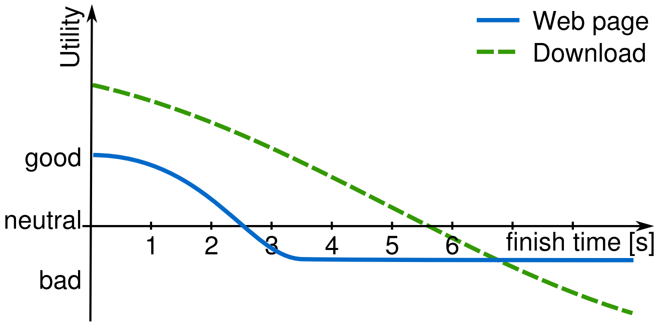


Figure 2.7: Example of a user value judgment for two different Finish-time Transactions (FTs).

2.5.4 Real-time Transaction (RT)

The user's experience, or the user's valuation for a RT, depends on how many packets fulfilled the requirement, and how many packets did not. In some cases, for example for some VoIP codecs, the user's valuation may also depend on the burstiness of lost packets (see [ITU11]). In some cases, some of the packets of the transaction might contribute more value for the user than other packets. This is true, for example, for i-frames and p-frames in some video codecs.

2.5.5 Streaming Transaction (ST)

The user's experience, therefore, depends on how often the buffer runs empty and how long it takes to recover.

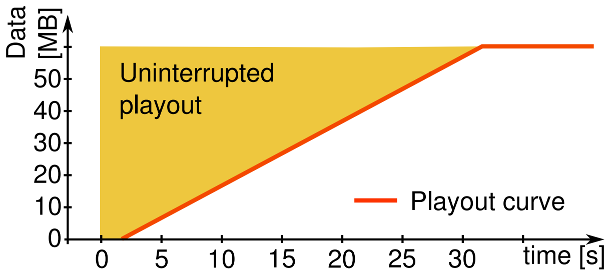


Figure 2.8: Example of the playout curve of a RT.

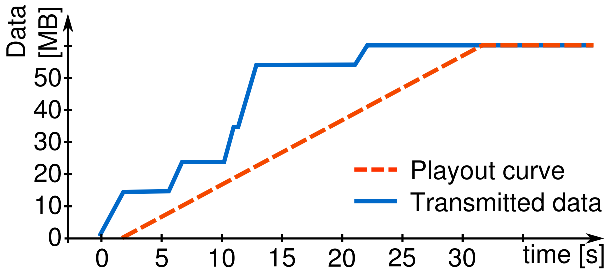


Figure 2.9: Example of the playout curve of a RT with an example of the progression of transmitted data of perfect service quality.

Figure 2.8 shows an example of such a ST. The transaction in this example could be a Constant Bit Rate (CBR) video which will start to play at $t = 2s$, with a duration of $30s$ and a bitrate of 32 Mbit/s. The playout curve (red line) shows the amount of data that needs to be received at each given point in time. As long as the currently received offset in the transmission is inside the colored area, there are no interruptions. When there are packets that are delayed long enough to cause the received offset to be to the right of the playout curve, there is an interruption, and the transaction is impaired.

The blue curve in Figure 2.9 is an example of the amount of data received for a perfect service quality, because the blue curve is always above the red line. In Figure 2.10, the blue curve shows an example of a transmission where starting from $t = 10s$ the buffer runs empty and the

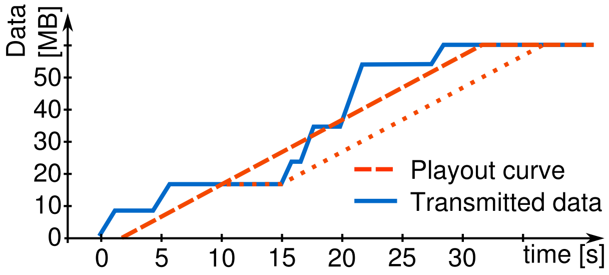


Figure 2.10: Example for the playout curve of a RT with an example of the progression of transmitted data of bad service quality.

video stops playing. From $t = 15s$ on, the video can resume playing, and the red dotted line shows the resulting playout curve after the video resumes playing.

2.6 Summary and Conclusion

The assumptions for designing a new QoS schema for Internet subscribers are:

- The access will continue to be the bottleneck for Internet subscribers
- Access media shared among multiple subscribers will become more widespread.
- DTEs will continue as intelligent devices that know which transmissions have significance for the user.

QoT has two key concepts: transactions and utility functions.

Transaction represents a result from network transmission from the user's view, for example, a website that is displayed. Transactions are at the same time defined, from a machine's view, as a set of sections from transport layer connections or other transmitted data streams. The user's service requirements are formulated and signaled to the headend, referring to the transactions; while the attribution in existing QoS approaches typically refers to packets or transport layer connections.

A second key concept is utility functions. Utility functions are a way to formulate the service requirements for transactions as a time-dependent function. The end users' applications define the requirements for each transaction, depending on the type of result as a function. The values of this function specify the value of the transaction for the user. It specifies this value depending of the on the scheduling of this transaction. This allows the scheduler to optimize the transmissions based on the current load and network state.

3 Architecture for Quality of Transactions

The QoT concept, as explained in Chapter 2, can be used in many different types of communication networks. In this chapter, an architecture for QoT for IP networks is developed. The architecture described in this chapter therefore considers only IP access networks, with the structure of current Internet access networks in mind.

Which information is relevant for QoT is detailed in Section 3.2. The information is defined and signaled on the granularity of transactions, which are introduced in Section 2.4. This includes the signalization protocol to signal the transaction information to the headend in Section 3.2.4.

The key component of QoT is the access network headend. The role of this component as well as some implementation considerations are given in Section 3.3, this includes implementation aspects and implementation complexity for the access concentrator. The scheduler in the headend creates the scheduling decisions. The problem formulation for such a scheduler is explained in Section 3.4.

The sources of transaction information in the end system are given in Section 3.5. This information is transferred to the headend using the above mentioned protocol. Section 3.6 gives an overview on possible topologies and deployment scenarios, where this approach can be used.

3.1 Basic Operation and Components

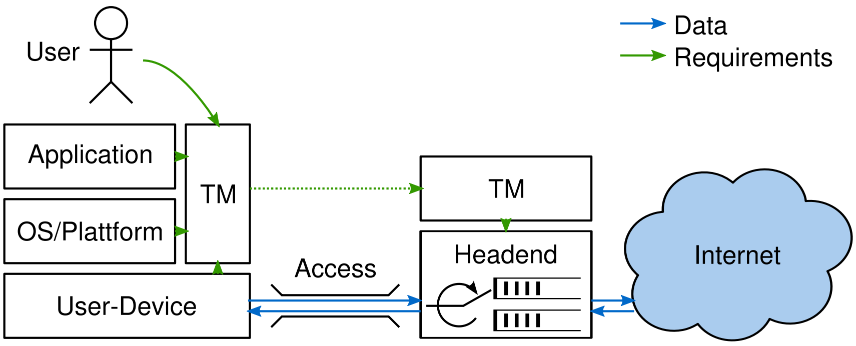


Figure 3.1: QoT system overview.

QoT tries to improve the user's service experience, by dividing the user's traffic into transactions (Section 2.4), by formulating requirements for these transactions, and by using knowledge about these to improve scheduling in the access network.

QoT requires several components. Figure 3.1 shows an overview of an exemplary QoT system. All required components are located either on the access network headend or on one of the customer's devices.

- The user, applications, OS, platform, and the device's sensors provide the information about the transactions' requirements.
- A Transaction Manager (TM) component collects all relevant information from the user, application, platform, operating system, and the device. The sources of information are listed in detail in Section 3.5. The collected information describes the requirements for all transactions.
- The TM transmits these requirements to a TM on the other side of the bottleneck. The corresponding node is typically a node of the access network operator, which is close to the headend. In a simple setups the TM on the operator's side is deployed on the headend.
- The scheduler on the headend uses the information from the TM to improve the service quality by scheduling.

3.1.1 Transaction Manager

Definition 3.1: ***Transaction Manager***
A TM is a physical or logical component that collects, combines, processes, and forwards information about transactions and their requirements.

For QoT the scheduler requires a component on the headend, to provide the necessary transaction information for scheduling. This is provided by a TM instance on the headend. The TM in the access network headend collects the signaled information and provides this information to the scheduler. It has to provide an interface to the scheduler, so that the scheduler can obtain all necessary information about each transmission or flow. It may be necessary to estimate missing information.

In addition, a TM may be deployed on a DTE or on another device on the subscriber's side of the access network, for example on the residential gateway.

A TM on the DTE has the following functionality:

- It intercepts the transaction requirements sent from applications, combines them, processes them, and forwards them to the headend.
- It may offer an Application Programming Interface (API) to applications, to help applications to create and signal their transaction information. This API should be integrated

with the API for opening network sockets and it should provide default values, so that the application only needs to change the values that are special for this particular connection. An example of such an API is given in Section 3.5.7.

- It makes use of as many of the information sources as possible (listed in sections 3.5.3, 3.5.4, and 3.5.5).
- It uses these information sources to provide sensible default transaction information for applications that do not support QoT.
- It may provide user configuration as specified in Section 3.5.1.

The OS, the platform, or a third-party application may provide such a TM.

3.1.2 Headend

The TM is only one part of the headend. Per definition 2.1, the headend is to be the node that controls resource allocation on the access network.

The Headend needs to be able to process network and transport layer information (especially IP addresses and TCP ports), to be able to recognize transactions. The headend also needs to be responsible for the resource allocation on the access medium. There are cases where the two functionalities (IP packet processing and scheduler) are not implemented on the same device. One prominent example of such a case is LTE, where multiple devices in the LTE System Architecture Evolution (SAE) backhaul together form a virtual headend. Details on the deployment in those special cases are given in Section 3.6.

The schedulers that are relevant to do the prioritization on the bottleneck (shared access medium) are on the headend and on the DTE or CPE.

In the downlink, QoT manages to bring the information about the requirements of the different connections to the headend. The headend identifies the corresponding data packets as they arrive from the Internet and uses the information about the requirements to fulfill these requirements and improve the user's valuation for the services.

In the uplink, the DTE or CPE uses the information about the requirements locally, to decide which data to send first. However, in case of an access network, where the headend also arbitrates the resource allocation of different users on the access medium, the signaled information about the requirements is also used to do this arbitration.

In the case that the subscriber has multiple DTE sharing his access line, the residential gateway has to do the prioritization in the uplink, and it has to combine the signaled requirements from all his DTEs. Such a combination of multiple sources is also necessary when the access network consists of multiple hierarchies of networks.

3.2 Information Flow and Signaling

One important concept of QoT is that the only additional signaling is from the customer to the headend.

All information that influences the user's requirements or that influences how the user will experience the service, are available on the user's side. Information sources may be the user himself, sensors of his device, the OS, the software platform, or the application. This information is needed in the schedulers on both sides of the bottleneck. These schedulers are in the DTE or residential gateways and in the headend. Therefore it is required to signal information for QoT from the DTE to the headend.

To inform the access network about the users' requirements, two things need to be transmitted:

1. A description of the transaction, especially how to identify data belonging to this transaction
2. A description of the user's requirements for this transaction.

Section 3.2.1 explains how transactions are formulated so that the headend can identify packets belonging to the transaction. Section 3.2.2 explains how the requirements are formulated using utility values and utility functions, so that the headend can optimize the sum utility to provide the best experience for the user. The metric or unit of these utility values is explained in Section 3.2.3.

3.2.1 Transaction Identification

The headend needs to be able to identify all data or all packets that belong to each transaction. The applications on the DTE define which connections or parts of connections belong to a transaction. The DTE sends packets in the uplink direction, it could therefore annotate these packets, so that the headend can identify them. But it is not possible to annotate packets in the downlink direction, without changing the protocols between the DTE and the server. Instead, the DTE needs to describe the transaction, so that the headend can identify all packets that belong to this transaction.

The Definition 2.2 of transactions is based on results for the user and not on transport layer connections. While it is common, that one ST or one RT matches exactly one transport layer connection, one FT will often consist of multiple transport layer connections or only parts of a transport layer connection, as detailed in Section 2.4.1.

Identification of a transport layer connection can be done using the IP 5-tuple (source address, destination address, IP protocol number, source port, destination port). There are only a small number of transport layer protocols that are widely used in the Internet today, which are TCP and UDP [Ros08]. The complexity to implement transport layer specific matching algorithms is therefore low. Matching the port numbers is specific to each transport layer protocol.

To be able to specify and recognize parts of a transport layer connection however is more complicated. Typically these parts are protocol elements of a higher layer, for example a HTTP element. There are however major disadvantages of such an approach:

- The number of protocols on higher layers than the transport layer is high[1], which would cause a high implementation complexity.
- Some of the application protocols are very complex to implement.
- Compared to transport layer protocols, application protocols are more often modified. Additionally, new application protocols are introduced often. Therefor larger maintenance effort is required.

It is therefore desirable, to define parts of transport layer connections independent of these higher layers, but instead only use terms of the transport layer protocols. As a compromise between flexibility and complexity, the data length inside a transport layer connection has been chosen. This decision is rather TCP centric, because in TCP the transmitted data forms one strictly sequential stream of data, and maintains a sequence number to refer to parts of this stream. However, the requirement that a transaction contains only parts of a transport layer connection, is mainly relevant for TCP connections. How these features can be formulated in a protocol from applications or from a TM on the subscribers side to the TM in the headend is explained in Section 3.2.4.

In the following throughout this work, these sections or parts of transport layer connections are called “chunk”. Each chunk is a part of a transport layer connection. A chunk is either in uplink or in downlink direction.

Definition 3.2: ***Chunk***
A continuous part of the data stream in one direction of a transport layer connection, identified by a start offset and a stop offset relative to the start of the connection.

3.2.2 Transaction Requirements

The requirement of a transaction is derived from the definition of the three transaction types in Section 2.4. All requirements are formulated as utility values depending on the metrics that are relevant for the corresponding transaction type. That means, the signaled requirement is not a scalar, but a function as defined in equation 3.2.

[1] For example the file `/etc/services` which contains an officially maintained list of protocols on layers above transport layer, contains 1678 services using TCP on Ubuntu 12.04

- For a FT the requirement specifies when the transaction should be finished. The utility value therefore depends solely on the finish-time f_T of this transaction T.
- For RT the requirement specifies acceptable packet loss rate and latency deadlines per packet. The utility value therefore depends on the relative frequency of packets that do not meet the latency requirement and are therefore lost. In some cases, the utility value may be different, depending on whether there are long sequences of lost packets or whether the losses occur randomly over time (Bernoulli process). A metric describing the burstiness of the loss events, the autocorrelation of packet losses of transaction T (Ψ_{lT}) is used to differentiate these cases. Multiple definitions for the autocorrelation of loss events can be found in [NR08, Formula 2]. Another metric is the probability, whether there are two loss events within a time duration d from any point in time t:

$$\Psi_{lT}(d) = \mathbb{E}\left(\min\left(1, \sum_{\tau=t}^{t+d} I_{lT}(t) \cdot I_{lT}(\tau)\right)\right) \tag{3.1}$$

  With $I_{lT}(t)$ as the loss indication of a packet at time t of transaction T. $I = 1$ in case of a lost packet, $I = 0$ otherwise.
- For ST the requirements specify the playout curve. The utility depends on the number of "rebuffering" events of transaction T (B_T) and the accumulated total "rebuffering" time of transaction T (b_T). This time is the time deviation at the end of the transaction between the actual playout and the planned playout curve. For a streaming videos, B_T is the number of times the video was "rebuffering" and b_T is the total time the video was paused due to this "rebuffering".

$$U_T = \begin{cases} U_T(f_t) & T \in \text{FT} \\ U_T(l_T, \Psi_{lT}) & T \in \text{RT} \\ U_T(b_T, B_T) & T \in \text{ST} \end{cases} \tag{3.2}$$

For the QoT principle it is irrelevant how these utility functions are encoded and signaled. For simplicity, it can be assumed that the utility function would be signaled as a lookup table. Section 3.2.4 shows how to encode utility functions more efficiently, for example by using a parametric curve and signaling just the parameters.

3.2.3 Unit of Utility

Utility values are to be compared. It is not sufficient to compare utility values of single transactions, instead an average utility value or the accumulated utility over a certain time period should be compared. Therefore it has to be possible to determine a mean of utility values. Therefore utility values have to be defined on a linear interval scale (definition of scales according to

[Ste46]). The main implication of this constraint is, that a small difference between two utility values reflecting a bad user experience have to be comparable to a small difference between two utility values reflecting a good user experience. This constraint excludes ordinal scale metrics such as the level of satisfaction where differences are not quantifiable and therefore a mean value has no meaning.

Utility values are further not bounded. An arbitrarily large transaction might produce an arbitrarily large utility. There is also no lower limit for the dissatisfaction. For example, a network with insufficient resources might not be able to fully transmit a certain transaction, this transaction is therefore dropped at some point in time and the resulting utility value will be low. Worse than dropping one transaction is dropping two transactions, the resulting utility value has to be even lower.

It is not mandatory to require a ratio scale. There is no reason for multiplying or dividing multiple utility values with each other. There is also no intuitive, inherent zero-point for utility values. When using a metric on rational numbers, the zero-point can be chosen arbitrarily, however a sensible and easily to understandable point should be chosen.

Independently of the choice of the zero-point, all utility values can be scaled with a constant without affecting their properties. A Scaling factor has to be equal for all utility values, while the choice of the zero-point could be different for each transaction.

MOS values cannot serve as utility values here, as they are bounded (1 to 5) and they are not linear over larger ranges. For example, a service that drops every second transaction (MOS 1) and has a perfect quality (MOS 5) for every other transaction, is still unacceptable for most users.

Monetary units are metrics on rational numbers, that are easy to understand and easy to match to everyday things. As the scaling, the unit and the zero-point of utility values are arbitrary, the following definition is used:

- Utility values are unit-less rational numbers
- Utility values are scaled to match monetary units
- The zero-point is used for a service, that a user would not pay for

Any other zero point would be equally possible, as this would only cause an offset on the sum utility. The choice of the zero point is therefore solely to allow a more intuitive interpretation of results.

3.2.4 Signalization Protocol

As mentioned above, the users' requirements need to be transported from the user to the devices on both sides of the bottleneck, especially to the access network headend. A new signalization protocol is necessary for this task. This QoT signalization protocol has to provide the following capabilities:

- Identification of the transactions as explained in Section 3.2.1 from a network perspective, e.g. in network and transport layer items.
- Formulation of the requirements as utility functions as explained in Section 3.2.2.
- It has to adopt to different deployment scenarios, for example there might be an arbitrary number of routers between the user's device and the CPE.

This information is gathered to signalization packets. A specification, how to formulate such a signalization packet is given in Appendix D.

These QoT signalization packets have to be sent towards the corresponding node, because data to the corresponding node will follow this path, and data from this corresponding node will follow the same path in the opposite direction. The signalization packets could therefore be sent to the IP address of the next IP hop towards the corresponding node. This node combines the signalization from all connected nodes and forwards them again and optionally uses these information for its scheduling decisions.

This approach will only be possible, if all IP nodes in between the user and the access network headend support this QoT signalization protocol. To allow for intermediate nodes, that are unaware of QoT, an IP anycast address [PMM93, AL06] is used, as the destination of QoT signalization packets. Any node that is unaware of QoT will forward these packets to its default gateway. Any node that is aware of QoT accepts and processes these packets.

3.3 Access Network Headend

The most important component for QoT is the access network headend. The headend performs the scheduling for the bottleneck in downlink direction and is therefore the main control vector in QoT. As defined in Definition 2.1, the headend has full control over the bottleneck. When the headend sends data or packets, these are directly transmitted over the access medium of the bottleneck. If there was a buffer between the headend and the medium, there would be an additional delay and the headend would not be able to control the timing of packets so that the requirements are met.

The implementation on the headend needs to perform the following tasks:

- It has to receive and collect the transaction information from the subscriber, using a TM.
- It has to evaluate the transaction identification as specified in Section 3.2.1 and match each packet received from the Internet, to which of the transaction it belongs.
- It has to buffer the packets for each transaction separately.

- It has to use a sensible default for packets or connections that are not attributed with QoT information.
- It has to perform its scheduling in a way that tries to maximize the sum of all utility values. This task is explained in the following Section 3.4.
- It has to manage fairness between users, otherwise one user can gain an unfair advantage by falsifying his transaction requirements to be much more demanding. There are different ways to manage the fairness. One possible approach is shown in Section 3.3.2.

3.3.1 Implementation Considerations

Depending on the parameters of the access network, it may be beneficial to manage the buffered packets intelligently for transport layer connections to avoid interfering with control loops in the corresponding transport layer protocols. A strategy that is well known for TCP is to terminate the TCP connection in both directions and to buffer the data stream in this TCP connection instead of buffering the packets. This approach is commonly referred to as TCP proxy. There are several commercial products that implement such a TCP proxy, to optimize network usage, these products are usually called “Network accelerator”. An example for such a product is given in [LST$^+$10].

Buffering data per transaction is comparable to per-flow packet buffering, which is already common practice in many access network technologies.

3.3.2 Fairness

In congested situations, subscribers may be tempted to cheat, by signaling arbitrarily demanding transaction requirements. For example, a user might signal a utility function for each transaction that has a huge drop in utility value shortly after the beginning of the transaction. This might lead the access network headend to prioritize the transactions of this user.

There are several possible solutions. For example, the headend implementation can equalize the utility curves, by multiplying them with a factor, so that in average all subscribers have the same utility ranges.

Or the headend may simply count, how many times a subscriber displaces transactions of other subscribers in the scheduler. If this counter raises over a given threshold, the utility curves for this customer are dimmed down so that other subscribers have better chances to be scheduled.

3.4 Access Network Scheduler

The scheduler is a component of the headend. The scheduler has to maximize the sum of the users’ utilities, which is an optimization problem.

3.4.1 Visualization of Scheduling Problem

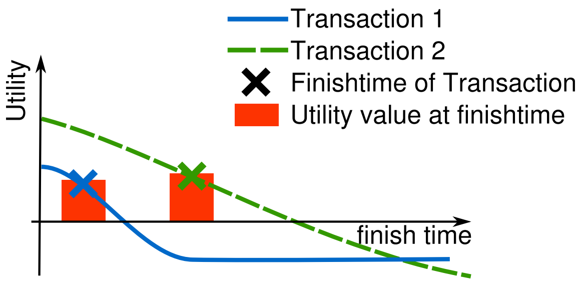


Figure 3.2: Example of the utility curve of two FTs.

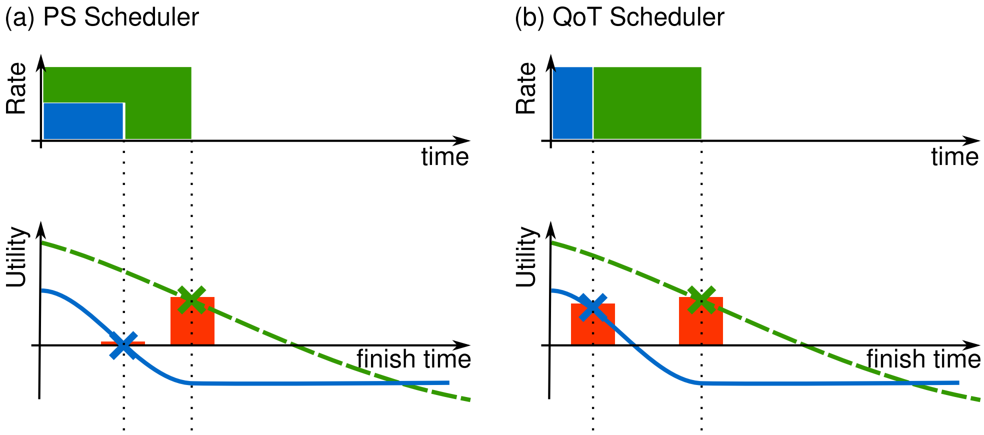


Figure 3.3: Example for the resulting utility for two different schedulers.

The optimization task for two FTs is visualized in Figure 3.2 and 3.3. Figure 3.2 shows an example with two different transactions. Transaction 1 is a web page, transaction 2 a download. The point in time when the transaction is finished is marked with a cross. The resulting utility is marked with a red bar. In this example both transactions are finished timely with a reasonable high utility.

The task of the scheduler is to maximize the sum of the height of the red bars. Figure 3.3 shows how two different schedulers would allocate the resources. A Processor Sharing (PS) scheduler would assign both transactions half of the available resources. Transaction 1 would be finished earlier than Transaction 2, because it is smaller. When Transaction 1 has finished, all resources would be assigned to Transaction 2.

A QoT scheduler could give all resources to Transaction 1, until it is finished. The finish-time of Transaction 1 is therefore only one half of the finish-time with PS scheduler. The finish-time of Transaction 2 would be the same for both schedulers, because the total data-rate is the same in both cases. As a result, the sum of the utilities (red bars) is lower for the PS scheduler.

Figure 3.4 shows two additional hypothetical schedulers that perform even worse than the PS scheduler: Scheduler 3 assigns resources proportional to the transaction size, which effectively

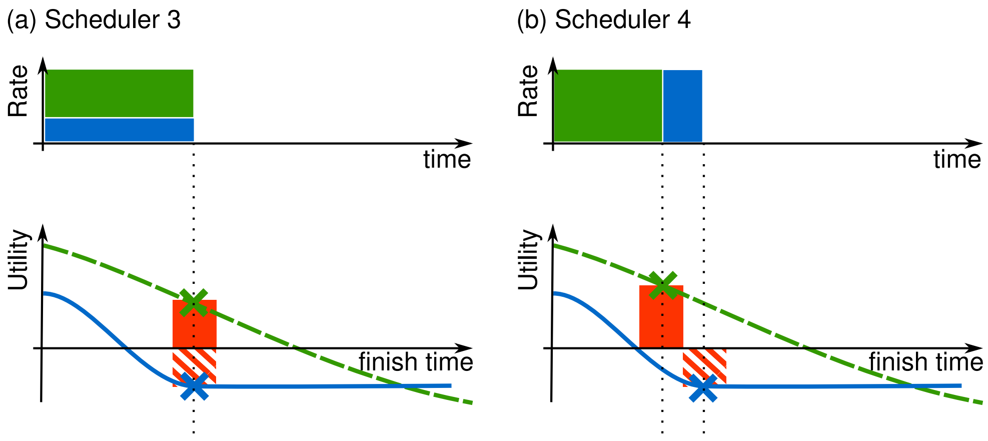


Figure 3.4: Example for the resulting utility for two different schedulers.

maximizes the finish-times and minimizes the sum utility. Scheduler 4 prefers Transaction 2 over Transaction 1, however Transaction 1 suffers more from the additional delay than Transaction 2 benefits from its earlier finish-time.

3.4.2 Optimization Problem

Under the following assumption a linear optimization problem can be formulated:

- utility functions are arbitrary, monotonically decreasing functions
- there are only FTs
- all utility functions are known to the scheduler at start time t_0
- the size of all transactions is known to the scheduler at start time t_0

For radio access channels with fading channels, this optimization problem has been published in [PKV11, Section 5.B]. For fixed access with a constant data-rate per customer, an analogous linear formulation can be found.

The objective function is to maximize the sum of the finish-time utility values:

$$U_{total} = \sum_t \sum_T U_T(t_f) f_{T,t} \tag{3.3}$$

The term U_T is the utility function of the finish-time transaction T and t_f is the finish-time of transaction T. The time is discretized into arbitrarily sized time intervals (corresponding scheduling intervals) with duration Δt. $t \in \mathbb{N}$ denotes the index of the time interval.

To linearize the maximum-operator, a flag $f_{T,t} \in \{0,1\}$ is introduced which is one for the finish time $t = t_f$ of transaction T and zero for all other time intervals t.

The resource allocation is defined as $r_{T,t} \in [0,1]$, which specifies the share of the scheduling interval t, that is assigned to transaction T. This resource allocation are therefore the decision variables of the linear optimization problem.

The optimization problem has the following constraints:

$$\begin{aligned}
\forall T : \sum_t f_{T,t} &= 1 && (3.4)\\
\forall T,t : f_{T,t} &\leq \frac{1}{s_T}\left(\sum_{\tau=0}^{t} r_{T,\tau}\frac{R}{\Delta t}\right) && (3.5)\\
\forall t : 1 &\geq \sum_T r_{T,t} && (3.6)\\
\forall T : s_T &= \sum_t r_{T,t}\frac{R}{\Delta t} && (3.7)\\
\forall t < t_{0T} : r_{T,t} &= 0 && (3.8)
\end{aligned}$$

Constraint 3.4 enforces that there is exactly one finish flag for each transaction. The finish flag is defined by 3.5 stating that a transaction can only be finished after all of its data has been transmitted. In contrast to the formulation in [PKV11], instead of the channel quality, here the available data-rate of the access medium (in bits per second) (R) is used. Constraint 3.6 makes sure that resources are used only once. With 3.7, it is ensured that all transactions have to be finished. With constraint 3.8 it is ensured, that transactions cannot be transmitted, before their start time $t_{0,T}$.

This linear program can be formulated for the uplink direction, for the downlink direction, and for a scheduler that performs scheduling for uplink and downlink.

Such an optimization problem is however not feasible for a scheduler, because of its complexity and because the preconditions are not realistic.

- The number of decision variables $r_{T,t}$ is growing with the number of transactions $|T|$ and the number of time intervals $|t|$. It grows with $\mathcal{O}(|T| \cdot |t|)$. For realistic traffic the required time period results in a large $|t|$.
- The optimization problem can only be solved, when the size and the utility functions are already known in advance, which is not realistic.
- The arrival of future transactions cannot be neglected.

The optimum scheduling has to fulfill the following condition: All transactions form a sequence, and transaction n of that sequence is not assigned any resources until transaction $n-1$ is finished. If there were resources assigned to transaction n while transaction $n-1$ is not finished, the result can be improved by assigning these resources to $n-1$ instead. By doing so, transaction $n-1$ finished earlier, while the finish-time for transaction n remains unchanged.

This condition for an optimal solution already simplifies the problem significantly, because it implies, that the complexity of finding an optimal solution is only depending on the number of transactions.

However the problem can still not be solved in polynomial time. The complexity is growing exponentially with the number of transactions. Because it is not possible to formulate finite bounds for any variation of a given sequence, a small change in the resource assignment for one transaction can lead to an arbitrarily large change in the resulting utility, because the shape of the utility functions allows arbitrarily steep slopes.

Limiting the slope of the utility functions to realistic values does not seem to be a practical solution, because rough estimations suggest that the exponent of a polynomial algorithm will be in the hundreds or thousands, which is the same order of magnitude than the assumed number of transactions. The difference between the biggest and smallest slope has to span several decades to account for the different traffic requirements. A change at one transaction can therefore in the worst case affect the order of several decades of transactions.

Heuristics are therefore needed to perform the scheduling. The heuristics that are used in the simulative performance evaluation are explained in Section 4.3. In addition to the FTs in the example of Figure 3.2 and Figure 3.3, a QoT scheduler also needs to consider STs and RTs.

3.5 Sources of Information

The user is the only relevant source of information about the requirements for his transactions, because the goal of QoT is to improve the users' experience. However, it is neither possible nor feasible for the user to specify these requirements. It is impossible, because a user will not know in advance, how he would rate his experience given a hypothetical outcome of a transaction. He might not even be able to explicitly formulate his experience afterwards. The effort that would be needed is large compared to the value of the transaction.

The amount of information about the transactions requirements, that a user can provide directly, is limited. However the needed information can be obtained or derived by the user's device, the OS or the applications.

Definition 3.3: ***Context***
Context is all information that can be used by one of the involved components to derive information about the transactions and the users' requirements for these transactions.

In general, there is information usually regarded as context, which is not available in a usable form. For example, the user's mood is usually not recognized by its computer. Such information is not regarded as relevant context for QoT.

The following sections show which context features can be used and in which of the components these context features can be obtained. Those regarded components are:

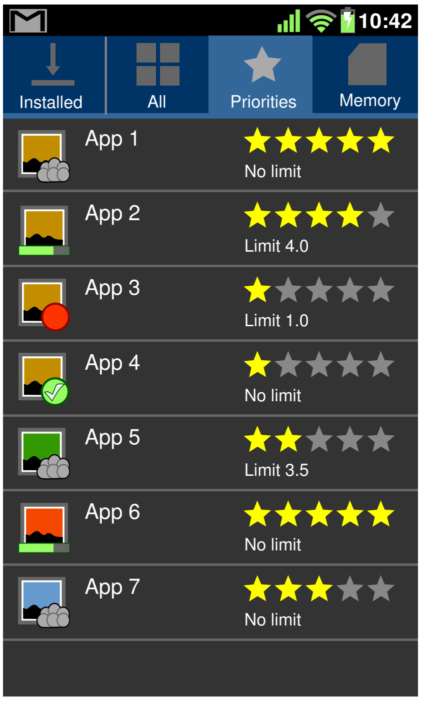
- User configuration

- Applications
- Platform
- Operating System (OS)
- Sensors of the user's device or Data Terminal Equipment (DTE)
- Residential gateways

3.5.1 User Configuration

As mentioned before, it is not possible to ask the user about each requirement. There are only few ways how the user can interact directly and provide additional information.

A configuration dialog can offer the user to adjust parameters that will influence the signaled requirements. This might be relevant, for example to adjust weights between multiple applications. For example, the user could be presented a list of those applications that generate most of his traffic and the user can organize them into three groups: important, normal, and unimportant. An example of how such a configuration dialog could look like is sketched in Figure 3.5. The user can configure his preference for certain applications by setting a weight (one to five stars) and a maximum priority, which limits the utility function to applications with that weight.



App 1 ★★★★★ No limit

App 2 ★★★★☆ Limit 4.0

App 3 ★☆☆☆☆ Limit 1.0

App 4 ★☆☆☆☆ No limit

App 5 ★★☆☆☆ Limit 3.5

App 6 ★★★★★ No limit

App 7 ★★★☆☆ No limit

Figure 3.5: Example of configuring priorities of applications in a smartphone.

A download manager may offer a "turbo" button that a user can press to inform the download manager, that he needs this single download as soon as possible. When this button was pressed, the download manager will signal demanding requirements for the corresponding download, while for other downloads it will signal relaxed requirements. An example of how this could be implemented in the download manager of a browser is shown in Figure 3.6.

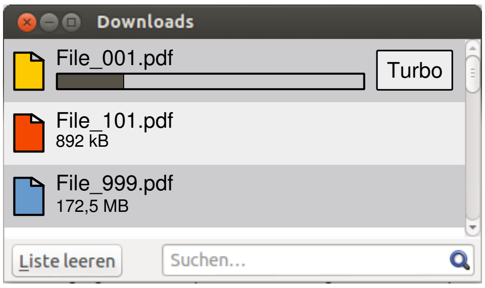


Figure 3.6: Example of a download manager with “turbo“ button.

3.5.2 Application

Besides the users, the applications are the second best source of information. The user interacts with the applications directly, the applications react on the user’s input, and the applications define what the user sees on the screen. In many cases, the application developer already has to think about how long something should take and how long it will take. For example, it is long known [Mil68, CRM91, Nie94] that a step in an application that takes significantly longer than one second will interrupt the user’s flow of thought. If the application developer thinks that this is the case, he is advised to make the following steps not dependent on the results of this step.

The application developer often knows why data is being transmitted by the application, because he programs the code to do the transmission on purpose. This context of each data transmission is available within the application:

- The application should know, whether the transaction is required for an interaction, whether the transactions is one step in a flow of thought of the user, or whether it is the result of a task where the user expects to wait. Depending on that, the application already should take care to make the application usable [Nie94].
- The application knows the source of an event. Events can be triggered by the user (mouse-click, touch gestures, keyboard presses), by a timer, by data from another transmission or by the OS. Usually, the events that are triggered by the user cause transactions that the user expects to finish soon, while events that were triggered by timers or other background events cause transactions that have relaxed requirements.
- The application also knows what it will do, once a transaction is finished. When the result is just written to disk, it might be less urgent than if the result will be directly displayed to the user. When the Graphical User Interface (GUI) is programmed to block until the transaction is finished, it is in general even more urgent.

Further, there is relevant context outside the application that the application can take into consideration to determine the requirements for a transaction. Such information can be obtained and provided by the application, but it could also be provided by the OS or platform for all applications.

- Type of application: Often the type of application already gives a good guess about its requirements. For example game, backup software, web browser, etc.
- Foreground/Background: When an application is currently not being displayed in the foreground, many of its transactions will have relaxed requirements.
- A proximity sensor that detects whether the device is held against the ear can be used to prioritize applications with audio output, while applications with visual output become less important.

More complicated are generic applications, where the application forms a flexible framework to perform tasks, which are specified as loadable scripts. The behavior of such an application is to some degree determined by these scripts. For some of the functionality of the application, the application developer therefore does not know why this functionality is used and how its result will interact with the user. A highly complex example is a web browser running an application written in JavaScript. Many cloud applications are fully implemented in JavaScript and run in a web browser. It is unrealistic to expect the JavaScript developer to formulate the requirements. Therefore a web-browser needs to implement heuristics that analyze how each data transmission will be used in the user interface.

Currently we see a trend, that for many applications which were previously run in a web browser, small dedicated smartphones applications are created and distributed. These applications often still use JavaScript to formulate the behavior and a Hypertext Markup Language version 5 (HTML5) widget to display. In most cases, such applications could easily formulate the requirements for their data transmissions.

3.5.3 Platform

Besides HTML5 we see a similar trend in other fields: the platforms increase their functionalities to higher level functions, which previously applications had to provide on their own. Examples are the download manager and the voicemail API in Android API level 14 (Android 4.0) and many more listed in [Goo11]. These trends will make it easier for application developers to support QoS schemes such as QoT, because most of the knowledge about the transactions is also available in the platform and the platform can provide a sensible default behavior.

In this context, a platform is a comprehensive abstraction layer for application development.

The most popular platforms today are current versions of Microsoft Windows, Android and

Apple iPhone OS (iOS). Especially for mobile devices, in the last years an extreme consolidation of platforms for smartphones has taken place. In the second quarter of 2012, 85% of the sold smartphones use one of the platforms Android or iOS [IDC12].

Typically a platform offers high level building blocks for applications with a unified and consistent behavior and a high degree of compatibility and interoperability with other applications and services. Platforms go beyond classical application libraries, as their library functions are highly integrated with services and management applications provided by the platform. These platforms usually also integrate abstractions for many functionalities, that were previously considered OS functionalities. Prominent examples are file and storage management and networking functions.

Those platforms have access to some of the context sources of applications listed above:

- Application type
- Foreground/background
- Event source

Beyond that, a platform can keep record of an applications history, e.g. how the result of a data transmission has been used.

With these context features, the platform can estimate the requirements of each data transmission. With such an estimation signaled to the headend, for many applications there is no need to formulate and signal the requirements. The networking API of the platform should be structured in a way, that for applications that do not support QoT, estimations can be used. Applications that can provide more accurate requirement information should be able to override these estimations. An example how such an API can be build is given in Section 3.5.7.

3.5.4 Operating System

In general OSs have the same possibilities than platforms, can use the same information and offer the same API to the applications. When there are applications that do not use the platform, but use the functionality of the OS directly, this OS should offer the same API for formulating QoT requirements and should try to use sensible defaults for applications not supporting QoT.

However, classical OSs offer a lower level of abstraction in the interface to applications than platforms. This means that platforms provide higher level functionality to the application, which had to be implemented inside applications for classical OSs. Therefore OSs have less information about what an application is doing. Estimations of an OS about the requirements of an applications transactions are therefore less accurate.

3.5.5 Device or Data Terminal Equipment

Several device sensors can provide relevant context information. In most cases, the device is not able to use or signal any information, instead, this information can be used by the applications, the platform or the OS.

- Is the devices screen on?
- The device orientation sensor gives hints on how the user is using his device.
- Some devices contain a proximity sensor which senses, whether the device is held to the ear and the devices screen is held to ones cheek. In that case, the screen and the touch screen are switched off to prevent the recognition of bogus touch gestures.
- Some devices have a camera facing the user. This camera can be used to recognize whether the user is looking into the direction of the device. The range camera in the popular Microsoft Kinect [Mic10] can even recognize the position of things, and therefore also the user's head position, in 3D.

In addition, some DTEs, for example appliances, smart meters etc. typically have a thinner user interface than applications on Personal Computers (PCs). Usually, such appliances perform a limited number of well-defined task and therefore have a limited number of different transactions. Examples of appliances are weather stations that upload sensor readings and download weather forecasts, digital photo frames that download images or Set-top boxes downloading or receiving TV programs or video.

3.5.6 Residential Gateway

Routers that forward traffic of one or multiple DTEs can also modify or add the transaction information. As DSL is currently the most widespread access technology, these devices are commonly referred to as "DSL routers".

Such a forwarding device can add default transaction information for transactions originating from DTEs that have not yet deployed QoT. The forwarding device might use the following context to create transaction information:

- The device type of the DTE. This might be configured manually by the user or might be automatically detected during the host configuration of the corresponding DTE. For example, the forwarding device can use the information the DTE provides in its Dynamic Host Configuration Protocol (DHCP) request.
- The type of the service each DTE provides. This can be also either configured by the user or detected form service announcement and service discovery protocols. These discovery protocols are typically used, so that the user's PC or smartphone can use the services

offered by other DTEs or vice versa. There are many different protocols and implementations for this task that are usually not compatible. Therefore a forwarding device will need to understand many of these service discovery protocols, to be able to reliably detect the service type of the DTEs transactions. Universal Plug and Play (UPnP) is the probably the most popular protocol for this task, as it is partially implemented in many multimedia appliances and residential gateways, mainly to enable dynamic port forwarding through the Network Address Translation (NAT) of the residential gateway for streaming and P2P applications. Service Location Protocol (SLP) is standardized by IETF, but did not gain general acceptance. Zeroconf was an unsuccessful standardization effort[2], that is partially implemented by Apple Boujour, Microsoft Windows and the OpenSource project Avahi. For Java programs, the Jini framework offers similar functionality.

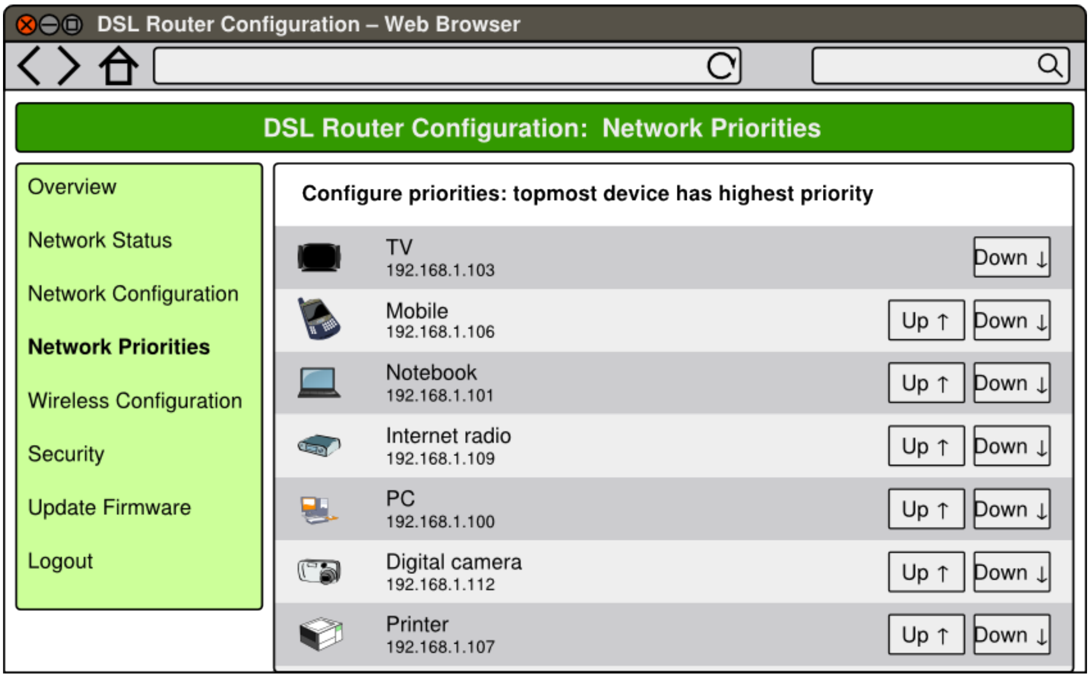
- The forwarding device can build up statistics about the history of each DTEs network usage and use these to attribute its future transactions. Such statistics are for example useful on the size, duration, bitrate, and used ports of transport layer connections. When the forwarding device uses stateful NAT, these necessary values are already determined.
- Instead of building statistics on transport layer connections, the forwarding device can use DPI to determine the content of each connection.
- When a DTE uses a different QoS mechanism, for example UPnP-QoS [HMG$^+$05], the forwarding device might receive and translate these information into QoT information.

Further, a forwarding device can also offer its user to manually configure some aspects of the transaction requirements, for example the user could configure the relative priority of the DTEs it is forwarding traffic for. Forwarding devices are typically configured using a configuration web interface. Figure 3.7 shows an example, how such a prioritization might be configured. The user simply states his preference of the devices and can move his devices up or down in this list. This list however rather configures weights than strict priorities. A transaction with real-time requirements will still have higher utility values, than a bulk transfer from a device higher in the list. Advanced settings might enable the user to configure the weights for each device manually. In addition, statistics might be displayed for each device. For example the transmitted and received data volume, and how often traffic from this device has blocked traffic from other devices in the list.

3.5.7 Implementation Considerations

On many levels, the transaction information can be transported using an API instead of sending these information directly using the signalization protocol given in Section 3.2.4.

[2] The IETF working group Zeroconf was closed without providing a standard: `http://datatracker.ietf.org/wg/zeroconf/charter/`



DSL Router Configuration: Network Priorities

Configure priorities: topmost device has highest priority

TV	192.168.1.103
Mobile	192.168.1.106
Notebook	192.168.1.101
Internet radio	192.168.1.109
PC	192.168.1.100
Digital camera	192.168.1.112
Printer	192.168.1.107

Figure 3.7: Example for a configuration page of the priorities in the residential gateway.

- OS functions for networking
- Platform libraries for network communication
- Scripting services and environments

Specifying QoT information in an API allows the application to describe its requirements together with the networking request. For example, in an API that allows an application to retrieve data using HTTP, the application does not need to specify the transport layer parameters, instead the library implementing the API can insert these information when assigning the corresponding request to a transport layer connection.

3.6 Deployment Scenarios

There are several possibilities, how QoT can be deployed on the subscriber's side. Many of the information sources listed in Section 3.5 are optional or provide defaults in case one the other information sources cannot be used. These alternatives are intentional and provide flexibility, which of the components implement QoT, because it is unrealistic to expect for example all applications to implement QoT at the same time.

In the simplest scenario, only one or multiple relevant applications implement QoT and signal their transaction requirements directly to the headend. All other applications as well as the OS or platform are not aware of QoT. The headend assumes default values for all other applications.

A second option is a system-wide component on the DTE, the TM. This component can be provided by the OS or the platform.

A third option is that a forwarding device uses its information sources specified in Section 3.5.6.

- It intercepts the transaction requirements sent from the connected DTEs.
- It should make use of as many as possible of the information sources listed in sections 3.5.6 to provide sensible default transaction information for DTEs that do not support QoT.
- It can provide user configuration as specified in Section 3.5.6.

Examples for such devices are WLAN Access Points (APs), switches, and routers. Of these forwarding devices only the residential gateway is significant for QoT, because it is the device that terminates the access network from the subscribers side and all traffic the subscribers DTEs sent and received over the access network is routed through the residential gateway.

This flexibility and modular setup is possible, because the signaling protocol described in the following Section 3.2.4 defines an IP anycast address that is used as the destination for all signaling messages. If for example the residential gateway supports QoT, it receives IP packets to this anycast address and evaluates them. Otherwise it will forward these anycast packets to its default route to the headend.

3.7 Summary and Conclusion

Based on the QoT concept in Chapter 2 an architecture is developed in this chapter. This architecture specifies the functionality of the required components, and the information that need to be exchanged between these.

The information about transactions and their requirements can be obtained from a number of sources. The applications are the most valuable information sources, because the applications define the user's workflow and know which result the user is waiting for. Modern platforms offer functionalities that were previously offered by applications, therefore platforms can provide most of the information, which is important because it allows a large number of applications to be supported at once.

These transaction requirements are sent to the headend using a signalization protocol specified in Appendix D.

4 Evaluation Methodology

The goals of the evaluations in Chapter 6 are to analyze the performance of access networks using QoT, compare it to other approaches, analyze the robustness of the approach and analyze the effects of variants and parameters of QoT.

In this chapter, the simulation models, the scheduler models and the relevant metrics for the evaluations in Chapter 6 are introduced.

In Section 4.1, the general approach for the evaluations is introduced. Section 4.2 explains the simulation model and the additional components such as models for hosts and Internet nodes. There are many different scheduler models, listed and explained in Section 4.3. The evaluation metrics are explained in Section 4.4.

The performance of QoT and other QoS approaches are heavily dependent on the users' behavior and on the traffic, generated by the users. Also, to evaluate the users' experience, the users' behavior and the users' view need to be modeled. A traffic model in combination with a utility model is therefore introduced in Chapter 5.

4.1 Simulation Methodology

The evaluation of QoT in an access network is performed using network simulation on packet level. Requests for transmitting data are generated by traffic models that model the behavior of given applications. The data requests are split into packets that could be transmitted via TCP through the Internet. The forwarding of these packets is modeled to match characteristics of packet forwarding in the Internet, without implementing IP and TCP protocol engines. Packetizing data and forwarding single packets is necessary to evaluate scheduler implementations that perform scheduling decisions on packet level. Network simulations that represent single packets, are typically discrete-event simulations.

There are several more reasons for this approach:

- There are feedback loops between the network behavior and the traffic model. The model for the users' behavior influences the network, for example the data that is available at the scheduler. The network – especially the scheduler – influences the users' behavior, because in many cases, the user will send further data only after the previous data has been received. For example, in a realistic traffic model HTTP one data transmission starts only after another data transmission has been transmitted successfully.

Analytical approaches to decompose the problem into smaller consecutive problems are therefore not possible.

- Many aspects of the system are non-linear, for example the packetization of the transmitted data. And even if the transmitted data were not packetized, the transmission becomes instantly finished when all of its data has been received. These non-linear relations make both analytical approaches and optimization techniques significantly harder.
- The large number of involved components, separate transmissions of different traffic types, and the degrees of freedom of the packet scheduling in the access headend also makes analytical formulations unfeasibly extensive.
- The traffic and user models are not deterministic, which requires stochastic methods for analytical approaches and optimizations.

These characteristics can be easily handled using discrete-event simulations: The simulation of feedback loops is straight forward and non-linear models fit the algorithmic nature of simulations. Non-deterministic models are a common thing in event-driven network simulations, although it may require extensive computational efforts for sufficiently reliable results. The combination of stochastic methods with non-linear models and feedback loops is suitable for discrete-event simulation. Under these circumstances other approaches would be unfeasibly complex.

4.1.1 Independent Scenario Repetition

For the types of simulations used in this work, the scenario has an influence on the resulting metrics. Especially the traffic model uses many random values drawn from long-tailed distributions. These random variables influence the number of transactions, the size of each transaction, the traffic characteristics and requirements of each transaction.

Most of the metrics used in this work are highly correlated to these scenario parameters. Averaging or aggregating these metrics over longer simulation times will usually reduce the variance. However, even when averaging or aggregating over long periods of time, there is still a huge variance in traffic characteristics. This matches the expectation: For example, the traffic volume of one broadband subscriber varies from month to month, although this metric is an aggregation over the course of one month (about 2.6 million seconds).

The performance evaluations in this work therefore repeat each simulation multiple times using independent random values for the scenario parameters. These repetitions are generally called *seeds*, because they are generally realized by seeding a deterministic pseudo-random number generator [LK00]. The scenario parameters of all seeds are independent from other simulation runs. Within one longer simulation run, the metrics are correlated over time. If one would take the metric aggregated over the first and over the second half of one simulation run, these values will be correlated and it will be hard to determine the amount of correlation a priori. Therefore, these independent scenario repetitions will therefore reduce the variation, because there are more completely independent measurements for the same simulated time.

For each metric, a combined value using all seeds is calculated. Additionally, a confidence interval for each metric is calculated using the Student-T-Test. If not noted otherwise, a 95% confidence interval is used throughout this work.

Preconditions for the Student-T-Test are:

- All values have to be independent. This condition is clearly met, because all random variables of the scenario are independent and there are no values otherwise correlated between the seeds.
- All values are from the same distribution. This condition is also met, since there are by definition no other factors that influence the results.
- All values are from the same, normal distribution. This condition is met due to the central limit theorem, when the number of measurements of the corresponding metric is large enough. How the underlying distribution converges to a normal distribution can be estimated from its third momentum (Berry–Esseen theorem).

4.1.2 Variance Reduction

Random values in the scenario are depending on the type of parameter and the seed for each simulation run. This means that the scenario parameters of different simulation runs with the same configuration are independent. But at the same time, this means, that the scenario parameters will be equal on simulation runs with the same seed but different configuration.

For example: the simulations with the same traffic configuration and the same seed but with different schedulers, will have the same traffic. This is a well-known technique to reduce the variance of simulations [LK00].

However, the traffic model used in this work is not in all cases behaving equally, when its random variables have the same value. The exceptions with the biggest influence is, that the model for the *P2P* traffic class starts new transactions depending on the time when the last transaction finished.

Therefore the calculation of the confidence interval in this work uses the conservative approach and does not take the variance reduction techniques into account. This means that the difference between two values (e.g. different schedulers) will often have a higher significance level than the confidence interval suggests.

4.1.3 Transient Phase

Discrete-event network simulations, typically employ a transient phase at the beginning of each simulation run. During the transient phase, the simulation model and all simulated components have their normal behavior and properties, but all measurements of metrics are discarded. [LK00] The transient phase should be long enough, so that transient effect had enough time to decay.

The goal of this technique is to obtain only measurements from a steady state. This is useful, to avoid influences from transient effects onto the resulting metrics. It is also important for statistical analyses that require all measured values to originate from the same distribution.

Because of the long-tailed distributions in the traffic model introduced in Section 5.3, an unfeasible long transient phase would be required. Therefore, the implementation of the traffic model takes great care to start each traffic class model as close as possible to the steady state. The implementation therefore calculates the proportion of the application states (for example *On* and *Off*), and starts in each state according to its steady-state probability. For the remaining duration of the state or the residual size of each transaction, the forward recurrence time of the relevant distributions is used. Therefore, the transient effects that need to settle before starting the measurements, need a short period of time to settle. The main two effects are:

- Some traffic class models need to transmit more data for the start of a transaction.
  The traffic class models are implemented to initialize its state so that the mean initial state matches the stead-state as close as possible. However the overhead at the start of each transaction is not omitted for transactions that are initialized with the corresponding residual size. This means, that transactions, that run immediately from the start of the simulation will cause some additional traffic: *Web* transactions need to send the request, and the source of a *Streaming* transactions will transmit faster within the first seconds, to fill up the buffer in the receiver.
- Buffers in the network are initially empty, while they are filled to some degree in the steady-state.

4.2 Simulation Model

In this section the simulation model is explained, which is used for the evaluations in Chapter 6. The main model components are shown in Section 4.2.2 and common building blocks of the model are explained in Section 4.2.4. The architecture for modeling traffic and utilities is explained Section 4.2.3, which is used by the traffic and utility model in the following sections

5.3 and 5.2.

4.2.1 Abstraction Levels

The evaluated system QoT is an approach for improving Internet access networks. Therefore the components of the access network are modeled in detail, while Internet servers and core network are modeled more abstracted.

QoT focuses on transactions and their context from a user's perspective. Therefore applications on the devices of subscribers are modeled in detail.

Applications are modeled in detail. Several different application models each cover certain types of end user applications. An application model for real-time streaming for example covers VoIP, live video, and other types of real-time traffic which CBR traffic. Each instance of an application model has internal states that model the behavior of this application. For each user model, one or multiple of this application models are instantiated. Each simulation typically contains several subscribers. For each subscriber an independent user model is instantiated.

The applications generate chunks of data, which is segmented into packets. The event driven network simulation library handles these packets. While the access network components are modeled in detail, core network and servers are modeled to form a single effective delay. Cross-traffic is also only considered at the access network.

Transport layer effects are not modeled, but taken into consideration. It is assumed that the transport layer provides the following abstraction:

- No congestion, or lost packets on the core network
- No delay and no additional traffic for connection setup, bitrate probing, and connection tear down
- Active transmissions always build up a queue in front of the bottleneck, this queue never gets empty until the end of the current transmission

4.2.2 Model Components

Figure 4.1 shows the active components that are involved in the transmission of data from or to an Internet access subscriber. These components form five groups that are sketched in Figure 4.2:

- The subscriber side models the user's behavior, the user's DTEs, further CPEs such as forwarding devices and the residential gateway
- The access medium, with its limited resources and characteristics
- The access network headend

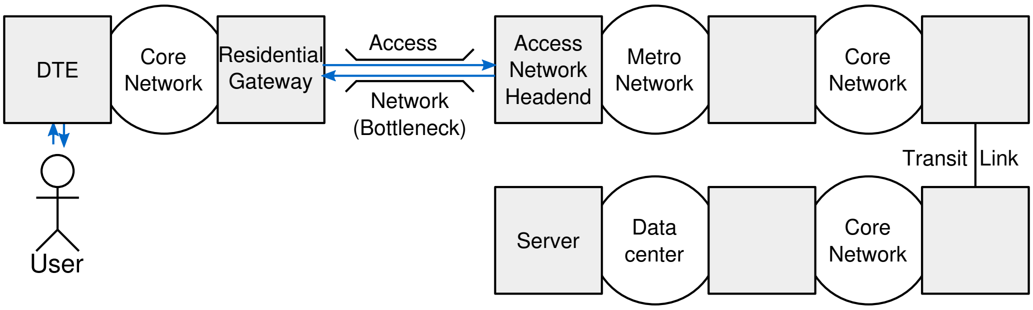


Figure 4.1: Transmission path on the Internet on IP level.

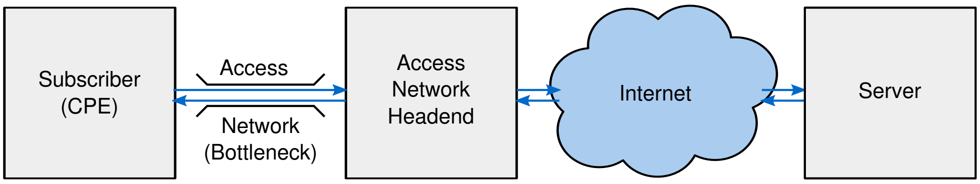


Figure 4.2: High level components of the model to evaluate QoT.

- The server that provides services for the Internet access subscribers
- All networking components that connect the access network headend to these servers

These figures use symbols to depict the data flow, which are listed in Appendix A.

Subscriber

The model for the subscriber side of the access network has to cover a wide variety of use cases:

- There might be one or multiple users sharing the access line
- One user might possess one or multiple devices (DTEs)
- There might be multiple applications running in parallel on one DTE
- A DTE might be connected directly to the access medium (e.g. using a Modem), or there might be one or multiple forwarding devices connected in between, for example a residential gateway

Most of this flexibility is provided by the traffic model introduced in Section 5.3. Most of the traffic model components model follow an inherent infinite source characteristic, so that the number of actual devices that originate this traffic is not significant.

Figure 4.3 shows the structure of the model of a subscriber. Multiple applications create traffic. This traffic is separated into transmissions, which are also buffered separately. A scheduler selects how the transmissions are multiplexed on the Uplink of the access medium. In case the headend manages also the resource assignment of the Uplink direction on the access network,

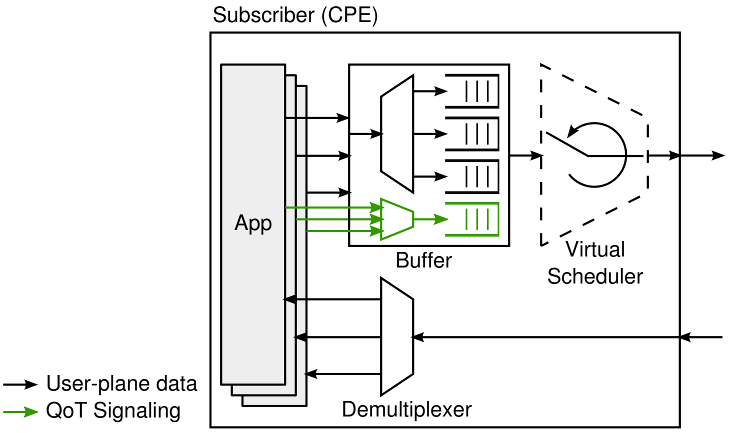


Figure 4.3: Model for an access network subscriber.

the scheduler on the subscriber side is replaced by a virtual scheduler that is partially or fully controlled by the headend. In that case, a central scheduler on the headend is informed about the bitrate demands of this subscribers transmissions. Access networks where the Uplink resources are arbitrated by a central component, have features to signal these demands and the resource assignments on lower layers with very little extra delay. This extra delay is therefore not modeled explicitly, but is included in the Uplink delay of the access medium.

The delay on the user's Local area network (LAN) is generally negligible and the capacity of it is almost never limiting the transmissions. Therefore it is inconsiderable, whether the applications of the subscriber are running on the same or on different DTEs.

QoT signaling messages are inserted into the user-plane packets. But typically, the signaling messages will be prioritized against user-plane packets. The delay of the QoT signaling is therefore less or equal to the delay of user-plane packets from the application to the access network headend.

Access Medium

The access medium is modeled as two simple links with a fixed data-rates and fixed delays, sketched in Figure 4.4. The maximum available data-rate is R_{UL} in uplink and R_{DL} in downlink direction. The delay on the access medium models the propagation delay, the processing delay and delays that are introduced by signaling on lower layers of the corresponding access network technology. This delay is modeled as a fixed value Δt_{UL} for uplink and Δt_{DL} in downlink direction. An inverse multiplexer distributes the packets in the downlink direction to the corresponding subscriber.

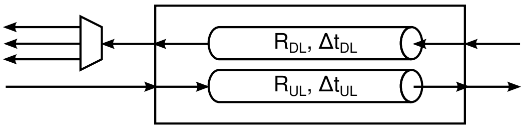


Figure 4.4: Model for the access medium.

Access Network Headend

The model for the headend is pictured in Figure 4.5. Handling for the Downlink and Uplink are handled separated. Packets in the downlink direction from the Internet to the subscriber are separated into transmission and buffered separately. In the downlink direction, the packets that the scheduler selected for transmission, are directly sent onto the access medium.

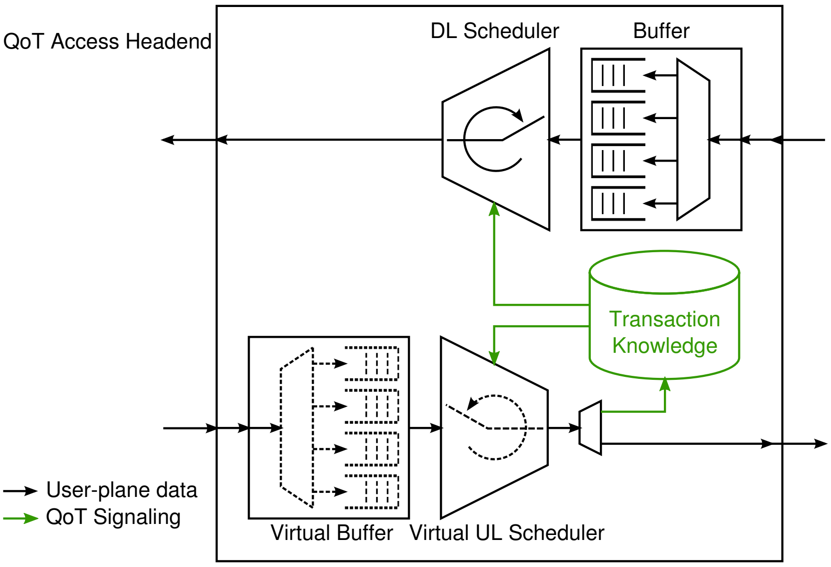


Figure 4.5: Model for the access network headend.

In the uplink direction, a scheduler is controlling the virtual schedulers on the subscriber's side. QoT signaling messages are separated from user-plane data, and recorded in a TM. The schedulers use the transaction knowledge from the Transaction Manager (TM) and the current queue length from the buffer for their scheduling decision.

Downlink scheduler and the controller for the uplink schedulers can be integrated to a single component, sharing the scheduling algorithm.

All evaluated schedulers (see Section 4.3) implement a general prioritization for signaling messages, to avoid effects that result from different scheduler behavior because of different availability of the transaction information in the signaling messages.

Core Network

This model component comprises all networking components that connect the access network headend to servers that act as corresponding nodes for the transmissions of the Internet access subscribers. Although, these components are not just on the core network, for simplicity, the whole model will be called “Core network”.

For the evaluation of Internet access methods, the core network can be reduced to a simple model, depicted in Figure 4.6.

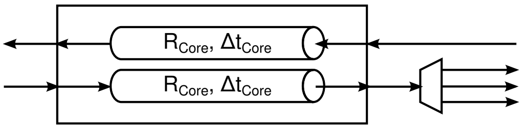


Figure 4.6: Model for the core network.

Packets are delayed symmetrically in downlink and uplink direction by Δt_{Core} and the maximum available data-rate is also limited symmetrically to R_{Core}. To evaluate access network procedures, the available data-rate of the core network is usually chosen so that it will be more than sufficient in all cases. An inverse multiplexer distributes the packets in the uplink direction to the corresponding server.

Server

The communication partner of the subscriber are servers providing a service. The model for such a server depicted in Figure 4.7 is in principle structured similar to the model of a subscriber: one ore multiple applications generate traffic, which is multiplexed on a common link by a scheduler.

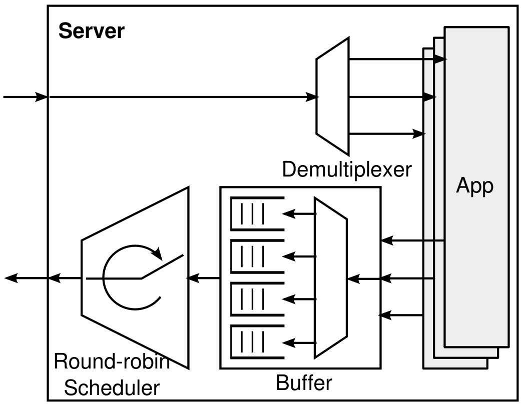


Figure 4.7: Model for the server.

For evaluations of the access network, the data-rate of the core network dimensioned so that it is not the bottleneck. Therefore even simple schedulers are sufficient in the servers. All evaluations in this work use a Round-Robin scheduling (RR) scheduler, which distributes the available data-rate equally to all queues.

4.2.3 Traffic Model Architecture

This section explains, how traffic between the subscribers and the servers is modeled. A traffic model that uses this architecture is introduced in Section 5.3.

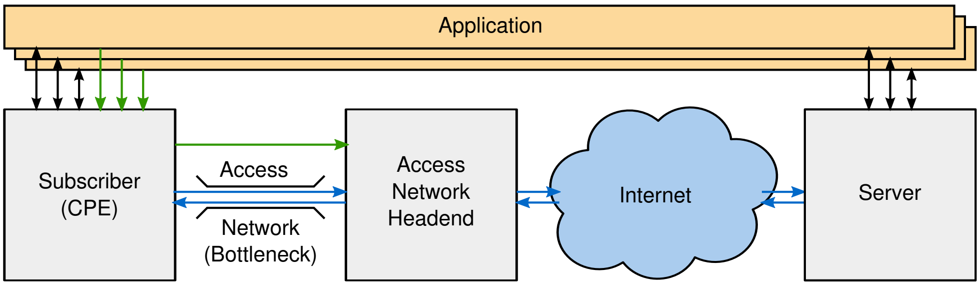


Figure 4.8: Model for applications.

As depicted in Figure 4.8, applications are modeled as global instances that communicate on both sides of the connection: from the subscriber's side, and from the servers. This structure allows to implement a realistic application behavior, without actually implementing the applications protocol.

In Figure 4.9 the relationship between subscriber model, application model, transactions, chunks and packets is depicted. In a steady-state simulation, subscribers, servers and applications are instantiated before the simulation is started. Transactions, Chunks and Packets are instantiated at runtime, according to the procedure of each application class model, explained in Section 5.3.

Applications Each subscriber has an arbitrary number of instances of an application model, in the following such an instance is called "application". Each application belongs to exactly one subscriber. Each application is also connected to exactly one server, while each server can handle multiple applications.

Transactions Transaction models directly correspond to the definition of transactions in Definition 2.2 and the description in Section 2.4.

Chunks According to Definition 3.2, a chunk is a part of a transport layer connection, either in uplink or downlink direction. As chunks are defined to allow recognizing the transaction of a packet, a transaction has to consist of one or multiple chunks. A protocol for signaling transaction requirements (see Section 3.2.4) may allow, that a chunk is part of multiple transactions, this model for evaluation does not generate such transactions or chunks.

The chunks of a transaction do not necessarily need to originate from the same transport layer connection, in fact the mapping of chunks to network layer connections is not part of the evaluation model. The transport layer protocol is assumed to be optimal and not to cause any additional delay for setting up and tearing down transport layer connections. This assumption is made, since there are already many mechanisms deployed, to reduce those delays (e.g. HTTP version 1.1) and there are will be future improvements to further reduce these delays. For example by improving HTTP using SPDY which is described in [Spd], is already implemented in many browsers, and a standardization proposal exists in [BP12]. Also by speeding up TCP by tweaking parameters such as the initial congestion window and by adding functionality, for example TCP Fast Open (TFO) which is explained in [RCC^{+}11] and a standardization proposal exists [CRJC11].

Segments A segment is a part of a chunk. Chunks need to be segmented into packets, because the network simulation model used in the evaluation is a based on packet transmission.

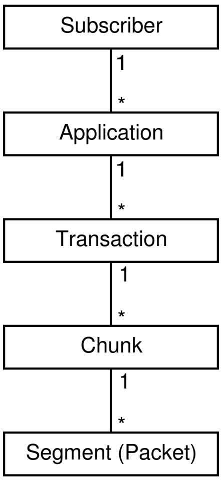


Figure 4.9: ER diagram of applications, transactions, traffic chunks and packets.

4.2.4 Model Building Blocks

The models for subscriber, headend, and server contain common blocks. These common blocks are reused multiple times in the simulation model.

- Buffering data separately for different transactions is done with a chunk buffer. Chunk-buffers are used in all modeled devices that handle transactions.
- Schedulers have a common interface to make them exchangeable.
- Transaction managers collect and offer transaction information.

Chunk-Buffer

The chunk-buffer could be considered a part of a scheduler, because it separates the received traffic into transactions or chunks, and queues them separately. To simplify the implementation of different scheduling algorithms, the chunk-buffer implements most of the common functionality, needed to support a scheduler.

- Receiving segments of chunks (packets) from a network link
- Matching and aggregating the segments to chunks, buffering its data
- Receiving partial or complete chunks from an application model
- Segmentation of the chunks
- Matching the chunks to transactions
- Offering an interface to query active chunks and transactions
- Offering an interface to query the available data (buffered size) of chunks and transactions
- Offering an interface to retrieve a segment of a chunk
- Keeping track of possible timeouts of transactions, cleaning up the buffered data
- Recording statistics about the received, buffered and transmitted data

In a real implementation, this would be implemented as per-flow packet buffer or a transport layer connection proxy buffering the data of each transport layer stream. The differences and characteristics of these implementation choice is detailed in Section 3.3.1. The chunk buffer model for the evaluation corresponds to a transport layer proxy, which means that effects from buffering transport layer control messages are not modeled.

Since the chunk-buffer implements segmentation and aggregation, it is also used in front of simple RR schedulers inside the server model and also in those cases, when the subscriber model or the headend is modeled without QoT. The application models generate large chunks of data, which have to be segmented before transmitting. In case QoT is also used in the uplink, all chunk buffers in the subscriber models are accessed by a virtual chunk buffer of the headend, which offers the same interface to the scheduler, by querying and combining the information of the subscriber's chunk buffer.

Transaction Manager

The Transaction Manager (TM) offers an interface to query the utility functions, the association of chunks to transactions and the sizes of chunks and transactions.

There are several implementations that have different sources for this information:

- A TM that uses the information directly from the transactions and therefore has ideal knowledge. Such TMs are used for the evaluation of the service quality. Such TMs are also used for models where a scheduler has ideal transaction knowledge.
- Delayed TM, which is used to implement the delayed availability of transaction information. It retrieves its transaction information from a second TM, but delays the information.

- Signaling TM, which retrieves its transaction knowledge from signaling messages, sent by the subscriber models. This TM corresponds to the component used in a real implementation of QoT. It collects the signaling messages containing transaction information. In a real implementation, the TM would also have to generate estimates for transactions, for which no signaling message have arrived yet.
- Degraded TM, which is one way to model degraded transaction knowledge. It A second method to model degraded transaction knowledge, is implemented in the application models.

It is possible, to chain the different TM implementations. For example, a Signaling TM can be chained with a Delay TM, to simulate an additional delay on the signaled messages.

Scheduler

Schedulers are the most influencing components for the evaluations. It is therefore important, to evaluate several schedulers and evaluate the same scenarios with different schedulers.

The interface for schedulers in the QoT evaluation model is designed to allow exchanging all schedulers in the model with a wide variety of different scheduler types.

Each scheduler that performs scheduling for one direction (uplink or downlink) has access to

- one a Chunk-Buffer that queues the data for each transaction separately
- one TM that provides the transaction information
- one data link, where the data is transmitted onto

There are also schedulers that perform scheduling for uplink and downlink direction together, such schedulers therefore have access to

Different scheduler models are explained in the following Section 4.3.

4.3 Scheduler Models

Schedulers are the main point of action for QoT. While all other components collect and gather information about transactions, the schedulers perform the decisions based on these information.

After the formulation of the problem for such a scheduler in Section 4.3.1, several schedulers are described. These schedulers are used and compared in the performance evaluations.

Beneath the QoT heuristic, several schedulers are for comparison:

- Round-Robin Scheduler (*Best effort*)

- Smallest-Size Scheduler (*Smallest*)
- Traffic-Class Scheduler (*Class*)
- Size-and-Class Heuristic (*QoT-Simple*), a second scheduler based on QoT information

4.3.1 Problem Description of a QoT Scheduler

The goal is to maximize the sum of the finish-time utility values of all transactions:

$$U_{total} = \sum_{T} U_T(f_T) \tag{4.1}$$

The linear optimization problem formulated in Section 3.4.2 is not a feasible solution for a real scheduler, for the following reasons:

- This optimization problem cannot handle uncertainties in the transactions sizes or utility functions. A solution to this optimization problem is therefore not the optimal solution, but the optimal solution under the assumption, that the current estimations are correct.
- Optimizing the utility of the transactions that are currently known to the scheduler does not take into account, that there might be additional transactions, starting while these current transactions are transmitted.
- Similarly, some of the transactions might be stopped by the user, for example by closing the application, or even by shutting down the computer.

Formulating an optimization problem that takes the probability functions for all these cases and uncertainties into account seems unfeasible. Even if it was possible, measuring and adjusting these uncertainties in a real world scenario seem too complicated. And even if this was possible, the computational effort for solving this optimization problem will be prohibitively high. Even for the simplified, linear optimization program in Section 3.4.2 solving real-world sized problems is impossible. It is therefore necessary to use heuristics for the scheduling decision.

The focus of this work is to show the potential of QoT, not to find an optimal scheduler. A good-enough heuristic is therefore sufficient for the evaluations in Chapter 6.

4.3.2 QoT Heuristic

This scheduler is a simple heuristic that uses the information, provided by QoT. This scheduler sorts the transactions by their relative, time-dependent, remaining importance $i_T(t)$ and in each scheduling interval t schedules data of the transaction T with the highest $i_T(t)$.

The importance of the transaction T at time t in this heuristic is depending on loss of utility when the transaction is finished later than expected relative to the size of this transaction.

FTs

$$i_T(t) = \frac{U_T(\tilde{f_T}) - U_T(t + n_f \cdot (\tilde{f_T} - t))}{{s_T^r}^{n_e}} \tag{4.2}$$

The expected finish-time $\tilde{f_T}$ is simply calculated as:

$$\tilde{f_T} = t + \max\left(\frac{s^r_{T,\mathrm{DL}}}{R_{\mathrm{DL}}}, \frac{s^r_{T,\mathrm{UL}}}{R_{\mathrm{UL}}}\right) \tag{4.3}$$

The parameters $n_f \in [0, \infty)$ controls, how much delay is assumed. When n_f is 1, the utility function is irrelevant for the importance, and only the remaining size of the transaction is considered. When n_f becomes infinitely, the currently possible utility value is compared with the utility value for dropping this transaction.

The parameter $n_e \in [0, \infty)$ controls how the size of the transaction is considered. When n_e is 0, the size is irrelevant and only the utility function is considered. When n_e grows against ∞, the size becomes the only relevant criteria. With $n_e = 1$, the size-dependency of the utility values is neutralized.

In most scenarios, the transaction information are not known ideally, therefore estimations need to be used:

- $\tilde{s_T^r}$ instead of s_T^r
- $\tilde{U_T}(t)$ instead of $U_T(t)$

RTs

The utility of a RT depends on how many of its packets miss the latency requirements. The finish-time for the whole transaction is irrelevant, because the bitrate of the transaction is determined by the application, not by the network. The finish-time of each chunk however, is determined by the network.

Instead of the finish-time, the relative frequency of lost traffic chunks l_T is the key parameter in this case. The heuristic compares the utilities $U_{\mathrm{RT}}(l_T)$ depending on the rate of lost packets up to now ($l_{T,m}$) with the case where the following n_r chunks are to late.

The best (minimum) achievable rate of lost chunks up to now is $l_{T,m}$

$$l_{T,m}(t) = \left(1 - \frac{s_T^r}{s_T}\right) l_T(t) \tag{4.4}$$

For the remaining importance $i_T(t)$, the mentioned utility difference is put into relation with the size of the current chunk:

$$i_T(t) = \frac{U_{\mathrm{RT}}\left(l_{T,m}(t)\right) - U_{\mathrm{RT}}\left(l_{T,m}(t) + \left\lceil \frac{n_r s_T^r}{C_T s_T} \right\rceil\right)}{s_C^{n_e}} \tag{4.5}$$

For RTs, the utility of a whole transaction is effectively determined by just a few chunks, because a small number of missing chunks is sufficient to decrease the users experience significantly. It is clear, that the relative importance of a RT will exceed the one of FTs because the utility of a RT depends on each individual chunk, while the size of a single chunk is rather small. As a result, RTs is preferred by the scheduler in most of the cases. This is the desired behavior.

For comparable results with other schedulers, the scheduler will transmit all chunks, even if a chunk is already too late. In general, dropping parts of a RT requires more knowledge from the application. As mentioned in Section 1.2.1, there are several systems for such an approach. However it is not the goal of QoT to make use of such a technique, instead QoT can be combined with it.

STs

For STs the relative importance of the transaction is derived from the difference between the currently best possible result and the worst possible result.

- The best possible result $U_{T,\mathrm{now}}$ is the utility, the transaction would get, if there were no additional delays for the remaining transaction. That means the utility of the transaction is calculated as if the current number of rebuffering events B_T and the current rebuffering duration b_T are valid at the end of the transaction.
- The worst possible result $U_{T,\mathrm{drop}}$ is a fixed value for each transaction.

$$i_T(t) = \frac{n_s \left(U_{T,\mathrm{now}} - U_{T,\mathrm{drop}}\right)}{{s_T^r}^{n_e}} \tag{4.6}$$

Because the worst possible utility is used to calculate the relative importance, the relative importance will generally be higher than the relative importance of a RT.

Heuristic

The algorithm for the heuristic of this scheduler is shown in Algorithm 4.1. It is obvious, that the heuristic can be implemented efficiently, because it calculates a single scalar value for each transaction and schedules the transaction with the highest value.

Algorithm 4.1 Scheduling heuristic

```
procedure SCHEDULING DECISION (list $T_{1..N}$, Buffer)
    if $|\text{list}T_i| = 1$ then
        return $T_1$
    end if
    sort $T : i_{T_i}(t) > i_{T_{i+1}}(t)$
    return $T_1$
end procedure
```

In general it is desired to prioritize data of RTs are over STs and data of STs over FTs. This is achieved implicitly by the definition of the relative importance for the transaction types. Of cause, this is just a general tendency, but the relative importance of a FT may in some cases be higher than the relative importance of a RT. Such a case is for example, when the number of dropped chunks of the RT is already reached a level, where the utility of the transaction will be its worst possible value $U_{T,\text{drop}}$.

An effective heuristic for a scheduler without QoT is to prioritize RTs over other transactions. The heuristic for the comparison schedulers is to explicitly prioritize data of RTs over STs and data of STs over FTs.

4.3.3 Round-Robin Scheduler (Best Effort)

To model a scenario without any QoS support, a model is used that tries to reflect the current situation of Internet traffic. The model tries to include the effects from commonly used solutions to common problems.

Today, most of the traffic is transported using TCP. TCP tries to assign all transport layer connections a fair share of the resources on the bottleneck. This TCP fairness means that all connections achieve the same long-term average data-rate, which can be extended to protocols beyond TCP, see [HFPW03].

The scheduler model for this scenario assigns all transactions the same average data-rate on the bottleneck. This leads to the following model properties:

- Multi-connection download-managers, which try to cheat against the TCP fairness by using a large number of parallel connections are not used in this model, or their effect is canceled technically.
- P2P file sharing applications, that typically retrieve their data from multiple slow sources in parallel, obtain the data-rate that they would get using a single transport layer connection to a fast peer.
- There are no applications or services that overload the network, because of a missing

congestion avoidance behavior.

The scheduler algorithm used to achieve the same rate for all queues is a Round-Robin scheduling (RR) scheduler. According to its name, such a scheduler serves each queue once in each round. Each time the access link becomes idle, the next queue is selected. To avoid aliasing effects, a random order is used. This is therefore called a Random Round-Robin scheduler.

4.3.4 Smallest-Size Scheduler (Smallest)

For comparison, an extremely simple scheduling heuristic is used. It schedules always the transaction that has the smallest remaining size s_T^r. In most scenarios, the exact size of a transaction is not known at the scheduler, therefore an estimation of the remaining size $\widetilde{s_T^r}$ is used. Despite its simplicity, the results in Section 6.2 show that this scheduler achieves good results.

This scheduler uses the QoT transactions and transaction information, but it does not use the utility functions. It is therefore a good choice to assess the additional gain from using utility functions in addition to transactions.

4.3.5 Traffic-Class Scheduler (Class)

Another scheduler for comparison is the traffic-class scheduler. It assigns priorities to each traffic class: *Real-time* traffic is preferred to *Streaming* traffic, which is preferred to *Interactive* traffic, which is preferred to *Web* traffic, which is preferred to *Upload* traffic, which is preferred to *Download* traffic, which is preferred to *Background* traffic, which is preferred to *P2P* traffic.

Inside each traffic class, a RR scheduler is used.

This scheduler models a scenario, where a QoS schema similar to DiffServ is deployed throughout the Internet and both sides of each connection cooperate by setting fitting marks to each packet.

4.3.6 Size-and-Class Heuristic (QoT-Simple)

Based on QoT information, one can combine information about the size and the traffic class of a transaction.

This simple heuristic combines the two previous schedulers: Transactions are classified as in the Traffic-Class Scheduler application class. Inside each application class, the transactions are scheduled using the Smallest-Size Scheduler.

This scheduler does not use the utility functions of QoT, instead it only uses the overall utility of each traffic class (utility per byte) to order the traffic classes.

4.4 Evaluation Metrics

The simulations scenarios represent the situation at busy hour. At other times the bitrate of the bottleneck is less scarce.

The average active broadband usage in Germany in the year 2011 is two hours and 17 minutes according to [Bun11]. From the measurements in [FBC99, FML$^+$03] for aggregated links, the busy time of each day is between four and six hours. Measurements from 2012 in [San12b] show that for broadband access subscribers, the average bitrate during peak hour is 1.61 times greater than the average bitrate over a whole day. These measurements fit together, since the time each subscriber is using his broadband service is varying.

For some metrics, it is easier to relate to known values, when the reference duration is one month. Examples are the data volume per month and the monthly fee for the Internet access. To relate these metrics to a duration of one month, the busy-hour metrics are assumed to last for three hours each day, to account for additional traffic outside the subscribers' online active usage period.

4.4.1 Data Volume

The data volume or the average throughput of a link or of a network is not relevant, as it is determined by the traffic model. With the traffic model detailed in in Section 5.3 and Section 5.2, which is used for the performance evaluations, the offered data (total amount of data that is to be transmitted), is mostly independent of the network characteristics. Except for the *P2P* application class, all application classes generate new requests or transactions following a Poisson process, without regard whether there are still unfinished transactions. Since all these transactions will be transmitted later, the total amount of transmitted data is determined by these application class models.

The limits and exceptions of this assumption are:

- When the amount of offered data is greater than the capacity of the access link, the simulation will likely become non-stationary because the number of unfinished transactions constantly grows.
- The application class models define a drop-time for each transaction, see Section 5.4. At this point in time, the transaction becomes worthless for the user, for example the user has closed the corresponding application or browser tab and the connections are teared down. Except for *P2P*, in all application class models this drop time is many times the duration, it would normally take to finish this transaction.
- The headend or a scheduler may decide to drop a transaction.
- The *P2P* application class model defines a duration of an On-phase, which may not be

sufficient to transmit all elements, that where requested in this On-phase. The end of On-phase has the same effect as the drop-time: all unfinished transactions or transmissions are teared down immediately.

- The number of concurrent uploads of the *P2P* application class model are limited by a constant number. Thus the earlier one P2P upload transaction is finished, the earlier the next upload transaction can start.

Therefore the data volume or average data-rate only deviates from the value given by the traffic model, when the data-rate of the access network is not sufficient.

4.4.2 Utility

One of the underlying ideas of QoT is to formulate a measurable metric that approximates a user's experience. This metric is obviously also a metric relevant for the evaluation.

The relevant value is the utility value of a transaction after this transaction has been finished or has been finally dropped. These values can be aggregated for one application, for subscriber, or for all subscribers of the access network. This therefore equals to the metric that QoT tries to maximize, which is the sum of all utility values of all transactions.

Because this utility metric is a new metric that is not known and familiar from literature, it is important that the metric can be intuitively matched to everyday things. Therefore these utility values are related to a time period of one month and are normalized so that one utility unit matches one Euro. To obtain values for a reference period of one month, again three hours of the busy-hour simulation are assumed per day.

Beyond the utility that has been achieved after the simulation finished, the utility value that could be achieved by an infinitely fast network is also relevant for comparison.

To get more insight on the characteristics of certain scenarios, further metrics of utility values are viewed:

- Distribution of utility values
- Average utility value depending on the size of the transaction
- Average utility value depending on the utility value that could be achieved with an infinitely fast network
- The sum of the utility values of all transactions with a given size
- The sum of all utility values of all transactions smaller than a certain size

In general, utility values have a high variance, when the scenario has a high variance. The reason for this is, that the all utility values (ideal utility, drop utility, ...) of a transaction depend on the type and parameters of the transaction. The sum of the utility for multiple transactions is also affected by the number of transactions. The number of transactions and many of the parameters of the traffic model are drawn from long-tailed distribution.

Therefore the metrics derived from utility values also show high variances. To obtain statistically significant results, a very large number of samples is required.

4.4.3 Data-Rate

The short-time data-rate in most access networks is either 0% or 100% of the available bitrate, because packets are sent with line rate.

The long-time average of the data-rate corresponds to the data volume, which is given by the traffic model. See Section 4.4.1.

The data-rate for short time intervals can be used to visualize the scheduling behavior.

4.4.4 Observed Data-Rate

Instead, the observed data-rate for a subscriber or an application is evaluated.

Definition 4.1: ***Observed data-rate of subscriber***
The observed data-rate of a subscriber is the data-rate that he can use send or receive data on the access link, under the condition that there is currently data available to be transmitted for this subscriber in the corresponding direction.

Subscribers are assumed to have multiple devices and multiple applications transmit data in parallel. The applications should not interfere with each other. Therefore another metric is used to evaluate the data-rate that each application can utilize.

Definition 4.2: ***Observed data-rate of application***
The observed data-rate of an application is the data-rate on the access link used to send to or receive from this application, under the condition that there is such data available.

Since each application class model can perform multiple transaction in parallel, it is relevant to evaluate such a conditional data-rate alike for the transactions.

Definition 4.3: ***Observed data-rate of transaction***
The observed data-rate of a transaction is the data-rate on the access link used to send or receive for this transaction, under the condition that there is such data available.

Transactions can consist of multiple traffic chunks. To get a more detailed view on the performance of the network, it can be beneficial to evaluate the observed data-rate for each traffic chunk individually. The traffic model transmits traffic chunks without inherent pauses as specified in Section 4.2.3. The observed data-rate of a traffic chunk is therefore the quotient of the chunks size and the duration it takes to transmit it on the access link.

Definition 4.4: ***Observed data-rate of chunk***

The observed data-rate of a traffic chunk is the data-rate on the access link used to send or receive data for this chunk.

Characteristics

For fast networks, where transmissions are always finished before the next transmission starts, the observed data-rate converges to the data-rate of the access link.

When there is a large number of elements that are transmitted in parallel, each element only gets a fraction of the total data-rate. In case of a scheduler that distributes the available data-rate equally to all active elements, the observed data-rate for one element is the available data-rate divided by the number of active elements. Schedulers that approximate such a behavior are Generalized Processor Sharing (GPS) and RR schedulers. With TCP as the transport protocol for transmitting the elements, even with a FIFO scheduler such a behavior is approximated.

For RTs and STs the data-rate of each transaction is defined by the application class model. As long as the network is able to provide this data-rate, the observed data-rate is equal to the requested data-rate of the transaction.

Average Observed Data-Rate

When taking the average of such observed data-rate values, it is important to note, that this average will be dominated by the observed data-rate of small elements. The distribution of the sizes of the elements in the Internet is assumed to be a heavy-tailed distribution, and is therefore modeled using log-normal distributions in the traffic-model introduced in Section 5.3.

To overcome this, in the following chapter this metric is plotted depending on the element sizes. That means the metric is evaluated and plotted as a function of the size.

4.4.5 Latency

Latency here denotes the one-way delay it takes to deliver an element from the sender to the receiver. An element is either a single packet or an element on a higher layer. In most cases these higher layer elements are Chunks. Depending on the type of a transaction, different latencies are relevant.

Finish-time Transaction (FT)

For FTs, latency is relevant for higher layer element. This metric is meaningful as it is the base for the evaluation of the users' experience. The earlier the element is received completely, the better. When a FT consists of multiple elements, the latency can be measured for each element separately or for the whole transaction, which is the transaction finish-time. From the transaction finish-times, the utility values are derived.

When taking the average of such latency values, it is important to note, that this average will be dominated by the latencies of small elements. The distribution of the sizes of the elements is assumed to be a heavy-tailed distribution in the Internet, and it is modeled by log-normal distributions in the traffic-model introduced in Section 5.3. This means, that a few large elements account for most of the data volume, while the majority of elements are orders of magnitudes smaller. Since the finish-time of such an element will be proportional to the size, these few large elements dominate also this average.

When the finish-time of each transaction is set into relation to the corresponding transaction size, the resulting metric is equal to the reciprocal conditional throughput. This relation describes the time it takes to transmit a certain amount of data of an element. For Internet traffic with long-tailed distribution of sizes, the average of this relation is dominated by the characteristics of large number of small elements.

Streaming Transaction (ST)

For STs, a latency can be measured for transmitting one chunk, thus the data from one checkpoint to the next checkpoint. Latencies of single packets are less relevant, since STs usually apply larger buffers than RTs. A latency metric for the whole stream is also not meaningful, since the duration of the stream is stated by the playout curve and it is neither common nor necessary to provision the whole stream at the receiver.

The sizes of the chunks of the STs have a small variance and are not distributed with heavy-tailed characteristics. The traffic model described in Section 5.4.3 even models the size of these chunks to be deterministic. Therefore the average of the latency of the chunks can be calculated without problems.

Real-time Transaction (RT)

For RTs, the latency of each packet is the relevant metric that determines the user's experience. The utility value is usually derived from the latency through a non-linear function. For example a step-function: when the latency is above a threshold, this packet cannot be used anymore and has to be considered lost. Due to this non-linearity, the latency is therefore a distinct metric.

The packet sizes of RTs in the Internet do not show heavy-tailed characteristics. The traffic model in Section 5.4.1 uses a bounded normal distribution for the size of packets of the *Real-time* application class model and a binomial distribution for the size of packets of the *Interactive* application class model.

However, since these packets are usually transmitted atomically over the network, the only delay component of a single packet that is proportional to the packet size is the serialization delay. Since the serialization delay is only a negligible part of the overall latency, the average of the packet latencies is neither dominated by the small nor the large packets.

Combinations

When combining latency metrics or throughput metrics for all types of transactions, one has to keep in mind which of the transaction types and application classes will dominate the result. Due to the long-tailed distribution of sizes, a few huge elements will dominate metrics proportional to the size of the elements, while the huge number of small elements will dominate metrics that are not proportional to the size of the elements.

To overcome this, in the following chapter some metrics are evaluated in dependence of the element size. That means the metric is evaluated and plotted as a function of the size.

4.4.6 Drop Rate

As mentioned above, when the data-rate of the access network is not sufficient, some of the transactions will eventually be dropped. The number of dropped transactions is therefore a metric for this. For acceptable service quality, the number of dropped transactions should be small, for example smaller than 1% of the transactions. Since the size of a transaction has influence on the drop probability of some schedulers, the drop rate can be viewed depending on the transactions size.

4.5 Summary and Conclusion

In this chapter, the simulation models, the scheduler models and the relevant metrics for the evaluations in Chapter 6 are introduced.

For the evaluations, an event driven simulation approach is used. The simulation model consists of components that are relevant for analyzing QoT. Since QoT is an approach concerning the access network, the components around the access network are modeled in higher detail than other components that forward the subscriber's data in the Internet. Those components are:

- A model for the subscriber, including a traffic model that models the users' traffic and a model for the subscriber's devices, e.g. the DSL router
- A model for the access medium
- A model for the access network headend including a scheduler
- A model for the core network between the access network and the corresponding server, including the Internet

- A model for servers that are corresponding nodes for connections of the subscriber

The scheduler in the access network headend is either a QoT scheduler or one of multiple comparison schedulers.

The model of the subscriber contains a model for generating traffic. This traffic model is introduced and described in the following Chapter 5.

5 Traffic Model

In this chapter, a traffic model in combination with an utility model is introduced. This combined model is necessary for the evaluations in the following Chapter 6.

The goals of the evaluations in Chapter 6 are to evaluate the performance of access networks using QoT, to compare it to other approaches, to analyze the robustness of the approach and to analyze the effects of variants and parameters of QoT. The performance of QoT and other QoS approaches are heavily dependent on the user's behavior and the traffic. Also, to evaluate the user's experience, the user's behavior and the user's view need to be modeled.

Section 5.1 describes the modeling approach of the developed traffic model. Section 5.2 introduces a model for the requirements of transactions. These requirements are combined with a model for the user's experience, depending on how the transmission of a transaction has fulfilled the requirements.

In an access network the aggregation level is quite low. The number of subscribers sharing one access medium ranges from only one to several hundred subscribers. Therefore, a traffic model for a single subscriber is required, which is explained in Section 5.3. This traffic model is utilized in simulations that model the Internet connections from the servers to the subscribers.

The traffic model consists of seven separate application class models. Each application class model models represents the traffic of a group of applications with similar traffic characteristics. These seven application class models and their respective models for the user's experience are described in Section 5.4.

The parameterization of the simulation model used in Chapter 6 is listed in Section 5.5. In Section 5.6, the traffic models introduced in this chapter are analyzed for their characteristics to meet the requirements for the evaluation of QoT in Chapter 6.

5.1 Modeling Approach

There is no standardized traffic model that can be used for the evaluations in this work. The network topology of access networks is different for each operator. Each user is different, each application is different, and each device (e.g. computer) a user uses is different. The results of empiric studies for Internet traffic and the users' perception of Internet services therefor vary widely. Some examples for this variance in empiric studies are listed in Section 5.4, when searching for the characteristics of certain traffic types.

In the worst case, empiric studies could yield completely different results, if the measurement was taken one day later or one day earlier. Reasons for such drastic deviations can for example be:

- A popular program or a popular website changed its internal protocols or its internal structure
- A popular service might be unavailable
- An extraordinary long or extraordinary short video is spearing virally on this day

When the evaluation of QoT and the comparison schedulers are done with empiric measurements, for example traffic traces, the results would be only valid for the exact situation, where the empiric measurements were performed. One remedy could be to obtain many different traffic traces from very different sources and situation. This approach proved to be impossible:

- Pure network traces do not reflect the behavior of the application and the user. It is therefore necessary to also record internal state of the application and the user's actions, in case this is possible.
- Recording network traces invades the connected users' privacy. The legal challenges for recording traces in networks with a larger number of users make this approach virtually impossible. Either all users of that network need to explicitly agree to the measurements, or the measurements need to be anonymized to a degree, where the transactions and the interrelation between transactions are not visible any more.
- The recording of traces that include the application state and the user's actions, usually requires modifications to these applications and the user interface. Such a modification cannot include all possible applications, the users are using. The number of applications used by a larger set of users is huge, the effort to obtain measurements from all these applications is prohibitively high. For many applications, it may even be impossible, when there is no source code available and the application does not offer any usable interface to obtain information that can be used to derive the transactions.

It is therefore necessary to find generalization for the relevant characteristics of the networks, the devices, the applications and the users. This means, that it is necessary to make assumptions about these characteristics. The main challenge for the evaluation models in this work is therefore, to make the evaluation results robust against these variations in the characteristics and therefore robust to deviations of the assumptions.

To be able to obtain results, the model therefore needs to replicate the general characteristics directly without artificial simplifications. This can only be achieved, when the traffic class models from Section 5.4 are implemented algorithmically. An algorithmic model is easier to understand and can be compared to real implementations. Beyond that, an algorithmic model requires less abstract assumptions, because the behavior is defined by the algorithm.

For example: assume a request/response protocol such as HTTP. In an algorithmic model, when requests are delayed, the delay of responses is automatically correct, because the response is only sent after the request has arrived. For a non-algorithmic model, this delay has to be modeled differently. A simple approach would be to neglect this delay.

5.2 Modeling Transaction Utility

In this section a model of the user-centric metric *Utility* of a transaction is developed. As mentioned in Section 1.4, network-centric metrics do not reflect the user's experience. Therefore the concept *utility of a transaction* was introduced in Section 2.5. In this section, a model is developed to use such utility values in network simulations.

5.2.1 Overview

It is probably impossible to measure the value or utility, a given transaction has to the user. There are two reasons for this:

1. It may be impossible to query the user's opinion on a single transaction without interfering with the user's attention to the corresponding application.
2. Even if it was possible to query the user's opinion, the user is probably not able to indicate the value or utility, this single transaction has to him.

There are studies (e.g. in [NUN10] and [Int05]) where users are asked to indicate their level of satisfaction, depending on the time the users need to wait for a result, similar to a transaction. To the author's knowledge, there are no studies that measure the users' valuation directly. The level of satisfaction however is a completely different metric, as discussed in the next Section 5.2.2

Furthermore, it is difficult to distinguish between the users' valuation for the transmission of a given data volume, the data-rate for this transmission and the connectivity, thus the general possibility to transmit data whenever the user wants to.

There are several studies analyzing the user's valuation for properties of an Internet access.

[SH00] showed in 2000, that customers will pay a significant surcharge for Internet service with addition functionality such as the capability to watch videos.

The utility values for positive and negative experiences are usually considered highly asymmetric. A negative experience, for example, will outweigh multiple positive experiences. [TC05] models this asymmetry using non-linear fuzzy integrals and Choquet integrals.

[ESR03] gives an overview over several models or the customer satisfaction for Internet service, with respect to properties of the customer support, ease of use, price, payment method and further metrics.

5.2.2 Utility, Expectation and Level of Satisfaction

At a first glance, utility and the level of satisfaction seem to be related metrics: an increased level of satisfaction implies an increased utility value, and vice versa.

However there are obvious differences:

While a larger or a more important transaction has more value to the user, the level of satisfaction is independent from such parameters. When for example a small download and a huge download are both finished timely, they yield the same level of satisfaction. The user's valuation for the huge download will be larger if a user would have to pay for each transaction separately, he would accept a higher price for it. Therefore the utility value for this second download is higher, while the level of satisfaction is the same. In general and independent from the size, there are items or transactions that are more important to the user than others. While the utility value will reflect this differentiation, the level of satisfaction does not.

The level of satisfaction is typically measured on an ordinal scale with five values: ”very unsatisfied”, ”unsatisfied”, ”neither satisfied nor unsatisfied”, ”satisfied”, ”very satisfied”. It is not reasonable, to calculate an average of multiple of these values. When for example a user is ”very satisfied” with his access network as often as he is ”very unsatisfied”, in total he will still be unsatisfied with this access network, because it is ”not working properly” every second time he tries to use it.

This asymmetry can be best explained by taking into account the user's expectation.

[KLL$^+$08] and [RRO98] suggest that there is a non-linear connection between the users' valuation and QoS parameters. In [LFJF05] there is even an exponential connection between the available data-rate and the perceived MOS.

These works are mainly based on information gathered using questionnaires. However none of these can predict a user's experience for a given transactions based on network metrics.

In general, an outcome that is worse than the expectation weights much more than an outcome that is better than expected.

This asymmetry is modeled using an asymmetry factor h_U. This factor defines, how many times one bad experience weights more than one good experience.

5.2.3 Utility Values and Transaction Size

The utility amplitude is the difference of the utility function between the utility for a good service quality for this transaction and the utility for a bad service or even dropping this transaction. This utility amplitude can be seen as the value of a transaction for the user.

The cost of a transaction for the access network is related to its size. Whether the user's valuation is also coupled to the size of a transaction is unclear. In some cases, a user is unable to know the size of his transactions. In other cases the user is able to know the size or it can be assumed, that the size of less important items is smaller.

Therefore, in the following, three different *utility models* are used for the evaluations:

Proportional The utility amplitude is proportional to the transaction size

Fixed A significant share of the utility amplitude is independent of the transactions size.

Basic A combination in between *Proportional* and *Fixed.* Unless otherwise noted, this combination is used in the simulations.

How these three approaches affect the scaling and shape of utility functions of each application class model is listed in the following sections.

5.2.4 Expectation

In this section, estimations are given for how a user's expectation depends on properties of the access network.

As of today, in most cases, users will know what data-rate they can expect from their access network. For DSL and CaTV the data-rate is still the main advertised competition feature. In addition, many programs such as download-manager of browsers and file sharing applications display the current data-rate. However this might change in the future.

To account for this dependency, many of the application class models require to calculate an expected finish-time $t_{T,\mathrm{exp}}$ based on the size s_T of the transaction T. Therefore a model parameter configures the bitrate R_{exp} that the user expects from his access network. Usually separately for uplink direction ($R_{\mathrm{exp,UL}}$) and downlink direction ($R_{\mathrm{exp,DL}}$).

5.2.5 Signaling Model

To use transaction information inside a simulation, each application model generates signaling messages for this transaction. At the beginning of each transaction, the application generates such a signaling message and sends it over the uplink of the corresponding user.

The application model includes all information that are known at this time. Application models that generate additional chunks afterwards, send an updated message each time one of the values is changed.

The size of a binary representation of these values is assumed to be constant 100 bytes per message and transaction.

To model additional delay or degraded knowledge, the signaled information can be delayed or degraded, using the building blocks, introduced in Section 4.2.4

5.2.6 Degraded Knowledge

The valuation of the user for a given transaction is not fully reflected by the utility model. Instead there are factors that influence the user's valuation that cannot be modeled reliantly. Examples for such other factors are the user's mood, the importance of a transaction in a workflow or whether the user is currently distracted. These other parameters are not modeled in the traffic model nor in the utility model, and are therefore modeled as random values.

- characteristic of the corresponding person
- the persons mood
- characteristics of the specific program or transaction
- specific context of the individual situation

To describe these influences, the parameters of the utility model are viewed as a superposition of variables that are known to the model and parameters that are not known to the model. From the perspective of the model, the parameters not known to the model are random.

There are two type of parameters that make up the utility functions:

- Utility values
- Time values

The parameters consist of the following contributions

- The part covered by the model
- A random part that is known to the real system, but not included in the model. This random component is drawn per transaction from a random distribution and is signaled and available to the scheduling algorithms.
- A random part that is neither known to the real system not to the model. This random value is also drawn per transaction from a random distribution, but it is not available to any of the algorithms. However it is used for the utility evaluation of each transaction.

Since it can be assumed, that these parameters combine multiplicatively, the random values are from log-normal distributions.

Variation in Time Parameters

There is a certain variation in patience of users which has influences on the time the user expects his transaction to finish.

The random component of time values for a given transaction T that is known to a real system but not included in the model is represented by the factor $v^d_{T,t}$.

$$v^d_{T,t} \sim \mathscr{L}(0, \sigma_t^{v2}) \tag{5.1}$$

This factor $v^d_{T,t}$ is drawn from log-normal distribution with median value of 1. This means that in half of the cases the value is less than predicted by the model and in the other half of the cases it is bigger.

The random component of time values for a given transaction T that is neither known to the real system nor included in the model is represented by the factor $e^d_{T,t}$, which is also drawn from a log-normal distribution with median value of 1:

$$e^d_{T,t} \sim \mathscr{L}(0, \sigma_t^{d2}) \tag{5.2}$$

The deviation parameters σ_t^d and σ_t^v are parameters of the traffic model, which will be important parameters in the simulative performance evaluation.

Example: For a time value t from one of the traffic class models, the value $t \cdot v^d_{T,t}$ is used to construct the utility function signaled, and the value $t \cdot v^d_{T,t} \cdot e^d_{T,t}$ is used to construct the utility function for the evaluation of this transaction.

Variation in Utility Parameters

For the variation terms of utility values, there are also many influencing factors, which combine multiplicatively. The resulting factor that models these variations is therefore also from a log-normal distribution. In this case, the distribution has to have a mean value of 1 so that the resulting utility values are comparable.

A log-normal distributed random variable with mean 1 has the form $p_n \sim \mathscr{L}(-\frac{1}{2}\sigma_p^2, \sigma_p^2)$. The multiplicative combination of multiple of these values is possible in this case, since the product is also log-normally distributed with a mean of 1:

$$\prod_n p_n \sim \mathscr{L}\left(-\frac{1}{2}\sum_n \sigma_n^2, \sum_n \sigma_n^2\right) \tag{5.3}$$

The random component of utility values for a given transaction T that is known to a real system but not included in the model is represented by the factor $v^d_{T,U}$.

$$v^d_{T,U} \sim \mathscr{L}(-\frac{1}{2}\sigma_U^{v\,2}, \sigma_U^{v\,2}) \tag{5.4}$$

The random component of time values for a given transaction T that is neither known to the real system nor included in the model is represented by the factor $e^d_{T,U}$, which is also drawn from a log-normal distribution with median value of 1:

$$e^d_{T,U} \sim \mathscr{L}(-\frac{1}{2}\sigma^{d\,2}_U, \sigma^{d\,2}_U) \tag{5.5}$$

For different utility parameters in the utility functions, different independent random variables are used. For the utility for example

5.2.7 Utility Model for Finish-time Transaction

Utility functions for Finish-time Transactions (FTs) are functions that map a utility value U_T to a possible finish-time of transaction T (f_T) of this transaction T. An example of such a function is given in Figure 5.1.

Such utility functions typically show the shape of an s-curve:

- The utility function is monotonically decreasing, because the earlier the transaction is finished, the better it is for the user.
- For very small times, the user's additional benefit from even smaller finish times is low, because the user cannot distinguish or does not even notice the change.
- After a very long time, the utility value is not decreasing anymore, because the user or the application has canceled this transaction. For example in a web browser, the user has clicked away and opened another web page.

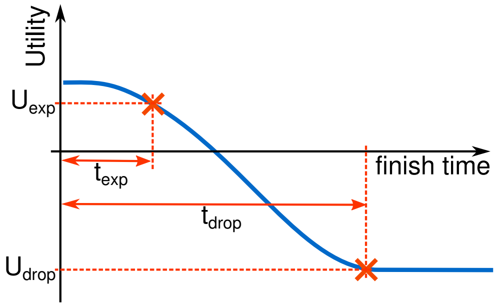


Figure 5.1: General shape of a utility function of FTs.

Two points on this utility function are used to define the scaling and shifting of the s-curve:

- The expected finish-time $t_{T,\text{exp}}$ and the utility value $U_T(t_{T,\text{exp}})$ at this time.
- The drop time $t_{T,\text{drop}}$ and the utility value $U_T(t_{T,\text{drop}})$ at this time.

Both time values and both utility values are different for each transaction. They depend on the application, on the size of each transaction, and on other application or transaction specific parameters. In the following sections, these parameters are explained.

Expected Finish-time

There are transactions, where a user is able to know the size s_T of the transaction T and the data-rate of the access network (R), and therefore knows when he can expect the transaction to be finished. Examples for this are downloads using a web browser. In this case, the expected finish-time (f_T) for a unidirectional transaction can be determined as

$$t_{T,\mathrm{exp}} = s_T / R \tag{5.6}$$

If the transaction contains chunks in both directions, $t_{T,\mathrm{exp}}$ is the earliest point in time, where with the given data-rate for upstream (R_{UL}) and downstream (R_{DL}) all chunks are completed.

In other cases, the expectation depends solely on the type of function the transaction serves in the user's interface. According to [Mil68] and [Nie94], interactive responses should be given to the user in less than 100 ms, navigation results and other responses should be visible earlier than 1 s after an action of the user requested it. In these cases, developers of web pages or applications will already take care that these criteria are not impossible to meet with an access network of average data-rate.

For each of the traffic model components, this has to be configured appropriately.

Drop Time

For each transaction, there is a time $t_{T,\mathrm{drop}}$ where the transaction is dropped or no longer served. Reasons are:

- The user cancels the transaction explicitly, closes the program, clicks away, etc.
- The transaction is canceled because of an application timeout.
- An underlying connection (e.g. transport layer) disconnects.

Whether this drop-time depends on the user's expectation is analogous to the previous section. The utility value does not decrease further after this point in time.

Curve Parameters

The goal is to determine the values of the two points that scale and shift the utility function:

- Expected finish-time $t_{T,\mathrm{exp}}$ and the utility $U_T(t_{T,\mathrm{exp}})$ at this point in time.
- Drop time $t_{T,\mathrm{drop}}$ and the utility $U_T(t_{T,\mathrm{drop}})$ at this time.

As mentioned before, the user's expectation is in some cases depending on the size of the transaction, in some cases it is independent of the size, and in other cases it is a combination of both. The parameters t_c ($t_{c,\text{exp}}$ and $t_{c,\text{drop}}$) determine a constant amount of time for the transaction. The parameters t_p ($t_{p,\text{exp}}$ and $t_{p,\text{drop}}$) determine the size dependent amount of time of the transaction.

Similarly, the user's valuation is in some cases depending on the size of the transaction, in some cases it is independent of the size, and in other cases it is a combination of both. The parameters U_c ($U_{c,\text{exp}}$ and $U_{c,\text{drop}}$) determine a constant utility value. The parameters t_p ($U_{p,\text{exp}}$ and $U_{p,\text{drop}}$) determine the size dependent utility value of the transaction.

The variation of utility values introduced in Section 5.2.6 also needs to be incorporated to determine the two above mentioned points:

$$t_{T,\text{exp}} = \left(t_{c,\text{exp}} + \frac{s_T}{R} t_{p,\text{exp}}\right) v^d_{T,t} \tag{5.7}$$

$$U_T(t_{T,\text{exp}}) = \left(U_{c,\text{exp}} + s_T U_{b,X} U_{p,\text{exp}}\right) v^d_{T,U} \tag{5.8}$$

$$t_{T,\text{drop}} = \left(t_{c,\text{drop}} + \frac{s_T}{R} t_{p,\text{drop}}\right) v^d_{T,t} \tag{5.9}$$

$$U_T(t_{T,\text{drop}}) = \left(U_{c,\text{drop}} + s_T U_{b,X} h_U U_{p,\text{drop}}\right) v^d_{T,U} \tag{5.10}$$

$U_{b,X}$ is the specific utility value for application class model X. The specific utility is the mean utility per byte at the expected finish-time for transactions of this application class model. X is one of the indices listed in Section 5.4. While s_T is the actual size of transaction T, $\overline{s_X}$ is the mean size of transactions from application class model X. For most traffic class models in Section 5.4, this mean transaction size $\overline{s_X}$ is used in the definition of the U_c values, to allow for an easy to understand overall utility for a given monthly data volume.

The parameters for the utility function in Equation 5.7 to 5.10 define the user's expectation and evaluation. Since the DTE, the platform or the application are not able to ask the user for this, they have to estimate it. This estimation is prone to estimation errors and therefore includes those terms that are not known to the system.

$$\begin{aligned}
\widetilde{t_{T,\text{exp}}} &= t_{T,\text{exp}} e^d_{T,t} && (5.11)\\
\widetilde{t_{T,\text{drop}}} &= t_{T,\text{drop}} e^d_{T,t} && (5.12)\\
\widetilde{U_T(t_{T,\text{exp}})} &= U_T(t_{T,\text{exp}}) e^d_{T,U} && (5.13)\\
\widetilde{U_T(t_{T,\text{drop}})} &= U_T(t_{T,\text{drop}}) e^d_{T,U} && (5.14)
\end{aligned}$$

5.2.8 Utility for Streaming Transaction

The *Streaming* application class model defines how the streams are split in chunks, when these chunks are sent, and how the receiver reacts on missing or delayed chunks. The receiver is either able play the stream uninterruptedly or has at least one rebuffering period. A rebuffering period is characterized by the time it occurred (the point in time when the stream stops to play and instead e.g. an hourglass symbol is displayed) and the duration until the stream continues to play back. Playing the stream is therefore characterized by a list of such rebuffering periods.

The utility value for a ST is at maximum the ideal utility, when the stream is played uninterrupted. The minimum utility value of a ST is the utility for a dropped stream that means the stream is not played at all or not finished.

The *Streaming* application class model defines that the receiver plays out each chunk that has been received in time. When one chunk is not received in time, the receiver has to stop playing. The receiver starts to play again, when there are enough chunks received for playing t_B seconds. The playout of all following chunks is then delayed by this rebuffering time.

The utility penalty for number of rebuffering events (U_{b_N}) and utility penalty for duration of rebuffering events (U_{b_t}) define how the utility is reduced depending on the number B_T and total duration b_T of rebuffering events.

$$U_T = U_{T,\text{drop}} + (1 - \min(1, \frac{U_{b_t} b_T + U_{b_N} B_T}{d_T}))(U_{T,\text{ideal}} - U_{T,\text{drop}}) \tag{5.15}$$

The value d_T is the duration of the transaction T. The parameters t_B, $t_{T,\text{exp}}$, U_{b_N}, U_{b_t}, $U_{T,\text{drop}}$, and $U_{T,\text{ideal}}$ are discussed and defined for the *Streaming* application class model.

The utility parameters $U_{T,\text{ideal}}$ and $U_{T,\text{drop}}$ are again a combination of a fixed utility value and a part proportional to the transaction size:

$$U_{T,\text{ideal}} = \left(U_{p,\text{ideal}} \cdot s_T + U_{c,\text{ideal}}\right) v^d_{T,U} \tag{5.16}$$

$$U_{T,\text{drop}} = \left(U_{p,\text{drop}} \cdot s_T + U_{c,\text{drop}}\right) v^d_{T,U} \tag{5.17}$$

The utility values the scheduler uses are multiplied by the factor $e^d_{T,U}$ to represent all influencing factors unknown to the scheduler:

$$\widetilde{U_T(t)} = U_T(t) e^d_{T,U} \tag{5.18}$$

5.2.9 Utility for Real-time Transaction

For RTs, the main fact that influences the user's perception, is which of the traffic chunks can be used to play back to the user. Chunks that arrive too late cannot be used for play back and are discarded. The utility value therefore depends on how often chunks of a transaction do not meet the latency requirement and are therefore lost.

Whether it is worse to loose subsequent chunks or random chunks depends on the application and the way the data is encoded into packets (codec). The utility therefore might also depend on the burstiness or autocorrelation of such loss events.

A metric describing the burstiness of the loss events, the autocorrelation of packet losses of transaction T (Ψ_{lT}) is used to differentiate these cases. Multiple definitions for the autocorrelation of loss events can be found in [NR08, Formula 2]. Another metric is the probability, whether there are two loss events within a time duration d:

$$\Psi_{lT}(d) = \mathbb{E}\left(\min\left(1, \sum_{\tau=t}^{t+d} I_{lT}(t) \cdot I_{lT}(\tau)\right)\right) \tag{5.19}$$

With $I_{lT}(t)$ as the loss indication of a packet at time t of transaction T. $I = 1$ in case of a lost packet, $I = 0$ otherwise.

The probability that subsequent traffic chunks of transaction T are lost is the autocorrelation value Ψ_T:

$$\Psi_T = \mathbb{E}\left(L_C \cdot L_{C+1}\right) = \frac{1}{C_T} \sum_{C \in T} L_C \cdot L_{C+1} \tag{5.20}$$

The auto covariance coefficient γ_T is limited to the range $(-1, 1)$ with $\gamma_T = 0$ in case the drops occur completely random:

$$\gamma_T = \mathbb{E}\left((L_C - l_T) \cdot (L_{C+1} - l_T)\right) = \frac{1}{C_T} \sum_{C \in T} (L_C - l_T) \cdot (L_{C+1} - l_T) \tag{5.21}$$

L_C indicates whether traffic chunk C is received within its deadline:

$$L_C = \begin{cases} 1 & f_C > t_{C,\text{drop}} \\ 0 & f_C \leq t_{C,\text{drop}} \end{cases} \tag{5.22}$$

l_T is the relative frequency of lost traffic chunks C in transaction T:

$$l_T = \frac{1}{C_T} \sum_{C \in T} L_C \tag{5.23}$$

The utility function of a RT in this model is defined as

$$U_{\text{RT}} = U_{T,\text{drop}} + \left(U_{T,\text{ideal}} - U_{T,\text{drop}}\right) \cdot {l_T}^{c_L + \gamma_T \cdot c_v} \tag{5.24}$$

The utility value cannot become larger than the ideal utility $U_{T,\text{ideal}}$ and it cannot become smaller than the utility for completely dropping the transaction $U_{T,\text{drop}}$.

The parameter c_L controls how fast the utility value drops with increasing number of lost chunks.

- When $c_L = 1$ the utility value decreases linearly with the number of lost chunks.
- Usually the value for the user is dropping rapidly when there are only a few chunks missing. This case is modeled by $c_L > 1$.
- The unrealistic case where there is already a significant value for the user when only a few chunks arrived, could be modeled with $c_L < 1$.

Whether it is better when lost chunks are sequential or randomly distributed over the whole transaction is modeled with the parameter c_v.

- With $c_v = 0$ the distribution of the lost chunks is not relevant.
- With $c_v > 0$ it is beneficial when lost chunks are distributed randomly. In this case, short missing sequences can be reconstructed using Forward Error Correction (FEC) or can be approximated from previous data using filters.
- With $c_v < 0$ it is beneficial when the lost chunks occur sequentially in bursts. This is the case when the data in one traffic chunk depend on previous data, and thus a single missing chunk also invalidates the following chunks.

The ideal utility value $U_{T,\text{ideal}}$ and the worst possible utility value $U_{T,\text{drop}}$ consist of a term depending on the transaction's size s_T and a constant term:

$$U_{T,\text{ideal}} = \left(U_{p,\text{ideal}} \cdot s_T + U_{c,\text{ideal}}\right) v^d_{T,U} \tag{5.25}$$

$$U_{T,\text{drop}} = \left(U_{p,\text{drop}} \cdot s_T + U_{c,\text{drop}}\right) v^d_{T,U} \tag{5.26}$$

Again, the utility values the scheduler uses are multiplied by the factor $e^d_{T,U}$ to represent all influencing factors unknown to the scheduler:

$$\widetilde{U_T(t)} = U_T(t) e^d_{T,U} \tag{5.27}$$

5.3 Application Model for Internet Traffic

In this section, a new traffic model for Internet access traffic is defined, which is suitable to evaluate QoT. This model copes with characteristics of Internet access traffic and its future trends by forming a traffic mix consisting of several application models with configurable shares that represent a variety of applications and services.

This traffic model has to fulfill several goals. It has to abstract from specific applications as far as possible, to cover a wider range of applications as well as future applications. It should be concrete enough to be able to derive the subscribers QoE. It should model the traffic of different application classes close enough, so that results show how different application classes or traffic types profit or suffer from QoT. It should be configurable enough, to allow to be able to cover different use cases, scenarios, and traffic predictions.

To achieve these goals, the traffic model consists of seven application class models. Each application class model is highly configurable. Eight different model variants specify parameter sets for these application class models, as well as defining the share of each model.

The application class models generate transactions consisting of one or multiple chunks of data, which corresponds to the desired level of abstraction. As mentioned in Section 4.2.1, the segmentation of the chunks is technology dependent. The chunks of this traffic model are therefore technology independent and can be used in a fluid flow network simulation or they can be arbitrarily packetized and processed with any packet level network simulator. The network model may include models of transport layer protocols or even real transport layer protocol implementations such as the Network Simulation Cradle (NSC) [JM06].

Applications, Transactions and Chunks are used as defined in Section 4.2.3.

One goal of this traffic model is to provide metrics that reflect application layer characteristics from a subscriber's perspective. This corresponds to the transactions introduced in Section 2.4. Examples of such metrics are:

- The duration until a transmission is finished.
- The number and duration of buffer underruns in a streaming video.
- The number of dropouts (dropped packets or excessive delay) in a VoIP call and the resulting MOS estimation [ITU11].

Since the traffic model offers subscriber metrics, it is especially useful for the evaluation of QoS and QoE mechanisms in the Internet access.

The model is designed for Internet access with one or multiple subscribers and their mix of applications. It is not designed to evaluate core networks with a high degree of aggregation, e.g. several thousand subscribers. It is further not designed to evaluate data centers. In data centers the traffic origins from a small number of services and a more detailed model of these services is required. To study effects of devices and operating systems, such as caching strategies, a more specific model of the applications is needed.

5.3.1 Other Traffic Models

Traffic modeling is no new subject. However, there is always the need to have a model that represents current traffic and future trends [FK03]. Internet traffic is constantly increasing [Cis12], but forecasts on the progression of Internet traffic and its characteristics show no clear prediction. Internet traffic is extremely heterogeneous. Bitrates, connection sizes, connection durations and delay requirements vary over many orders of magnitude. This is because the traffic origins from services and applications with extremely different characteristics.

Traffic modeling for dimensioning circuit switches telephone networks was done decades ago [Wil41]. In packet switched networks telephony plays a minor role. Measurements and analysis of Internet traffic in [San12a] and [Cis12] show that web traffic, file sharing, real-time entertainment and other services account for large fractions of today's Internet traffic. In the future, traffic of cloud applications and Machine-to-Machine (M2M) use cases [Cis12] will become more important.

Generally accepted models for the entire traffic of Internet access subscribers are rare. There is a traffic model for HTTP and FTP, which has been adopted by the 3GPP [3GP09] and the WiMAX foundation [For07]. However, these models are outdated as more recent measurements [Cha10] yield significantly different values for object sizes and data-rates.

In the last years there have been several attempts to measure real network traffic. These measurements can be separated into two classes:

1. Measurements of aggregated traffic of several subscribers. For example measurements in the campus network of the University of North Carolina at Chapel Hill (UNC) [WAHC$^+$06], in a large German access network [MFPA09], and in several wide area networks [OPTW07]. All these measurements do not allow modeling the behavior of independent subscribers and do not provide the required metrics from a subscriber's perspective.
2. Measurements of single applications or services.

In contrast to measurements of aggregated traffic, application layer measurements provide more detailed views into different application types, so that the behavior of each application type can be studied.

There have been many attempts to design traffic models from application layer measurements. But these models mainly focus on only one application. There are only a few models, which cover a whole traffic mix generated by multiple applications at the same time. For Universal Mobile Telecommunications System (UMTS) networks, two older traffic mix models exist ([KLL01] and [AVP$^+$03]). But none of the available models generates a mix on a per-subscriber level. And none of the available models can be used to formulate metrics from a user's perspective.

5.3.2 Steady State Model

The traffic is usually fluctuating over the course of a day and between weekdays, weekend and holidays, documented e.g. in [TMW97]. For the performance evaluation of a computer network, it is, however, desirable to examine a steady state. In the results of non-stationary simulations, some of the values cannot be regarded to originate from the same statistical population, when their distribution is depending on the mean load of the network. This violates the precondition for many statistical analysis methods.

Therefore this traffic model generates stationary load, which should be adjusted to the corresponding busy hours. The evaluation metrics in Section 4.4 take this into consideration. How to deal with such busy-hour simulations is explained there.

5.3.3 Model Variants

As mentioned above, there is no generally accepted prediction for future Internet traffic. Furthermore, the usage patterns vary among different types of networks, such as university campuses or residential areas. It varies with different legislation, for example regarding the criminalization of P2P file sharing. And it varies depending on the operator's business model, for example with the size of the free data volume and the procedure for excess traffic.

One goal of this traffic model is to represent many different possible future developments and trends. Therefore the traffic model defines eight variants, each with a significant bias to a basic variant and a name describing the tendency of the corresponding deviation. These variants allow evaluating systems under different scenarios. More important, they allow to state in which scenario a system under evaluation has its strengths and weaknesses.

There is one basic variant, which defines the share of the traffic of the application class models in the traffic mix as well as configuration profiles for each model. Starting from this basic variant, there are three variants which define a different share of the traffic of the application class components in the traffic mix. Four additional variants define different configuration profiles for the models, but leave the traffic shares unmodified. A short description of all variants can be found in Table 5.1. The configuration profiles for the corresponding application class models are listed in Tables 5.2 to 5.14.

5.3.4 Signaled Size

In Section 5.2 degraded knowledge for time values and utility values is introduced. In this section, degraded knowledge of size values is introduced.

In the previous sections, the various size variables are introduced. The variation of these size variables is defined in the traffic class models.

Because size values are part in the QoT signaling, these values can be partially erroneous, because the knowledge concerning size values is degraded in some cases.

For the same reasons as with size and utility value degradation, the knowledge error is also assumed to be log-normally distributed. Like time values, the distribution has to have a median value of 1, so that in half of the cases the estimation id larger than the real size and in half of

Table 5.1: Description of all model variants

Variant	Description
Basic	Traffic mix as it can be seen in today’s Internet
BigObjects	Same share as in the Basic variant, but all objects are bigger
SmallObjects	Same share as in the Basic variant, but all objects are smaller
BigVariance	Same share as in the Basic variant, but the variation of object sizes is bigger
SmallVariance	Same share as in the Basic variant, but the variation of object sizes is smaller
MoreBackground	*Download*, *Background* and *P2P* application class models have a bigger share
MoreUrgent	*Real-time*, *Interactive* and *Streaming* application class models have a bigger share
MoreStreaming	*Streaming* application class model has a bigger share

the estimations the real size is smaller than the estimation.

This factor $v^d_{T,t}$ is drawn from log-normal distribution with median value of 1. This means that in half of the cases the value is less than predicted by the model and in the other half of the cases it is bigger.

The size error of transaction T is a random value that is neither known to the real system nor included in the model. It is represented by the factor $e^d_{T,s}$, which is drawn from a log-normal distribution with median value of 1:

$$e^d_{T,s} \sim \mathscr{L}(0, \sigma^{d2}_s) \tag{5.28}$$

The deviation parameters σ^d_s is a parameter of the simulation, its value is chosen depending on the type of evaluation.

5.3.5 Transactions

In most of the application class models, new transactions are generated according to a Poisson Process. There are multiple reasons or advantages for this approach:

- Using a Poisson process, it is impossible for any component to predict, when the next transaction will arrive. If transactions were predictable, schedulers could use this to improve the performance. Such a gain from predicting future user behavior is not the goal of this work. It is therefore beneficial to avoid predicting traffic by using Poisson processes.
- Poisson processes are well understood and often used for modeling the arrival of any type

of interactive session, e.g. phone calls.

- A Poisson process is the result from the superposition of any process, when this superposition consist of a large number of independent sources. (Palm–Khintchine theorem) Even if the underlying processes for the traffic class models are not Poisson processes, the resulting process can be approximated with a Poisson process, because there are a large number of sources: Today, many broadband subscribers use multiple devices. For example one computer, one smartphone and some appliances such as TVs. On each device several applications are installed and are able to run at the same time and each application can start new transactions independently of other applications. Some applications can be viewed as multiple sources, as they can do multiple transactions in parallel.

5.3.6 Application Classes

The goal is to achieve both: modeling the main characteristics of current Internet traffic exactly, and abstracting from specific applications to also fit for future applications and protocols. Therefore the specific applications are abstracted to application classes. For each application class, the main characteristics are modeled.

These models are not exact copies of existing applications, but represent common properties of different applications. Thus the models are also suitable to cover future applications, with the drawback that implementation characteristics of today's applications, e.g. discretization of recent video codecs, are not represented. Unless noted otherwise, all sizes and characteristics are extrapolated to the values for the year 2013.

All models except the *P2P* component follow a stochastic renewal process for the arrivals of new transactions. Therefore, the amount of traffic, these application class models create, is independent of the available network capacity. Practically, this means that the applications do not adapt to the available bitrate on a longer time scale.

Each application class model can be configured, to generate a mean bitrate $R_{X,d}$ in downlink and $R_{X,u}$ in uplink direction per subscriber. Configuration parameters of the components that define the distribution of sizes of transmissions are different for each of the model variants.

5.4 Application Class Models

This section describes the models of the above mentioned application classes in more detail. The following eight application class models are defined:

- *R*: Real-time
- *I*: Interactive
- *S*: Streaming

- W: Web
- P: Peer-to-Peer file sharing
- D: Download
- U: Upload
- B: Background

5.4.1 Real-time

This application class — together with the *Interactive* application class — models applications with real-time behavior. The *Real-time* application class represents traffic that consists of transactions with CBR characteristic. This application class covers for example VoIP telephony, real-time video and sensor data in M2M applications.

The transactions are unidirectional and arrive according to a Poisson process with arrival rate λ_{Rs} (see Table 5.2). The transaction durations are negative exponentially distributed with a mean of $1/\lambda_{Rd}$. Each transaction consists of chunks, e.g. video frames, with a deterministic chunk-rate chosen uniformly from the set M_{Rc} at the sender. The sizes of the chunks during one transaction have a constant size, but for each transaction a new size is determined according to a normal distribution $\mathcal{N}(\mu_{Rl}, \sigma_{Rl}^2)$, truncated to a minimum of $n_{Rl,\min}$.

If the generated bitrate exceeds the bitrate that can be handled by the network, the whole transaction is dropped at once. Figure 5.2 shows the behavior of this model.

Table 5.2: Parameters for *Real-time* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
λ_{Rs}	$(R_R\lambda_{Rd}\lvert M_{Rc}\rvert)/(\mu_{Rl}\sum M_{Rc})$				
λ_{Rd}	$1/300$ s				
M_{Rc}	$\{100, 60, 50, 30, 25, 24, 20, 15, 12.5, 10, 8, 5, 1\}$ Hz				
μ_{Rl}	5000 B	20000 B	1000 B	5000 B	
σ_{Rl}	2500 B	10000 B	500 B	10000 B	500 B
$n_{Rl,\min}$	1 B				

The general utility model for RTs is explained in Section 5.2.9. The assumption for the *Real-time* traffic class is that already a few lost packets significantly reduce the user's valuation for a connection. Therefore the parameter c_L is chosen larger than 1. Parameter c_v is set to 0, because there is no clear trend, whether such connections are more affected when the losses occur in bursts or the losses occur randomly distributed. The deadline for each chunk is a fixed offset to the point in time the sender sends out this chunk. Depending on the utility model variant, the absolute value a user would assign to the transaction depends on the total size of the transaction and on fixed values per transaction. The total size of a transaction depends on the

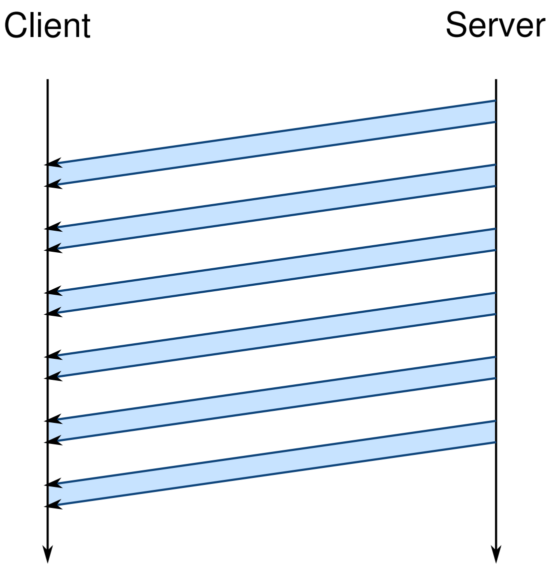


Figure 5.2: Data flow for *Real-time* application class model.

duration of the connection and the bitrate of the connection.

Unless noted otherwise, the following parameters are used for the evaluations in Chapter 6:

Table 5.3: Parameters for utility functions of *Real-time* application class model

Parameter	$U_{b,R}$	$U_{p,\text{ideal}}$	$U_{c,\text{ideal}}$	$U_{p,\text{drop}}$	$U_{c,\text{drop}}$	c_v	c_L	$t_{c,\text{drop}}$
Basic	U_b	1.0	1.0	-1.0	-1.0	0	4	$t_{C,0}$ + 100 ms
Proportional	U_b	2.0	0.0	-2.0	0	0	4	$t_{C,0}$ + 100 ms
Fixed	U_b	0	2.0	0	-2.0	0	4	$t_{C,0}$ + 100 ms

It can be seen, that depending on the utility model variant introduced in Section 5.2.3 (*Basic*, *Proportional*, and *Fixed*), the influence of the transaction size is different.

5.4.2 Interactive

This traffic model also has real-time requirements, but contrary to the *Real-time* application class model, this model generates bidirectional traffic and may have a varying bitrate, see Figure 5.3. This kind of traffic is generated by a wide variety of applications, for example by online games, interactive cloud applications, chats or M2M communication.

The model consists of transactions that are generated by a Poisson process, with an arrival rate of λ_{Is} (see Table 5.4). In every single transaction there are packets in both directions. The packets are generated by two independent Poisson processes. The arrival rate for packets in downlink direction is $\lambda_{Ip,\text{down}}$, in uplink direction it is $\lambda_{Ip,\text{up}}$. The size of packets is determined by two binomial distributions $B(n_{I,\text{down}}, p_{I,\text{down}})$ and $B(n_{I,\text{down}}, p_{I,\text{up}})$.

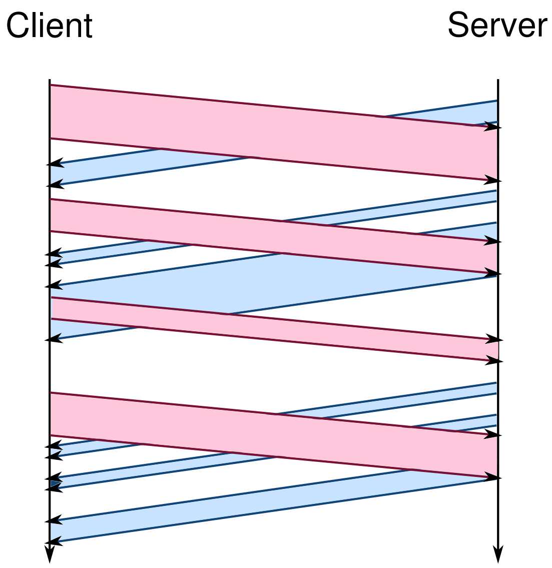


Figure 5.3: Data flow for *Interactive* application class model.

Table 5.4: Parameters for *Interactive* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
λ_{Id}	1 / 300 s				
λ_{Is}	$\sqrt{(R_{I,d}R_{I,u})/1\ \text{Mbit/s}}/600$ s				
$\lambda_{Ip,\text{down}}$	$R_{I,d}\lambda_{I,\text{down}}\lambda_{Id}/\lambda_{Is}$				
$\lambda_{Ip,\text{up}}$	$R_{I,u}\lambda_{I,\text{up}}\lambda_{Id}/\lambda_{Is}$				
$\lambda_{I,\text{down}}$	640 byte	3200 byte	128 byte	640 byte	
$\lambda_{I,\text{up}}$	320 byte	1600 byte	64 byte	320 byte	

The *Interactive* traffic class model uses RTs. Therefore, the utility model is similar to the utility model of the *Real-time* traffic class. The general utility model for RTs is explained in Section 5.2.9. While the deadline $t_{c,\text{drop}}$ of traffic chunks for the *Interactive* traffic class model is shorter, the parameters c_L and c_v are chosen equally to the *Real-time* traffic class explained in Section 5.4.1.

The variation of the size of transaction of the *Interactive* traffic class is larger than for the *Real-time* traffic class, as it covers a wider range of possible applications and protocols. The maximum achievable utility of an *Interactive* transaction is therefore always proportional to the size of this transaction.

The following parameters are used for the evaluations in Chapter 6:

Table 5.5: Parameters for utility functions of *Interactive* application class model

Parameter	$U_{b,I}$	$U_{p,\text{ideal}}$	$U_{c,\text{ideal}}$	$U_{p,\text{drop}}$	$U_{c,\text{drop}}$	c_v	c_L	$t_{c,\text{drop}}$
All	U_b	1.0	0	0	-1.0	0	4	$t_{C,0}$ + 50 ms

5.4.3 Streaming

Streaming applications and *Real-time* applications have a very similar behavior. The difference between them is that for the data stream of *Streaming* applications, a reliable transport is used. This introduces additional delay. Therefore these applications typically use a larger buffer of multiple seconds or more. This application class covers mainly non-life video traffic such as VoD services and popular video sharing services such as YouTube.

The transactions of this application class are parameterized similar to those of the *Real-time* application class. The bitrate and size distributions are chosen to be a compromise between different available measurements of YouTube traffic and general video traffic in [ZSGK09, YZZZ06, GALM07, PENUK08].

New unidirectional transactions arrive according to a Poisson process with arrival rate λ_{Ss} (see Figure 5.6). Each transaction has a duration d_T that is negative exponentially distributed with a mean of $1/\lambda_{Sd}$. The bitrate of a transaction is distributed according to a bounded normal distribution $\mathcal{N}(\mu_{Sr}, \sigma^2_{Sr})$, truncated to a minimum of $n_{Sr,\min}$ and a maximum of $n_{Sr,\max}$. To keep the mean bitrate constant, the maximum bitrate is chose as $n_{Sr,\max} = 2\mu_{Sr} - n_{Sr,\min}$.

Each stream is split into chunks of fixed size c_S. The chunks to play out the first 30 seconds of the stream are sent from the sender as fast as possible. The following chunks are sent 20% earlier than necessary, therefore the sender sends with 125% the data-rate of the stream rate. The receiver plays out each chunk, in case it has been received in time. When one chunk is not received in time, the receiver has to stop playing. The receiver starts to play again, when there are enough chunks received for playing t_B seconds. This "rebuffering" time is an important criterion for the user's experience. The playout of all following chunks is delayed by this rebuffering time.

Table 5.6: Parameters for *Streaming* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
λ_{Ss}	$\lambda_{Sd}R_S/\mu_{Sr}$				
λ_{Sd}	$1/300$s				
μ_{Sr}	1 Mbit/s	2 Mbit/s	500 kbit/s	1 Mbit/s	
σ_{Rr}	500 kbit/s	1 Mbit/s	250 kbit/s	1 Mbit/s	200 kbit/s
$n_{Sr,\min}$	1 kbit/s				
c_S	64 Kbyte				

The general utility model for STs is explained in Section 5.2.8.

The ideal utility $U_{T,\text{ideal}}$ is fully depending on the transaction size, while the worst possible utility $U_{T,\text{drop}}$ is partially a constant value and partially dependent on the size and duration. The reason for these settings is that it does not make a difference for the user's valuation of the network quality, whether a long video is offered as a single long video stream, or whether it is separated into multiple smaller segments. There is however a difference for the utility concerning the worst possible user experience: long interruptions of each and every segment are more annoying than one interruption. The equivalent to the single long video stream having the worst utility would be for this video disaggregated into several segments that some of the segments have the worst utility, while some of the segments have acceptable experience. This is because the long video stream with the worst possible utility still has some phases where it played OK.

Table 5.7 list the parameters used for the evaluations in Chapter 6.

Table 5.7: Parameters for utility functions of *Streaming* application class model

Parameter	$U_{b,S}$	$U_{p,\text{ideal}}$	$U_{c,\text{ideal}}$	$U_{p,\text{drop}}$	$U_{c,\text{drop}}$	t_B	U_{b_N}	U_{b_t}
Real-time	$2U_b$	1.0	0	-0.3	-5.0	0.5 s	$4h_U$	$2h_U\frac{1}{s}$

5.4.4 Web

Web-alike traffic accounts for a large portion of the traffic mix. There are several measurements and models of web traffic [3GP09], [EGS11] that show significantly different values, and there are publications that show how web traffic changed over time [CAP10] and with new technologies such as Asynchronous JavaScript and XML (AJAX) [SAAF08]. Some years ago, there was also the obvious performance limitation introduced by the overhead for opening TCP connections [PM95]. This problem is addressed today by using HTTP/1.1 [FGM$^+$99], concatenating elements and separating them with JavaScript on the client web browser, and there are standard proposals to allow intelligent servers to send elements to the client without waiting for a request [Spd]. Therefore, the model combines all elements of an entire website to one combined request and one combined result (cf. Figure 5.4).

The parameterization of each request/response is based on the values found in [NMM98], [THB06] and [DB99]. These measurements found out that file-sizes are distributed according to a log-normal distribution. This model however omits transfers of large files, because these will be modeled with the *Download*, *Upload* and *Background* application class models.

New transactions, which consist of request and response, are generated with an Inter-Arrival Time (IAT) according to a log-normal distribution $\ln\mathcal{N}(\mu_{Ws}, \sigma_{Ws}^2)$ (see Table 5.8). Each re-

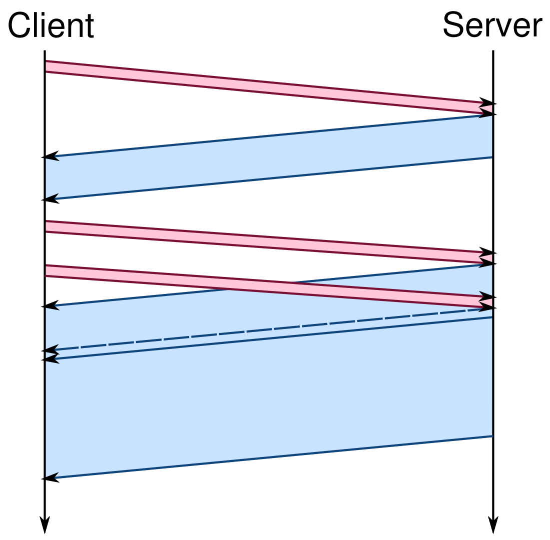


Figure 5.4: Data flow for *Web* application class model

quest has a geometrically distributed size with mean $1/p_W$. The sizes of the responses are distributed according to a log-normal distribution with an offset $\ln \mathcal{N}(\mu_{W,\text{down}}, \sigma^2_{W,\text{down}}) + o_{W,\text{down}}$.

Table 5.8: Parameters for *Web* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
$1/p_W$	1 kB				
$e^{\mu_{W,\text{down}}}$	20 kB	100 kB	4 kB	20 kB	20 kB
$\sigma_{W,\text{down}}$	2.5			3.0	2.0
$o_{W,\text{down}}$	250				
μ_{Ws}	$\ln(e^{\mu_{W,\text{down}}+\frac{1}{2}\sigma^2_{W,\text{down}}} + o_{W,\text{down}}) - \frac{1}{2}\sigma^2_{Ws} - \ln R_W$				
σ_{Ws}	2				

The *Web* application class is not limited to web surfing using web browsers. Many protocols show web-alike traffic with comparable sizes. For example protocols for handling emails and Internet news. Furthermore, many modern applications — especially on smartphones — produce similar traffic [MSF10]. This application class also comprises many cloud applications that perform request/response exchanges between clients and servers.

Utility of Web component

Web transactions are typically requested and viewed inside a web browser. Interactive requests in web browsers are modeled using the *Interactive* traffic class model. Therefore the utility model assumes that the user is not expecting any *Web* transaction to finish earlier than 100 ms after it started. In addition to the 100 ms, the expected finish-time is the time the network needs to transmit these transactions using 100% of the bitrate the user expects. Thus, the user is not expecting the network to finish the transactions to finish earlier than it is theoretically possible.

After some time (at least 30 s), the user clicks away. In addition, this time is longer for larger transactions. The time it takes to finish these transactions using 50% of the expected bitrate (time factor 2.0).

Table 5.9: Parameters for utility functions of *Web* application class model

Parameter	$U_{b,W}$	$t_{c,\text{exp}}$	$t_{p,\text{exp}}$	$U_{c,\text{exp}}$	$U_{p,\text{exp}}$	$t_{c,\text{drop}}$	$t_{p,\text{drop}}$	$U_{c,\text{drop}}$	$U_{p,\text{drop}}$
Basic	U_b	100 ms	2.0	$\frac{1}{4}u$	3/4	30 s	20	$-3u$	-2.0
Fixed	U_b	100 ms	2.0	$\frac{1}{2}u$	1/2	30 s	20	$-5u$	0
Proportional	U_b	100 ms	2.0	0	1	30 s	20	0	-5.0

In Table 5.9 the parameters for the utility function for *Web* transactions are listed. The shortcut u stands for the average proportional utility share: $u = \overline{s_{T,W}} U_{b,W}$.

The utility for a *Web* transaction T finishing at its expected finish-time is in average equal to the overall specific utility $U_{b,W}$ scaled with its size $s_{T,W}$. Thus in all cases $U_{c,\text{exp}}/u + U_{p,\text{exp}} = 1$.

5.4.5 Download and Upload

The behavior of the *Download*, *Upload* and the *Background* application class models are very similar to the *Web* application class model. All transactions consist of one request and one response. Also, they share the same type of distributions for chunk sizes. The difference is the size of the transmitted data and the frequency of these transmissions. New transactions are generated using a Poisson process with an arrival rate λ_{Us} resp. λ_{Ds}.

These application model classes are used to simulate traditional downloads or sharing of music, photos or videos via one-click hosting services, for example.

The *Download* application class model models the download of a larger amount of data than the *Web* application class. The subscriber is not necessarily waiting for the finishing of the transfer. Therefore it uses a log-normal distribution $\ln \mathcal{N}(\mu_{D,\text{down}}, \sigma^2_{D,\text{down}})$ with a larger mean to model the file-sizes. The size of the requests are drawn from a geometric distribution with mean $1/p_D$. The values of the distributions can be found in Table 5.10.

The *Upload* application class model is an exact copy of the *Download* model, with the exception that uplink and downlink directions are exchanged. The requests in the downlink direction are drawn from a geometric distribution with mean $1/p_U$. The upload file-sizes are from the log-normal distribution $\ln \mathcal{N}(\mu_{U,\text{up}}, \sigma^2_{U,\text{up}})$. The values can also be found in Table 5.10.

Table 5.10: Parameters for *Download* and *Upload* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
$1/p_D = 1/p_U$	1 kB				
$e^{\mu}_{D,\text{down}} = e^{\mu}_{U,\text{up}}$	7 MB	20 MB	2 MB	7 MB	7 MB
$\sigma_{D\text{down}} = \sigma_{U\text{up}}$	2.1	2.1	2.1	2.5	1.0
λ_{Ds}	$R_D/e^{\left(\mu_{D,\text{down}}+\frac{\sigma^2_{D\text{down}}}{2}\right)}$				
λ_{Us}	$R_U/e^{\left(\mu_{U,\text{up}}+\frac{\sigma^2_{U\text{up}}}{2}\right)}$				

Utility Download and Upload

Uploads and downloads are less demanding than *Web* transactions, because they usually do not directly delay the user's work flow. However, the user directly experiences when an *Upload* and *Download* transactions is finished.

Table 5.11: Parameters for utility functions of *Upload* and *Download* application class model

Parameter	$U_{b,D} = U_{b,U}$	$t_{c,\text{exp}}$	$t_{p,\text{exp}}$	$U_{c,\text{exp}}$	$U_{p,\text{exp}}$	$t_{c,\text{drop}}$	$t_{p,\text{drop}}$	$U_{c,\text{drop}}$	$U_{p,\text{drop}}$
Basic	$0.3U_b$	30 s	2	$\frac{3}{10}u$	7/10	60 s	10	$-u$	-1
Fixed	$0.3U_b$	30 s	2	$\frac{3}{10}u$	7/10	60 s	10	$-2u$	0
Proportional	$0.3U_b$	30 s	2	0	1	60 s	10	0	-2

5.4.6 Background

This application class model represents applications that generate traffic independently of any subscriber interaction. Automated traffic is generated by many applications, like operating system or application updates, online backups or cloud synchronization. Since these transfers are relatively seldom but may produce much traffic, background transfers are separated from the *Download* and *Upload* application class models.

Background transactions are generated following a Poisson process with arrival rate $\lambda_{B,s,\text{up}}$ and $\lambda_{B,s,\text{down}}$ for uplink and downlink transactions. Note that $\lambda_{B,s,\text{up}}$ and $\lambda_{B,s,\text{down}}$ might be different, because the requested bitrate R_B might be different for uplink and downlink. Request sizes are geometrically distributed with mean $1/p_B$. The size of the response is from log-normal distributions; $\ln\mathcal{N}(\mu_{B,\text{down}}, \sigma^2_{B,\text{down}})$ for downlink transactions and $\ln\mathcal{N}(\mu_{B,\text{up}}, \sigma^2_{B,\text{up}})$ for uplink transactions. The parameterization can be found in Table 5.12.

Table 5.12: Parameters for *Background* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
$1/p_B$	1 kB				
$e^{\mu}_{B,\text{up}} = e^{\mu}_{B,\text{down}}$	14 MB	40 MB	4 MB	14 MB	14 MB
$\sigma_{B,\text{up}} = \sigma_{B,\text{down}}$	2.1	2.1	2.1	2.5	1.0
$\lambda_{Bs,\text{down}}$	$R_B/e^{\left(\mu_{B,\text{down}}+\frac{\sigma^2_{B,\text{down}}}{2}\right)}$				
$\lambda_{Bs,\text{up}}$	$R_B/e^{\left(\mu_{B,\text{up}}+\frac{\sigma^2_{B,\text{up}}}{2}\right)}$				

Utility Background Component

Background traffic has extremely low demands or requirements, because the user's experience is not directly influenced by these transactions. In average size of *Background* transactions is even larger than for *Upload* or *Download* transactions. The user's valuation is lower, however it is still important to the user, that these transactions are not dropped, because it would otherwise affect the computers security (software updates) or the data safety (data backup).

Table 5.13: Parameters for utility functions of *Background* application class model

Parameter	$U_{b,B}$	$t_{c,\text{exp}}$	$t_{p,\text{exp}}$	$U_{c,\text{exp}}$	$U_{p,\text{exp}}$	$t_{c,\text{drop}}$	$t_{p,\text{drop}}$	$U_{c,\text{drop}}$	$U_{p,\text{drop}}$
Basic	$0.15U_b$	60 s	5	$\frac{3}{10}u$	7/10	600 s	10	$-\frac{1}{2}u$	-1/2
Fixed	$0.15U_b$	60 s	5	$\frac{3}{10}u$	7/10	600 s	10	$-u$	0
Proportional	$0.15U_b$	60 s	5	0	1	600 s	10	0	-1

5.4.7 Peer-to-Peer (P2P)

According to [San12b] Peer-to-peer files-sharing accounts for about one third of the traffic today. It is therefore an important part for a general model for a realistic Internet-access traffic mix. This application class represents P2P file sharing applications and similar applications that transfer large amounts of data between subscribers. Models for P2P file sharing applications have been presented in [GCX$^+$05, HLD$^+$05, EIP06].

The *P2P* application class model is a modified On-Off model, to reflect the currently common usage pattern that a P2P file sharing application is only used by some subscribers, and only in a part of the time.

The duration of the Off-phase is negative exponentially distributed with a mean duration $1/\lambda_{Po}$. At the beginning of an On-phase, the application issues a couple of search requests. The number of search requests is distributed shifted geometrically with mean μ_{Pm}. Each search is structured equally to a *Web* request and response. The request size is geometrically distributed with mean $1/p_P$. The size of the search result is distributed shifted geometrically with a mean of μ_{Pr}. After the search requests are finished, a couple of downloads are started. The number of downloads is shifted geometrically distributed with a mean of μ_{Pn}. Each download consists of a request of a geometrically distributed size with mean $1/p_P$ and the actual download, with size from a log-normal distribution $\ln\mathcal{N}(\mu_{Pd},\sigma_{Pd}^2)$.

During the On-phase, upload requests arrive as a Poisson process with arrival rate λ_{Pu}. Each request has a geometrically distributed size with mean $1/p_P$. The size of the upload itself is distributed log-normally $\ln\mathcal{N}(\mu_{Pu},\sigma_{Pu}^2)$. If the number of maximum parallel uploads is not yet reached, upload requests are answered with the requested upload. The maximum number of parallel uploads is binomially distributed with $\mathrm{B}(n_{Pu},p_{Pu})$.

There are several measurements of file sharing download sizes in literature. All of them show that the sizes follow a long-tailed distribution. Measurements results in [Tut04] are interpreted with log-normal distributions, in [BMM$^+$08] with Pareto and Weibull distributions. Plots in [GDS$^+$03] also show long-tailed behavior, and they show significant peaks at the typical file sizes of that time: a peak between 2 MB and 5 MB is probably caused by MP3 music tracks while the peak around 700 MB is obviously caused by CD-image files.

The file size of downloads is modeled with a log-normal distribution with parameters extrapolated from the values from above mentioned literature. Uploaded pieces are significantly smaller than downloads, because many P2P applications split a file into several pieces. These pieces can be retrieved from different hosts in parallel. For the data volume on the access network, it is however irrelevant, that a download is retrieved from several sources. Therefore, the characteristics of upload and download are different.

The On-phase ends depending on the total amount of data that is requested. From the expected bitrate for this application class model $R_{P,\text{down}}$ an expected duration is calculated and multiplied with an overtime factor. The overtime factor is normally distributed as $\mathcal{N}(\mu_{Pt},(\mu_{Pt}/2)^2)$, bounded to $(0,2\mu_{Pt})$. After this duration, all unfinished downloads are canceled and the application model enters the Off-phase again.

In contrast to the other six application class models, the amount of traffic, that is generated by the *P2P* model therefore depends on the available bitrate and network state. The *P2P* model will transmit more data, when there is more bitrate available. To define the amount of data, this application class model is generating, correction parameters c_{Pu} and c_{Pd} are necessary. These parameters specify the ratio of traffic in uplink and downlink direction that is transmitted in a given scenario, in comparison to an infinitely fast network.

When there is no QoS system in the simulated network, it has to be assumed that the *P2P* uploads are throttled with a bitrate limit, as it is common in most P2P file sharing applications.

The mean amount of data downloaded in one transaction can be calculated as:

$$v_{Pd} = c_{Pd}\mu_{Pn}e^{\mu_{Pd}+\sigma_{Pd}^2/2} \tag{5.29}$$

The mean transaction duration $m_{P\text{on}}$ is adjusted, so that the total amount of data fits the configured downlink bitrate R_{Pd}:

$$\overline{t_{P\text{on}}} = (v_{Pd}\mu_{Pt} + \mu_{Pr}\mu_{Pm}){R_{Pd}}^{-1} \tag{5.30}$$

$$\lambda_{Po} = \frac{R_{Pd}}{v_{Pd} - \overline{t_{P\text{on}}}R_{Pd}} \tag{5.31}$$

The IAT for upload requests λ_{Pu} is adjusted, so that the configured uplink bitrate R_{Pu} is met:

$$\lambda_{Pu} = (1 + \frac{1}{\overline{t_{P\text{on}}}\lambda_{Po}})\frac{R_{Pu}}{c_{Pu}e^{\mu_{Pu}+\sigma_{Pu}^2/2}} \tag{5.32}$$

Table 5.14: Parameters for *P2P* application class model

Parameter	Basic	BigObj	SmallObj	BigVar	SmallV
μ_{Pm}	5				
$1/p_P$	1 kB				
μ_{Pr}	100 kB	300 kB	30 kB	100 kB	
μ_{Pn}	5	10	3	100	
$e^{\mu_{Pd}}$	10 MB	30 MB	3 MB	10 MB	
σ_{Pd}	2			3	1
$e^{\mu_{Pu}}$	1 MB	3 MB	300 kB	1 MB	
σ_{Pu}	1			2	0.5
n_{Pu}	20				
p_{Pu}	0.5				
μ_{Pt}	2				

Utility for P2P File sharing Component

In general, the traffic generated by Peer-to-peer File sharing applications has very high data volume, but its requirements are not very demanding to the network. For most users, the traffic caused by Peer-to-peer File sharing applications should have lower priority than most other applications. Peer-to-peer File sharing applications try to affect other traffic of the user as little as possible. Since there is usually no way to control QoS parameters in the access network, these applications implement two approaches:

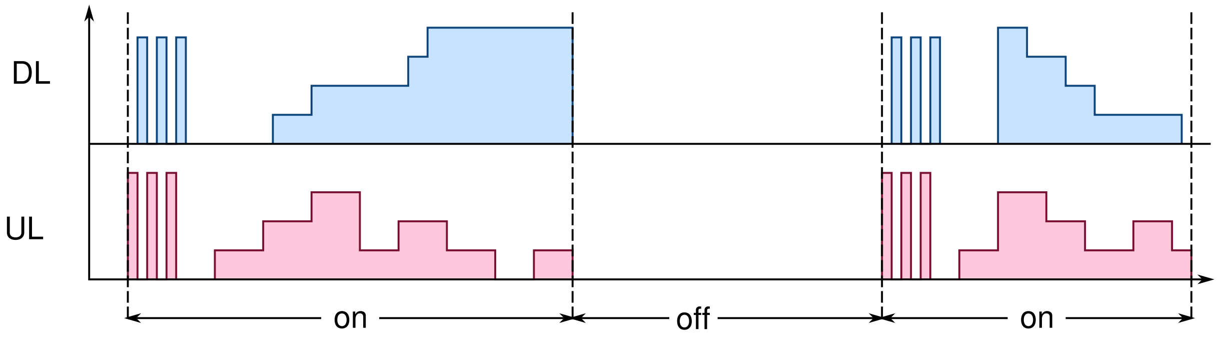
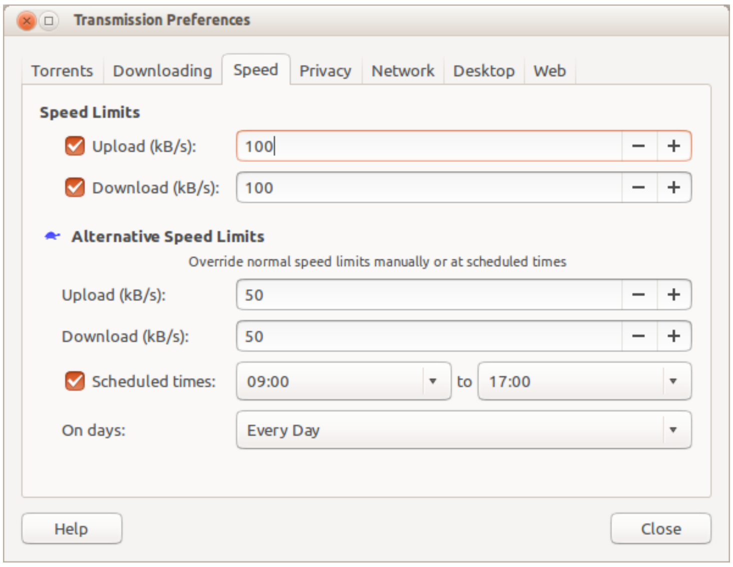


Figure 5.5: Data flow for *Peer-to-Peer* application class model.

1. Some Peer-to-peer applications allow to limit the data-rate for sending and receiving data on application layer. For example, the application *Transmission*[1] as shown in the screenshot Figure 5.6 of its configuration settings.
2. Bittorrent [Coh02] developers developed and implemented LEDBAT, a TCP flavor that tries to utilize only the bitrate that other applications are not trying to use.



Speed Limits

☑ Upload (kB/s): 100

☑ Download (kB/s): 100

Alternative Speed Limits

Override normal speed limits manually or at scheduled times

Upload (kB/s): 50

Download (kB/s): 50

☑ Scheduled times: 09:00 to 17:00

On days: Every Day

Figure 5.6: Screenshot of bitrate limiting settings of *Transmission*.

As shown in Section 4.2.2, for these evaluations, it can to be assumed that the corresponding remote station and the network of the remote station are not delaying or limiting the transactions. In the context of Peer-to-peer File sharing, this assumption obviously does not hold, since the remote station might have an even slower access data-rate than the examined user. However, such applications are usually able to receive from multiple remote stations in parallel. The superposition of the flows is able to saturate the access network of the considered user.

The *P2P* application class model defines three different transaction types:

[1] http://www.transmissionbt.com/

- Search requests, which are similar to *Web* transactions with respect to size and requirements. In many cases, these searches are even performed using a web browser navigating on web pages of special well-known torrent search engines.[2]
- Download requests have a small specific utility and the user does not expect these downloads to saturate his access. Instead he expects them to finish long after that time. The more downloads the user starts, the later the user expects them to finish.
- Upload request have an even smaller specific utility. The utility for dropping an upload transaction is not negative, because the user does not expect any specific upload to finish. Instead, the user interest is the number of finished upload transactions. For example to improve his ranking in the P2P network.

Table 5.15: Parameters for utility functions of *P2P* application class model

Parameter	$U_{b,P}$	$t_{c,\text{exp}}$	$t_{p,\text{exp}}$	$U_{c,\text{exp}}$	$U_{p,\text{exp}}$	$t_{c,\text{drop}}$	$t_{p,\text{drop}}$	$U_{c,\text{drop}}$	$U_{p,\text{drop}}$
UL	$0.1U_b$	120 s	10	0	$\frac{1}{5}$	600 s	30	0	0
DL	$0.1U_b$	60 s	10	0	1	600 s	30	$-\frac{1}{2}u$	0
Search	$0.1U_b$	3 s	1.5	0	3	60 s	10	$-\frac{1}{1000}$	0

5.5 Parameter Sets

5.5.1 Uplink and Downlink

In most of the scenarios in Chapter 6 the available downlink bitrate is the main bottleneck, while there available uplink bitrate is chosen so it is sufficient for the traffic requirements.

This corresponds to the current situation of broadband Internet access, where the uplink bitrate is usually smaller than the downlink bitrate, but the downlink bitrate is still the limiting bottleneck.

As long as the headend is performing the resource allocation on the bottleneck link of the access network in downlink and uplink direction, the QoT system can also be used in the same way when the uplink is the bottleneck.

When both uplink and downlink direction are limiting factors at the same time, QoT even has an advantage, as its scheduling algorithm can be made so it optimize both directions at the same time. Because the decision for uplink and downlink is at the same point, opposing decisions can be avoided.

Cases where both directions are limiting the performance equally are however uncommon, because most of the time one of the directions will clearly be the bottleneck and stay the bottle-

[2] For example *isoHunt*, *MiniNova* and *The Pirate Bay*

neck for a long time. Even when modeling a scenario where both directions are limiting the performance equally, the point where both directions are limiting is unstable. In this case, the simulation typically tilts into one direction being the bottleneck and stays this way for remaining simulation run.

Therefore the evaluations are performed on scenarios where the downlink is the bottleneck.

5.5.2 General Model Parameters

Unless otherwise noted otherwise, the parameters used for the evaluations in Chapter 6 use parameters from the following tables.

Simulated Network

The simulation model is set up as explained in Section 4.2. Parameters for the network model are listed in Table 5.16. The Transaction Manager (TM) (see Section 4.2.4) is a Signaled TM, chained with a Delayed TM and a Degraded TM.

Table 5.16: Parameters for the simulated network

Parameter	Value	Description
Users	10	Number of customers sharing the bitrate of the access network
Number of servers	3	Users connect to one of the servers randomly
Backhaul bitrate	300 Mbit/s	Available bitrate between the access network and a server
Backhaul scheduler	RR	All schedulers, not on the access network
Δt_{Core}	10 ms	Additional, fixed delay for packets on the backhaul links
Δt_{DL}	1 ms	Additional, fixed delay on the access network
Δt_{UL}	1 ms	
R_{DL}	15 Mbit/s	Available bitrate on the access network, shared by all users
R_{UL}	15 Mbit/s	
Signaling delay	0 ms	Parameter of the Delayed TM

The number of subscribers on one access network is set to 10 for most simulations. This matches the order of magnitude of subscribers sharing the resources in existing access networks, such as PON, Internet over CaTV, WLAN and radio access. Cisco suggest an upper bound of 100 active customers per CMTS in [CS06]. And in [HT04] the capacity of 802.11b WLAN is determined to around 10 to 15 parallel VoIP calls for moderate Bit Error Rate (BER) and delays. This implies a limit to a similar number of users. Because all simulated users are permanently active, the number of users in the simulated network corresponds to the number of subscribers in a network that are actively using the network at the same time. A number of 10 active users is also sensible for evaluating the considered schedulers. With a lesser number of customers, the degrees of freedom in the scheduling decisions are limited, with a larger number of customers, the multiplexing gain becomes stronger and the gain from scheduler strategies becomes less relevant.

The bottleneck for all connections is the access network, because the goal is to evaluate the access network. The available bitrate in the backhaul is large enough, so it never becomes the relevant bottleneck for a longer period of time. Since the structure of the Internet between the access network and the servers is not modeled explicitly, each *server* can be seen as one uplink connection or peering point of the access network provider. Because the simulated network is configured, so that the only bottleneck is the access network, the number of servers does not influence the results.

Both, the backhaul connections and the access network are modeled with a fixed delay, in addition to the queuing delay. In the backhaul, this queuing delay is small because of the large over-provisioning on the available bitrate. In the access network, the queuing delay is one of the main metrics for the evaluation.

In the backhaul, all schedulers are RR schedulers, because this models the rate-equalization of TCP without using a real transport protocol in the simulation model.

Traffic Model Parameters

The traffic model is explained in Section 5.3 and the classes of the traffic model are explained in Section 5.4. Most parameters of the traffic classes are listed under the corresponding traffic class model in Section 5.4. Generic traffic model parameters are listed in Table 5.17.

Unless indicated otherwise, the *Basic* variant of the traffic model is used. The requested bitrate is set per user (1.0 Mbit/s DL and 400 kBit/s UL), so the total load of the access network can be set by determining the number of users, sharing the access network. With 10 subscribers each requesting 1.0 Mbit/s in the downlink direction, the resulting data-rate is 10 Mbit/s. With an available bitrate of 15 Mbit/s, the utilization of the downlink bitrate is $\frac{2}{3}$, which is usually considered too large for networks without QoS mechanisms.

All users sharing the access network have the same traffic characteristics, because with different traffic characteristics in one evaluated scenario, there would be an additional dimension in the evaluation metrics. This means, that in addition to the metrics of the overall results (for example the overall utility), it would be necessary to evaluate each metric depending on the corresponding difference between these users. Example: there is an overall performance gain in a scenario with different requested bitrates for each customer. It would be necessary to evaluate whether all or only some users benefit from this performance gain. It might for example be, that the general gain comes only from those with a high bitrate and it is even worse for those with low bitrate.

The size of a packet that signals transaction information for one transaction from a user to the headend is assumed to be 100 bytes.

Table 5.17: Parameters of the traffic model

Parameter	Value	Description
Traffic variant	*Basic*	Traffic model variants, see Section 5.3.3
DL bitrate per User	1.0 Mbit/s	Average requested bitrate per customer on the access network in downlink direction
UL bitrate per User	0.4 Mbit/s	Average requested bitrate per customer on the access network in uplink direction
Signaling overhead	100 Byte	Required data volume on the access network in uplink direction to signal the transaction information once for one transaction.

Utility Model Parameters

A second part of the traffic model is the model for the transaction utilities. The utility model is introduced in Section 5.2. The parameters of the utility model are listed in Table 5.18.

Unless indicated otherwise, the *Basic* variant of the utility model is used. The transaction knowledge is not degraded, except for studies on the influence of degraded transaction information. The utility model contains parameters for additional variation of utility and time values, these variation parameters are set to $\sigma = 1$. In the utility models for some traffic classes, time and utility values partially depend on the size of the corresponding transaction. These time values also depend on the bitrate, the user expects to obtain for his transactions. This expected bitrate is usually lower than the available bitrate, since the access network is shared between multiple users. The expected bitrate is also asymmetric for uplink and downlink, because current access network technologies are usually asymmetric, and because the focus of this work is on the downlink direction.

Scheduler Parameters

Five different scheduler models are evaluated. The scheduler models are introduced in Section 4.3. The QoT scheduler contains two parameters, which are set according to Table 5.19.

These parameters are selected, so the scheduler works as explained. The objective of this work is not to find the best possible scheduler, therefore it is not necessary to find the optimal parameters. The QoT scheduler just needs to be good enough to show that it is possible to achieve a significant benefit by using the QoT transaction information.

Table 5.18: Parameters of utility models, see Section 5.2.

Parameter	Value	Description
Utility model	*Basic*	Utility model variant and influence of users expectation (Section 5.2.3)
σ_t^d	0	Degradation of transaction knowledge about time values sizes (Section 5.2.6)
σ_U^d	0	Degradation of transaction knowledge about utility values (Section 5.2.6)
σ_s^d	0	Degradation of transaction knowledge about transaction sizes (Section 5.3.4)
σ_t^y	1.0	Additional random component for time values for utility functions (Section 5.2.6)
σ_U^y	1.0	Additional random component for utility values in utility functions (Section 5.2.6)
h_U	10	Asymmetry factor for good versus bad user experience (Section 5.2.2)
$R_{\text{exp,UL}}$	1.0 Mbit/s	Expected data-rate of the access medium in uplink direction (bits per second)
$R_{\text{exp,DL}}$	10.0 Mbit/s	Expected data-rate of the access medium in downlink direction (bits per second)
U_b	$\frac{1}{1\text{ GByte}}$	Overall average valuation for transactions per Byte, depending on each transactions size.

Table 5.19: Parameters of QoT scheduler. See Section 4.3.

Parameter	Value	Description
n_f	4.0	Relative delay assumption of QoT scheduler
n_e	2.0	Size exponent of QoT scheduler

Simulation Setup Parameters

General simulation parameters are listed in Table 5.20. Each simulation run simulates 25000 seconds of real-time, which corresponds to the active period of one day, roughly 7 hours. Due to the high variance in the traffic model, the resulting metrics also have a high variance. Therefore, each simulation run is repeated typically 100 times with different seeds for all random number generators.

Table 5.20: Parameters for the simulation setup

Parameter	Value	Description
Simulated time	25000 s	The simulated time period of each simulation is roughly 7 hours
Transient phase	1000 s	The transient phase is 1000 seconds (17 minutes), see Section 4.1.3
Iterations	100	Repetitions for each simulation run

5.6 Model Characteristics (Analysis)

In this section, characteristic metrics of the traffic model are shown. It is shown, that the traffic model is suitable to analyze the performance and characteristics of QoT in the evaluations in Chapter 6.

The traffic model is analyzed in simulation studies with an access network of 10 subscribers, each subscriber generating 1 Mbit/s of traffic. The available bitrate in the access network is 50 Mbit/s to avoid any effect resulting from insufficient bitrate. A simple round-robin scheduler is used. To obtain comparable results, the simulation duration is 1 million seconds.

The generated shares of each application class model in the traffic mix is shown in Figure 5.7. The model is designed so these shares are comparable to the shares measured by Sandvine [San12a] and Cisco [Cis12].

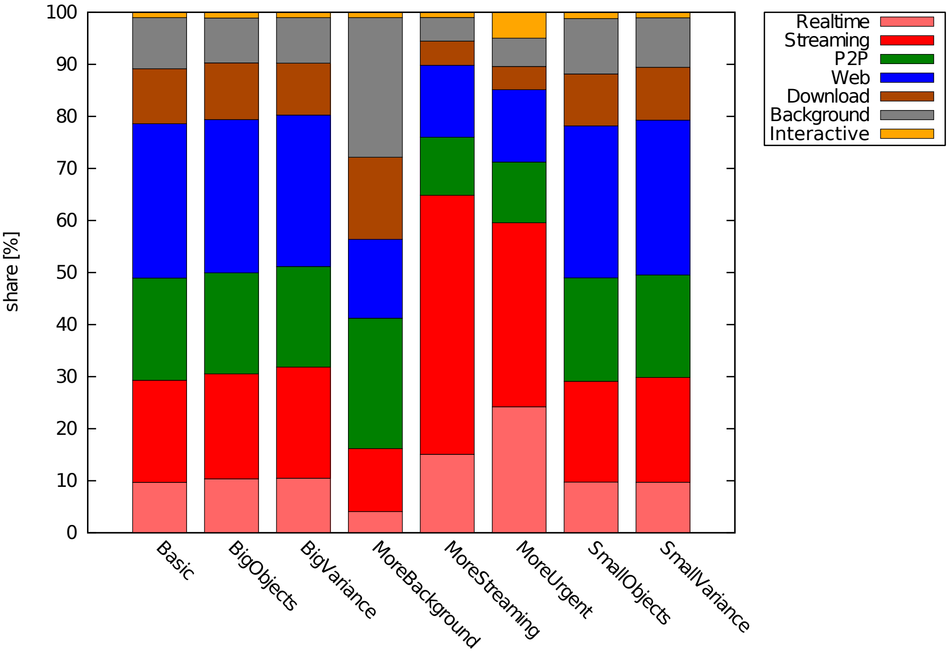


Figure 5.7: Traffic mix for all model variants

Figure 5.8 shows the Cumulative Distribution Function (CDF) of the sizes of chunks and trans-

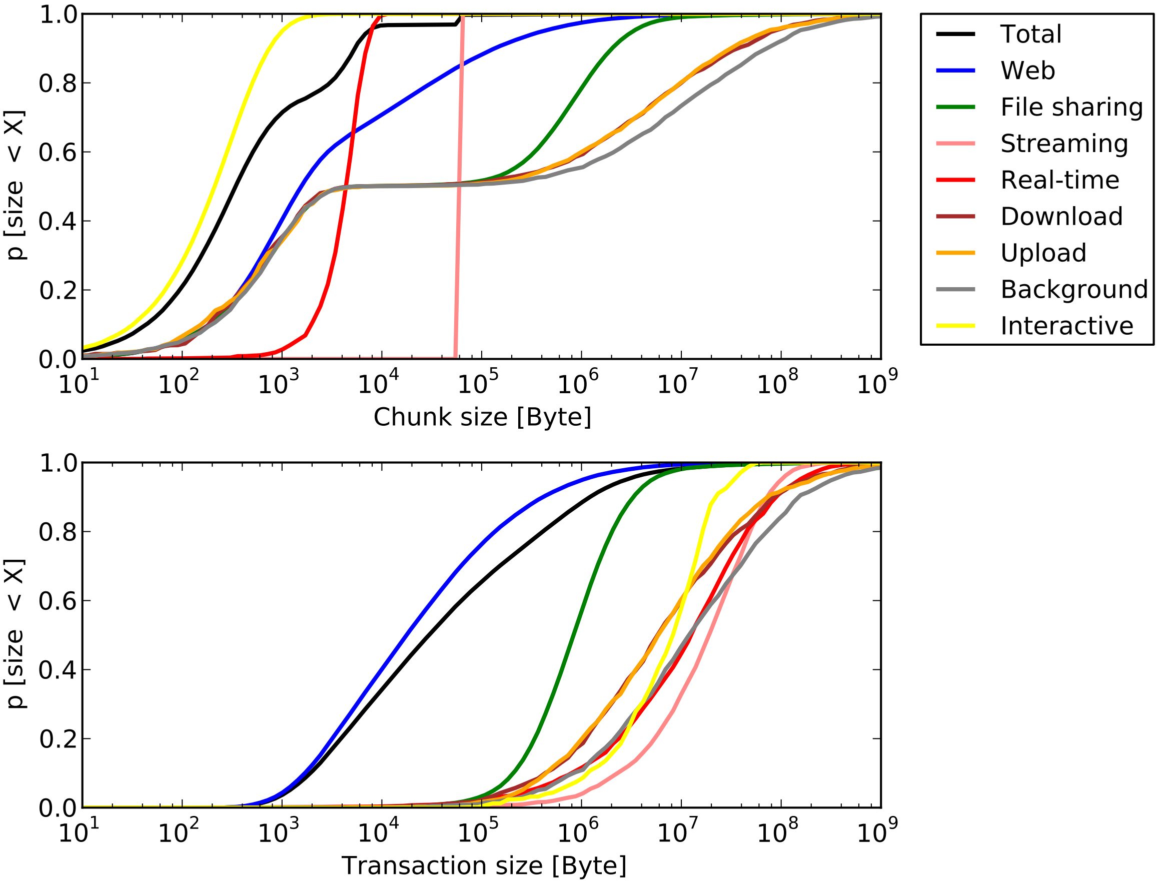


Figure 5.8: CDF of chunk and transaction sizes for model variant Basic

actions for model variant Basic. The upper subplot shows the CDF of the sizes of traffic chunks and the lower subplot shows the CDF of the sizes of transactions.

The vertical axis is the size of a traffic chunk respectively a transaction in bytes, the horizontal axes the relative frequency of traffic chunks or transaction to be smaller than the given size. Each colored curve represents the CDF of sizes of one application class model, the black curve represents the overall CDF for all application class models.

In the upper subplot, the chunks of the *Streaming* application class model have a constant size of 64 KB, because the chunks here are just checkpoints for the evaluation of the stream. Most other application class models show a multi-modal log-normal distribution, because they consist of multiple chunk types, for examples request chunks with a small size and response chunks with a large size.

As the corresponding transactions consist of multiple chunks, the size CDF of transactions of these application classes is not multi-modal. The distribution of the resulting transaction sizes is log-normal, which corresponds to the desired characteristic from literature. The total CDF of transaction sizes is dominated by the *Web* application class model, where there are many small transactions.

For additional analysis of the traffic model introduced in this chapter see Appendix B.

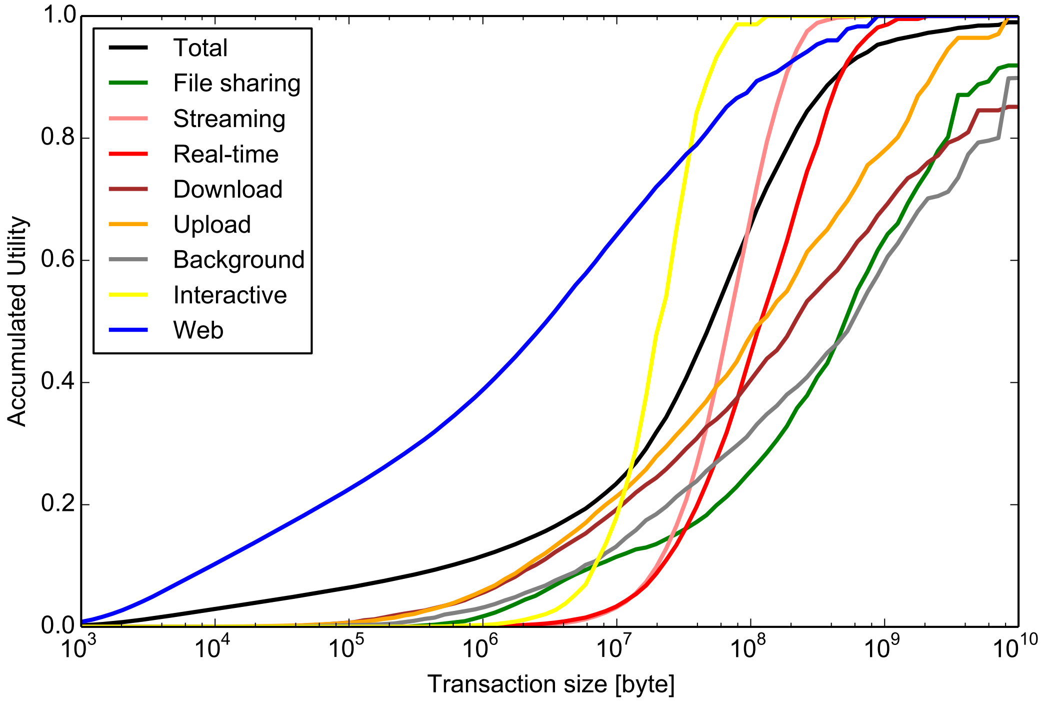


Figure 5.9: Accumulated utility depending on transaction size.

Figure 5.9 shows the accumulated utility for transactions depending on size. The vertical axis is the size of a transactions in bytes. The horizontal axes shows the percentage of the total utility for transactions of the given type. Each colored curve represents the accumulated utility of all transactions of one application class model, with transaction size smaller than the given value on the vertical axis. The black curve represents all transactions regardless of their application class model.

In comparison to the distribution of the transaction sizes in Figure 5.8, the accumulated utility in Figure 5.9 is shifted towards larger transaction sizes, because large transactions are comparably rare, but the utility of these large transactions has a large value, because the utility of a transaction has a term, that is proportional to the transactions size. The curves with the smallest shift towards larger transaction sizes are for the application class models *Web* and *Interactive*. This is the desired behavior, because these application class models use the larges fixed term for their utility models.

Further analysis of the utility model introduced in this chapter can be also found in Appendix B.

5.7 Summary and Conclusion

The traffic model introduced in this chapter is designed to represent current traffic in the Internet on an abstraction level that is best suited for evaluating the QoT approach in the following Chapter 6. This traffic model includes a model to evaluate the perceived QoS using metrics that can be evaluated in a network simulation.

The traffic model is based on the concept of transactions, introduced in Section 2.4. The transactions are generated by seven application class models. Each application class model represents the traffic of a group of applications with similar traffic characteristics.

This traffic model allows to simulate specific characteristics of use cases such as *Web browsing* or *video streaming* even with low aggregation or even a single subscriber on an access network. By abstracting from specific products, applications or Internet services, this approach is generic enough to be valid for a wide range of networks and user demography. This also allows to use this model to simulate and evaluate for future Internet traffic.

The analysis of the traffic model in Section 5.6 shows that all application class models behave as expected. Especially the sizes of the generated chunks are showing the desired characteristics.

6 Quantitative Performance Evaluation

In this chapter, QoT is analyzed for the technical criteria listed in Section 1.4. These technical criteria are:

- A system has to be able to improve the defined quality metric significantly.
- The improvement has to be robust against uncertainties in the measurement and the quantification of the metric.
- The system has to be robust against changes in traffic characteristics.
- The system has to work within a wide range of network topologies and network properties.

The analysis uses the methodology and models explained in Chapter 4. The comparison is made with the schedulers explained in Section 4.3. These are the Round-Robin Scheduler, the Smallest-Size Scheduler, the Traffic-Class Scheduler, and a second scheduler based on QoT information.

Section 6.1 shows that QoT can significantly increase the utility metric defined in Section 2.5. How QoT performs in different network topologies and its network properties is analyzed in Section 6.2. There, for example, access networks with different numbers of subscribers sharing the access medium are shown. Section 6.3 analyzes whether QoT is robust against changes in traffic characteristics. At the end, Section 6.4 analyzes the robustness of QoT against uncertainties in the measurement and the quantification of the metric.

6.1 General Performance of QoT

This section describes the general performance of QoT and shows that it significantly increases the utility metric defined in Section 2.5.

First, it illustrates the general performance gain. Following, in Section 6.1.1 the performance gain is shown from a user's perspective, and in Section 6.1.2, from an operator's perspective.

To show the general performance of QoT, Figure 6.1 compares the accumulated utility per subscriber and month for an access network with 10 active subscribers for QoT with a best-effort approach introduced in Section 4.3.3. The available bitrate of the network is varied, and

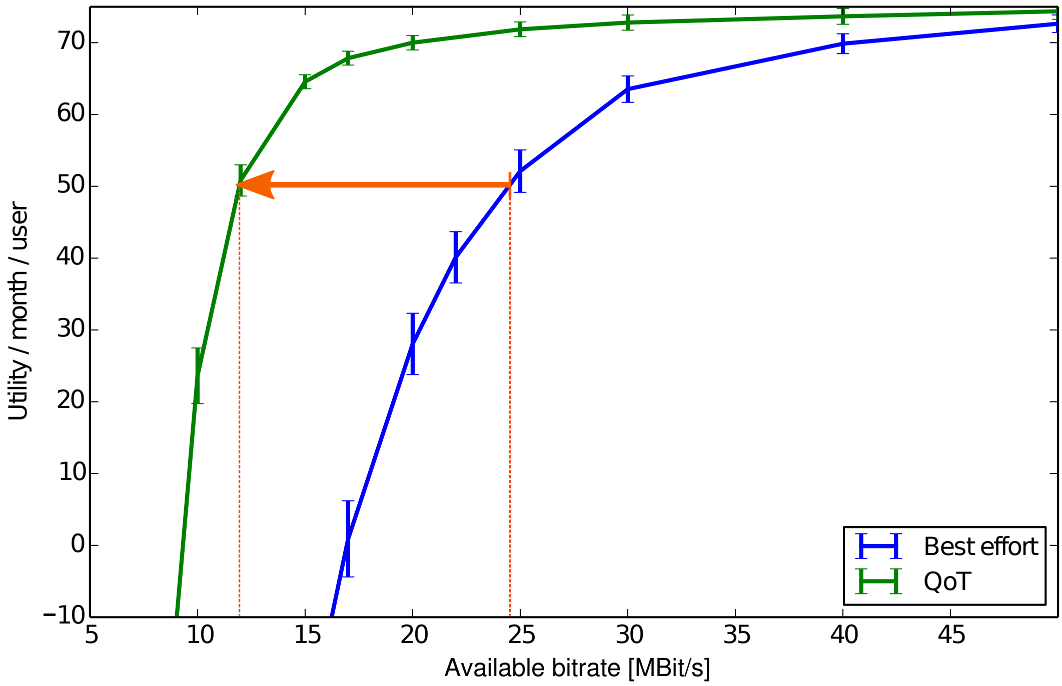


Figure 6.1: Total utility for a network with 10 subscribers, depending on traffic model variant, 1 Mbit/s offered load per subscriber.

plotted on the vertical axis. Each subscriber generates traffic in the downlink direction at an average of 1 Mbit/s with the traffic model variant *Basic*. Further parameters of the simulation environment are listed in Section 5.5.

The access network on the left hand side of the plot is sufficiently over-dimensioned, so that traditionally best-effort schedulers already yield good user experience. At below 20 Mbit/s, or a utilization of the access medium of 50%, the users' experience is impaired.

For a sum of 50 utility units per month, the access network capacity would have to be increased by 110% to achieve the same user experience without QoT.

In comparison to best effort, QoT achieves the same user experience but with a lower available bitrate, in the full simulated bitrate range.

6.1.1 From a User's Perspective

In Figure 6.1, the available bitrate is varied. From a user's perspective, however, the available bitrate is fixed. The subscriber may thus instead request and transmit more data over the access network.

Figure 6.2 shows the sum utility for each subscriber over one month. The access network and topology are fixed. The available bitrate in the access network is 20 Mbit/s and the network is shared by 10 subscribers.

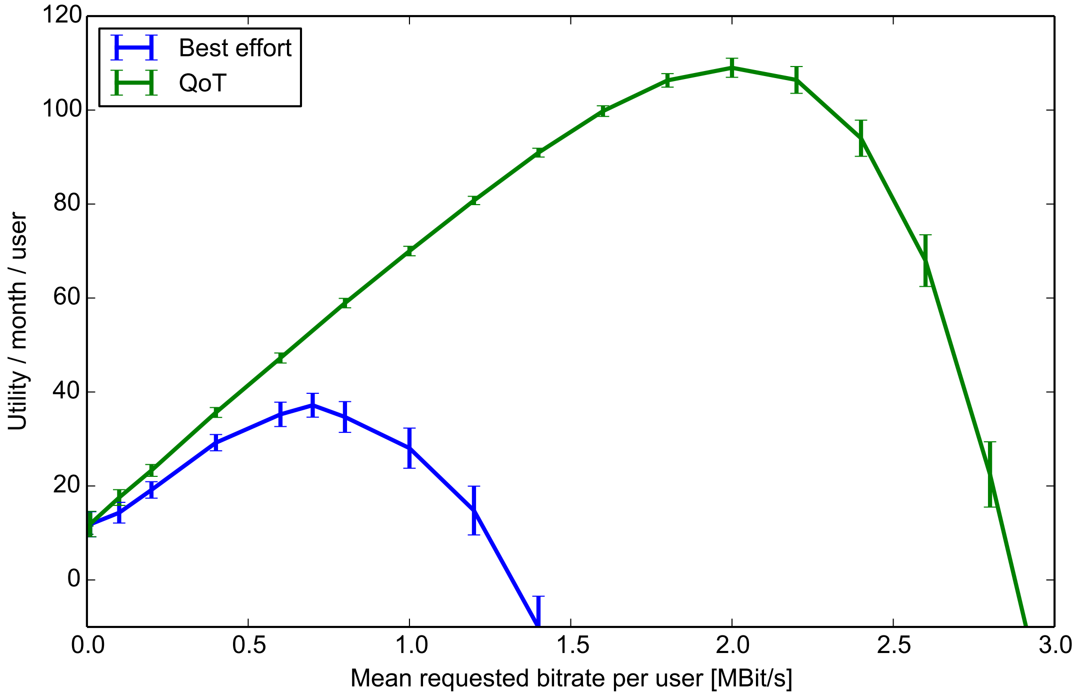


Figure 6.2: The total utility per user for a network with 10 subscribers, and 20 Mbit/s, where the load offered per subscriber is varied.

The vertical axis specifies the average requested bitrate of each subscriber. With an unlimited available bitrate, the utility value would increase linearly, because of the linear relation between the utility, and the transmitted amount of data inherent to the traffic and utility model.

Effects that arise from the different usage of subscribers on one access network are not of interest for this work, and should be avoided. This means that in this work, it is not relevant to observe how one user profits at the cost of other users. To avoid these effects, this evaluation provides that all users increase their requested bitrate equally.

With a best-effort scheduler, the utility increases up to around 600 kbit/s until it decreases, because the shared access becomes congested from time to time. With QoT, the utility of the subscriber exceeds the utility for a best-effort scheduler in all cases. In this case, the utility value increases to around 1.2 Mbit/s until it decreases, because the shared access becomes congested from time to time.

6.1.2 From an Operator's Perspective

For an operator of a shared access network, it is usually easier to vary the number of subscribers sharing the access medium than to vary the available bitrate, because the latter is often fixed for a given technology. Figure 6.3 shows the sum utility of all subscribers per month. This sum is the revenue the operator could demand from its subscribers, because the utility a subscriber achieves corresponds to the value this access network has for the subscriber. While the available data-rate of the access network is fixed at 15 Mbit/s, the number of users is varied.

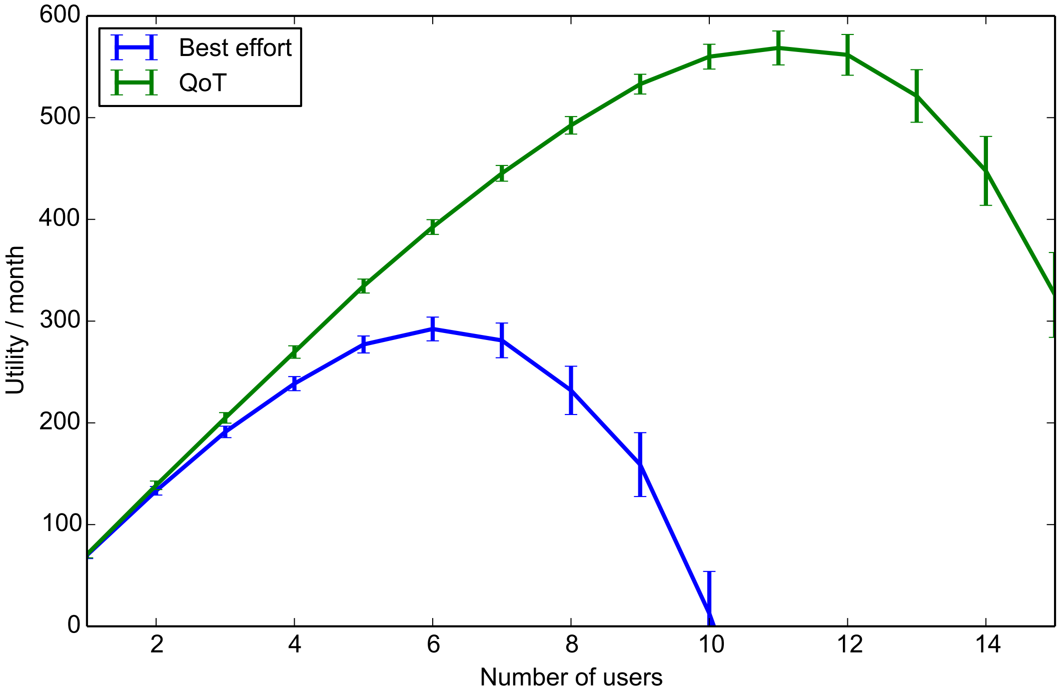


Figure 6.3: Total utility for a network with 15 Mbit/s available bitrate, where the number of subscribers is varied.

On the left hand side, a single subscriber requesting in average 1 Mbit/s of data-rate achieves a good sum utility of around 60. In this case, the network is mostly idle; therefore, the scheduler has no influence on the resulting utility value. With a second subscriber, the revenue of the network can be nearly doubled. Without a QoS schema for this access network, the maximum revenue is reached at 6 subscribers. With a lower number of subscribers, the utilization of the network is low, and adding another subscriber will increase the sum of the resulting utility for all subscribers. At already 6 or more subscribers, adding another subscriber will decrease the resulting utility for the existing subscribers more than the additional utility from the additional subscriber.

With an increasing number of subscribers, the total utility of the network increases in a roughly linear manner, until the utilization of the network becomes noticeable. The maximum amount of utility the network can generate with a *Best effort* approach is a utility of nearly 300 with six customers. With QoT, the network can generate a utility of 560 with eleven and with twelve customers sharing the access network.

6.1.3 Analyzing the Performance Gain of QoT

The performance gain of QoT is achieved by scheduling the transmissions on the access medium. In the presence of QoT, prioritised traffic such as *Web* traffic displaces less prioritised traffic such as *P2P*, *Background*, or *Download* traffic. In case of best-effort packet handling without a QoS scheme, different traffic classes share the resources equally, regardless of the different requirements. Figure 6.4 shows an example of how the resources are distributed for transactions of different traffic classes. The traffic for all users is shown, separated by traffic class and accumulated over one-second intervals. The access network offers 15 Mbit/s downlink data-rate, with 10 subscribers requesting each one Mbit/s on average.

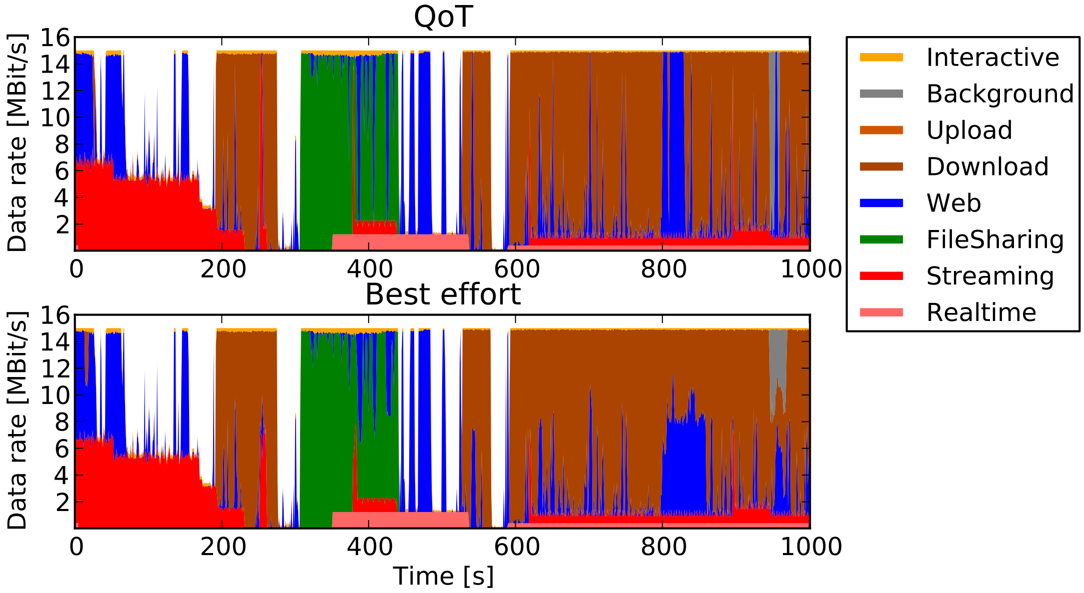


Figure 6.4: Example of the share of traffic types, with and without QoT.

This example shows how QoT influences the traffic. Over a longer time scale, the share of each traffic class is nearly identical. This means that, in both cases, the user is triggering the same traffic at the same time, and this traffic is transmitted nearly at the same time. On a smaller time scale, tall, narrow blue spikes can be seen in the case of QoT, while in the comparison, these spikes are wider and less tall. These blue spikes represent the *Web* traffic, which usually has a higher value to the user than *Download* traffic. In the case of QoT, the traffic of higher value can use the full bitrate; and the transactions are therefore finished earlier.

When the bitrate of the access network is sufficient for the requested average bitrate of the subscribers, nearly all of the requested data is transmitted. Therefore, the data volume of each traffic class depends only on the traffic model.

When the bitrate of the access network is not sufficient, some of the traffic cannot be transmitted. This mean, transactions are delayed up to the point where the transaction is dropped because of an application timeout, or because the user has canceled the transaction, e.g. by closing a browser tab or stopping a download.

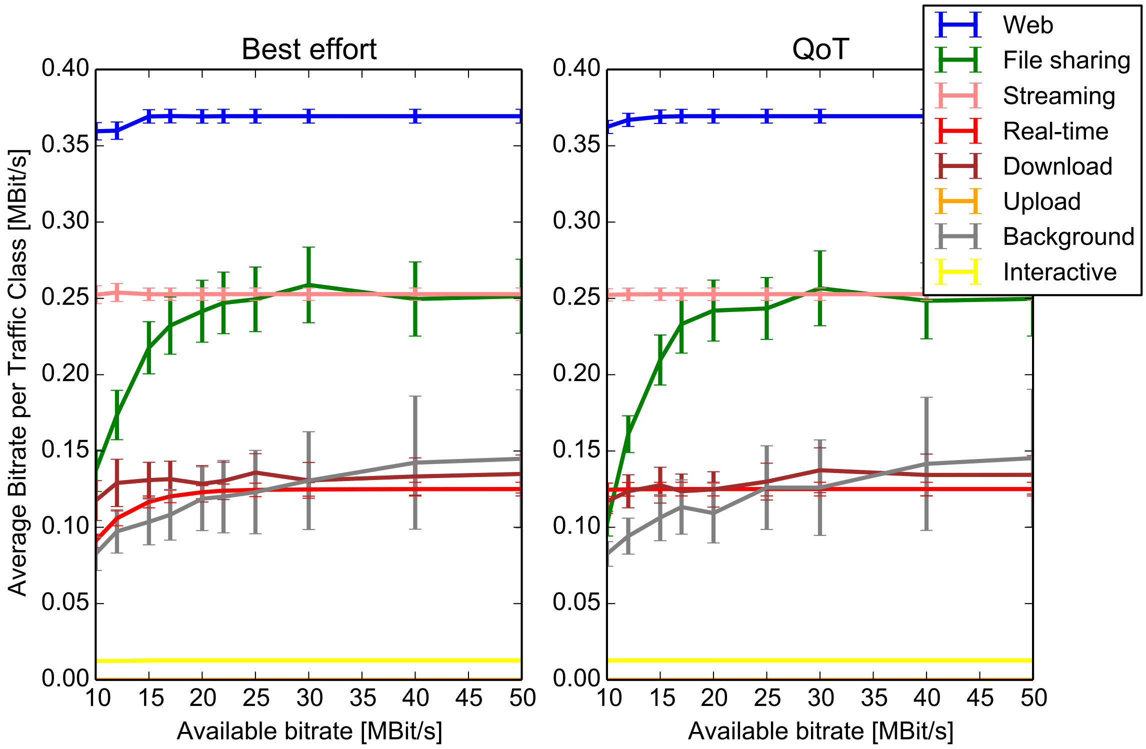


Figure 6.5: Proportion of traffic volume of each traffic class.

Figure 6.5 shows the traffic volume of each traffic class. The available bitrate is represented by the vertical axis. One might expect to see a sharp bend at the point where the bitrate of the access network is equal to the requested bitrate. However, there is a transition range. Because of the bursty behavior of Internet traffic, there are periods where the bitrate of the access network is not sufficient; nevertheless, it is sufficient in average, and vice versa.

The plots on the left hand side show the traffic volumes without QoS, the right hand side plots show the traffic volumes when using QoT. Differences can be seen for insufficient available bitrate, because some traffic types have a large share of transactions that have to be prioritized by QoT. These prioritized transactions will therefore less likely be delayed to the point where the transaction is dropped.

6.2 Scenario Dependency of QoT

This section shows how QoT performs with different network topologies and network properties. Because the network model abstracts from specific access technologies, the network topology of an access network in this model is mainly defined by the number of users sharing the same resources and the total available bitrate.

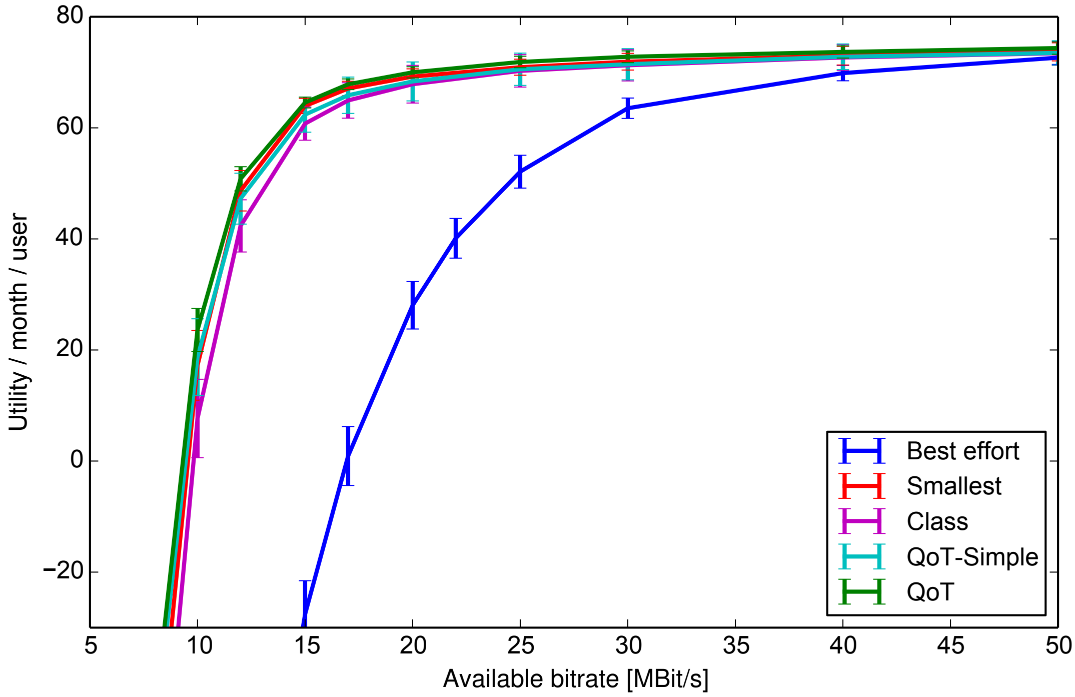


Figure 6.6: Total utility depending on the available bitrate.

Figure 6.6 shows how the total utility per user increases with increasing available bitrate. In addition to QoT and the *Best effort* approach, this figure shows three further QoS schedulers for comparison, which are introduced in Section 4.3:

- Smallest-Size Scheduler (*Smallest*)
- Traffic-Class Scheduler (*Class*)
- Size-and-Class Heuristic (*QoT-Simple*)

The differences in the resulting utility values between the four QoS schedulers are small; however, the *Best effort* approach without QoS leads to significantly worse utility values.

For larger bitrates (right hand side of the plot), all compared schedulers, except for *Best effort*, achieve nearly the optimal utility value. Therefore, in these cases, there is not much difference between those schedulers.

For lower available bitrates (left hand side of the plot), the available bitrate is not always sufficient to satisfy the users' required peak bitrate, which affects all schedulers to the same degree, therefore, the difference between these schedulers is small.

In Figure 6.6 the number of subscribers is constant while the available bitrate is varied, whereas in Figure 6.7 the available bitrate is constant while the number of subscribers is varied.

This figure is similar to Figure 6.3. In addition, the three comparison schedulers *Class*, *Smallest*,

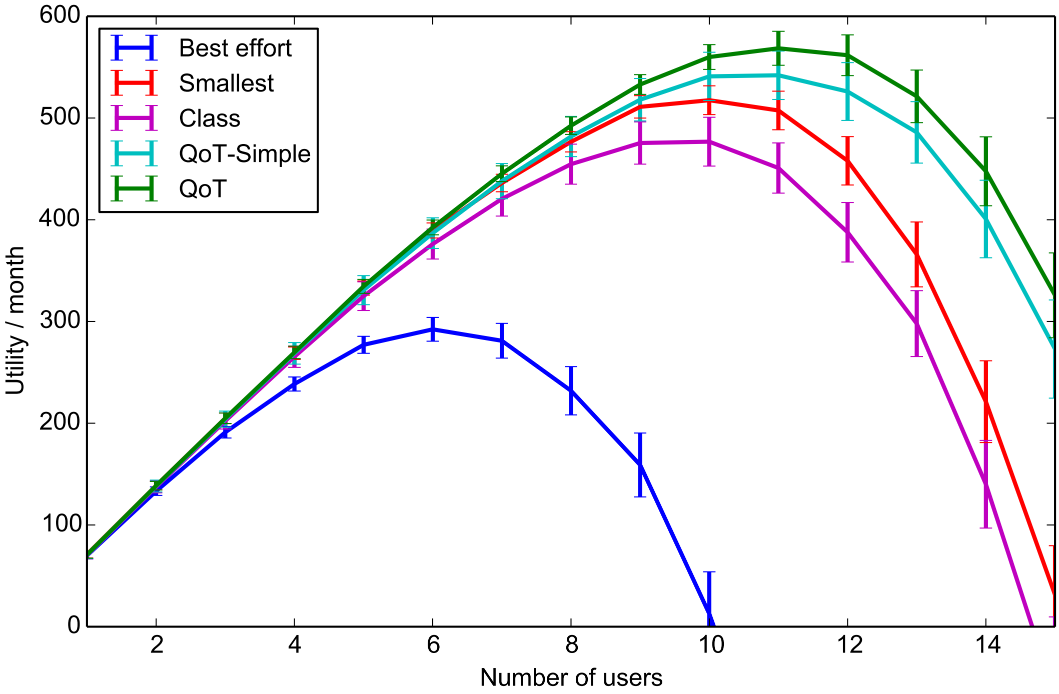


Figure 6.7: Total utility for a network, depending on the number of users.

and *QoT-Simple* are shown.

On the left side of the plot, there is only one subscriber using the access network with its 15 Mbit/s available bitrate. In this case, the network is mostly idle; therefore, the scheduler has no influence on the resulting utility value.

With an increasing number of subscribers, the total utility of the network increases in a roughly linear manner, until the utilization of the network becomes noticeable.

In this case, differences between the four QoS schedulers are more noticeable, because in all cases, the available bitrate is 15 Mbit/s, which is a sufficient peak bitrate for most of the users' transactions. All four QoS schedulers achieve a significant gain over *Best effort*. While QoT yields the highest revenue, the simple scheduler using QoT information yields the second highest revenue and the two schedulers that use only one single value of the QoT information (transaction size or general transaction importance) are third and fourth.

It is obvious that there are multiple factors that influence the users' experience in these evaluations:

- Scheduler
- Utilization
- Peak bitrate

When the number of customers that share the access network increases, and the available bitrate in this access network increases at the same time, this means that the utilization is constant, but the peak bitrate increases.

Beyond the obvious benefit from the increased peak bitrate, the users' experience is improved by the network effect (economy of scale), because the higher the number of connected customers, the less likely it is that all customers fully utilize their share of the bitrate.

It is, therefore, important to analyze how the gains of QoT compare to the gains from an increased aggregation. Figure 6.8 shows this interaction of aggregation gain and QoT. The vertical axis shows the average utility value for one subscriber, accumulated over one month. The horizontal axis shows the multiplexing factor, which is the number of users that share the available bitrate of the access network. For each of the curves, the available bitrate per subscriber is constant; therefore, the available bitrate of the access network increases proportionally to the number of users.

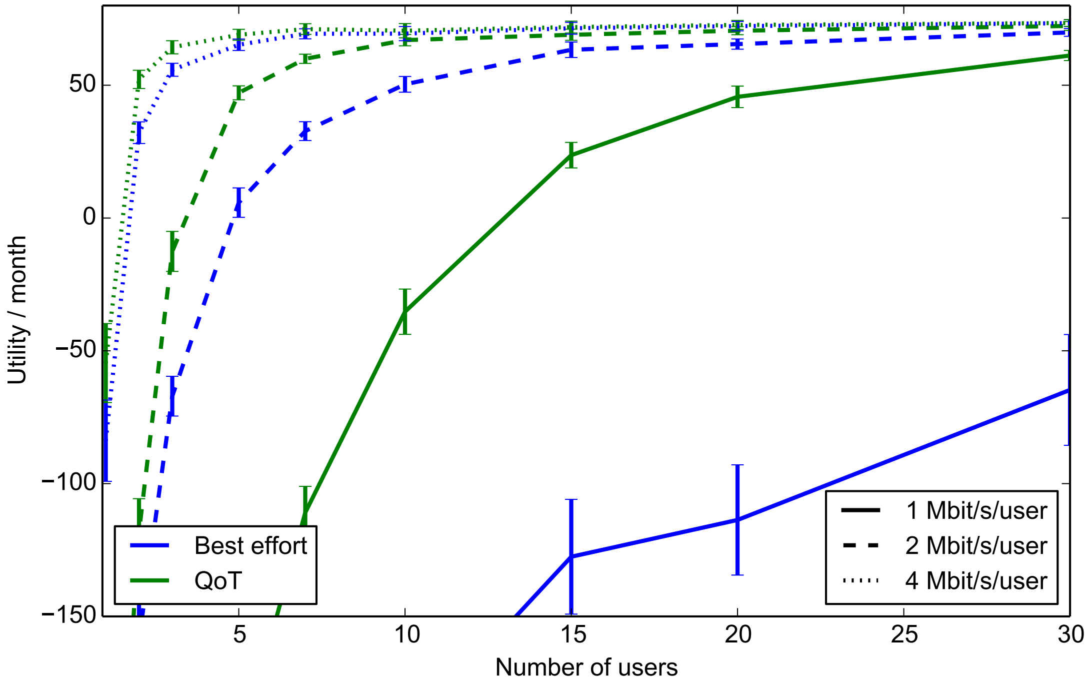


Figure 6.8: Aggregation gain: Varying numbers of users access network data-rate together.

For the dotted curves, there are 4 Mbit/s available per subscriber. This means that the available bitrate is 4 Mbit/s in case of one subscriber, and 40 Mbit/s in case of 10 subscribers. This is already sufficient in nearly all cases; therefore, there is no room for improvement using QoT or any other QoS approach. The utility is nearly the same perfect value in all cases, except for small number of users (one and two users), for who the peak bitrate (4 or 8 Mbit/s) is not sufficient in all cases.

The solid curves show the case, where the available bitrate of 1 Mbit/s per subscriber is not sufficient. The blue curve shows the situation without any QoS system, where the utility is below zero in all cases, which means that this network is unbearable for its users. With QoT, the utility is also below zero for less than 15 subscribers in a 15 Mbit/s network. However, the aggregation gain adds to the performance gain of QoT, and for more than 15 subscribers the system becomes acceptable for subscribers.

The dashed curves, which represent an access network with 2 Mbit/s per subscriber, confirms that an access network with QoT benefits in the same way from aggregation gain, as does a system without any QoS approach. Obviously, when the system has reached a perfect experience for the user, neither QoT nor a higher multiplexing factor can further improve the utility.

The *Best effort* cases show the gain from increasing the aggregation. QoT always offers additional gain. It can be seen that the benefit from QoT is even larger for networks with higher aggregation. This corresponds to expectation, because the higher the number of parallel transactions, the higher the degree of freedom for the scheduler to choose the right transaction.

6.3 Traffic Dependency of QoT

The previous sections show how the QoT approach performs under different network scenarios with one traffic scenario. It is, however, important that QoT works in a wide range of traffic types, because it is unknown what the characteristics of future traffic will be. Therefore, in this section, QoT is evaluated with different models for the traffic and different characteristics of the traffic model.

It is also important to evaluate that QoT works with a wide range of traffic models and parameterizations in the traffic model. As stated in Section 5.1, it is not possible to find a single traffic model, which fits all users in all situations and all networks. Therefore, it is important that the results are valid for a wide range of possible models and characteristics.

The most prominent of these characteristics is the data volume that each subscriber takes up each month. In the traffic model, this translates to a specific mean value of the requested bitrate. In Section 6.3.1, QoT is evaluated with the different requested bitrates of the subscribers.

In Section 6.3.2, the gains from several different traffic scenarios will be shown, and Section 6.3.3 shows different models for the formation of a user's valuation of a transaction.

In Section 6.3.4, QoT is analyzed under the additional variation of the users' requirements. These comprises additional variation of utility values. In Section 6.3.5, the asymmetry between the positive and negative user experiences is varied.

6.3.1 Requested Bitrate

In this section, the requested bitrate of customers is to be varied. QoT is analyzed in different scenarios where the subscribers to the access network generate different amounts of traffic. This differing amount of traffic is formulated as the mean requested bitrate per customer, because the mean requested bitrate corresponds to a data volume that each customer downloads each month. The data volume of a subscriber is a highly individual parameter, both for an individual subscriber and for a scenario. In particular, it is expected that this data volume will continue to increase over time.

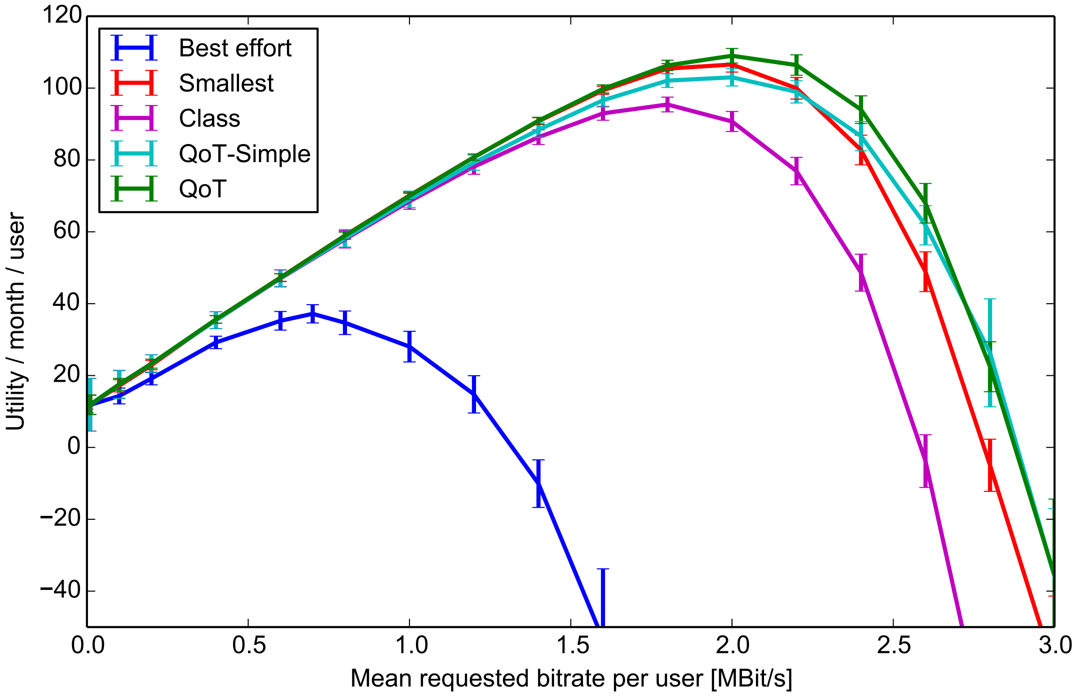


Figure 6.9: Total utility for a network with 10 subscribers, and 20 Mbit/s. The offered load per user is varied.

Figure 6.9 shows the average utility of each user over the mean requested bitrate. On the left side of the plot, the users do not request any traffic in the downlink direction. However, the uplink traffic is kept constant throughout this evaluation; therefore, the plots show an offset for the utility, which results from transactions that only generate traffic in the uplink direction. These transactions are from the traffic classes *Upload*, *Background*, *Real-time*, and *Interactive*.

With increasing requested bitrate, the traffic class models increase the requested downlink bitrate. For most traffic class models, this means that the arrival rate of transactions increases. The network, therefore, transmits more transactions. The maximum achievable utility for the traffic-class models is partially proportional to the transmitted data volume, and partially proportional to the number of transactions. In this case, the maximum achievable utility, therefore, is proportional to the requested bitrate. The results fit to this expectation: on the left side of the plot, the utility per customer increases nearly proportional to the requested bitrate. This is limited when the utilization of the access network becomes noticeable, which limits the resulting utility. The maximum utility resulting is reached at 0.7 Mbit/s in the *Best effort* case, and around 2 Mbit/s for the four QoS schedulers compared.

This result shows that when the requested bitrate is varied, QoT improves the performance of an access network. Again, the differences among the compared QoS schedulers are small, compared to the gap by the best-effort scheduler.

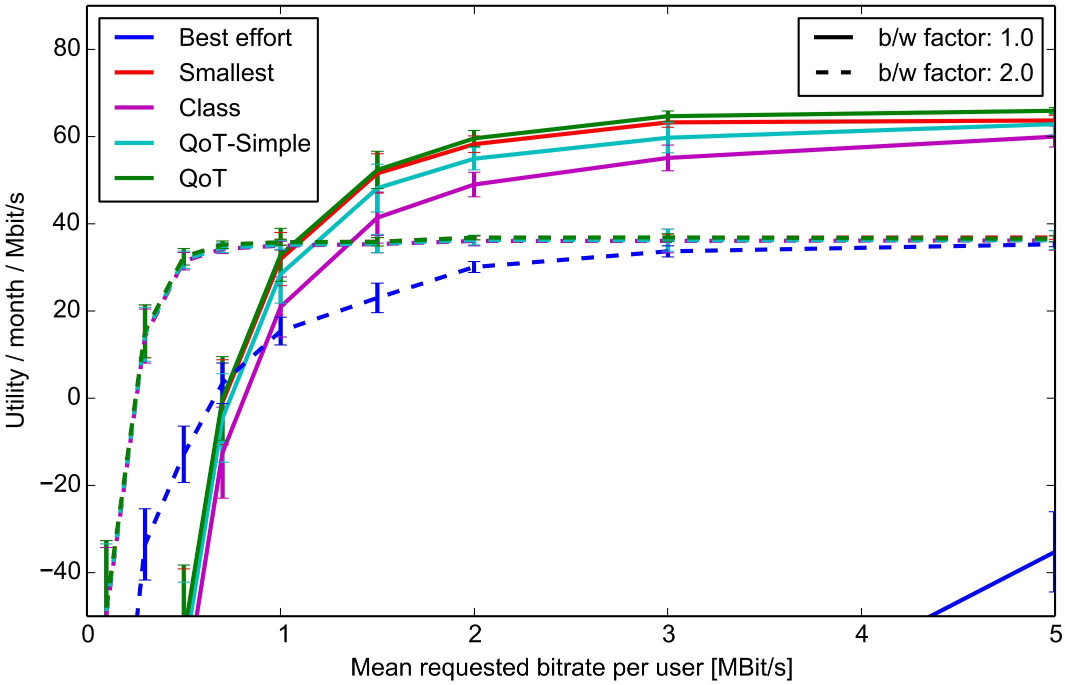


Figure 6.10: Total utility for a network with 10 subscribers. The available bitrate depends on the total requested bitrate.

Figure 6.9 shows simulation results for varying requested bitrates in a single network scenario with 10 users and 20 Mbit/s available bitrate, which means that the network utilization depends on the requested bitrate. In Figure 6.10, the network utilization is kept constant. The simulations scenarios for this figure set the available bitrate of the access network proportional to the requested bitrate. Therefore, the requested bitrate is effectively the only varied parameter, and the remaining network scenario is configured accordingly.

The mean requested bitrate in the downlink direction is shown on the horizontal axis in Figure 6.9. The mean requested bitrate in the uplink direction is proportionally configured to 40%

of the mean requested bitrate in the downlink direction.

The available bitrate of the access network is equal to the combined requested bitrate of all users, multiplied by a given factor. When this factor is 1 (solid curves), the available bitrate is just enough to transmit the requested data in the downlink direction. This means that the utilization of the access network in downlink direction is nearly 100%. When this factor is 2 (dashed line), the available bitrate is twice the requested bitrate in the downlink direction; then the utilization is 50%.

The vertical axis shows the total utility in proportion to the available bitrate of the access network in downlink direction in Mbit/s.

The available bitrate in the uplink direction is equal to the available bitrate in the downlink direction. This ensures that the downlink direction has the main influence on the results, as the downlink direction has more traffic and a higher utilization. This also avoids an offset such as depicted on the left hand side of Figure 6.9.

On the left hand side of Figure 6.10, the resulting utility drops because the peak bitrate of the network becomes insufficient. On the right hand side, the curves become nearly horizontal because the resulting utility is mostly proportional to the requested bitrate, which in these simulations is configured as proportional to the available bitrate.

It can be seen that, in all configurations QoS systems can achieve a significant gain over best effort. In all cases, QoT achieves the highest gain.

With a lower factor for over-provisioning the available bitrate, the relative utility on the right-hand side of the plot is higher, because the utilization is higher. This is expected, as the curves in Figure 6.9 increase to a utilization of around 100%.

6.3.2 Traffic Variants

This section will present the gains from several different traffic scenarios. These traffic scenarios have been introduced and explained in Section 5.3.3.

Figure 6.11 shows the total utility per month per subscriber for a network with 20 Mbit/s available bitrate and 10 subscribers. The different schedulers are represented according to color. The different traffic scenarios are listed horizontally. It can be seen that QoT outperforms the other schedulers in almost all cases.

It is, however, obvious that the different traffic models do not yield the same overall utility for the same network. For some of the traffic variants, all of the schedulers yield a high utility value, while for other variants, none of the schedulers provides a sufficient utility. This is acceptable, because the traffic model variants are not designed to yield commensurate utilities. The traffic variants are designed to request the same average bitrate; however, different traffic classes and

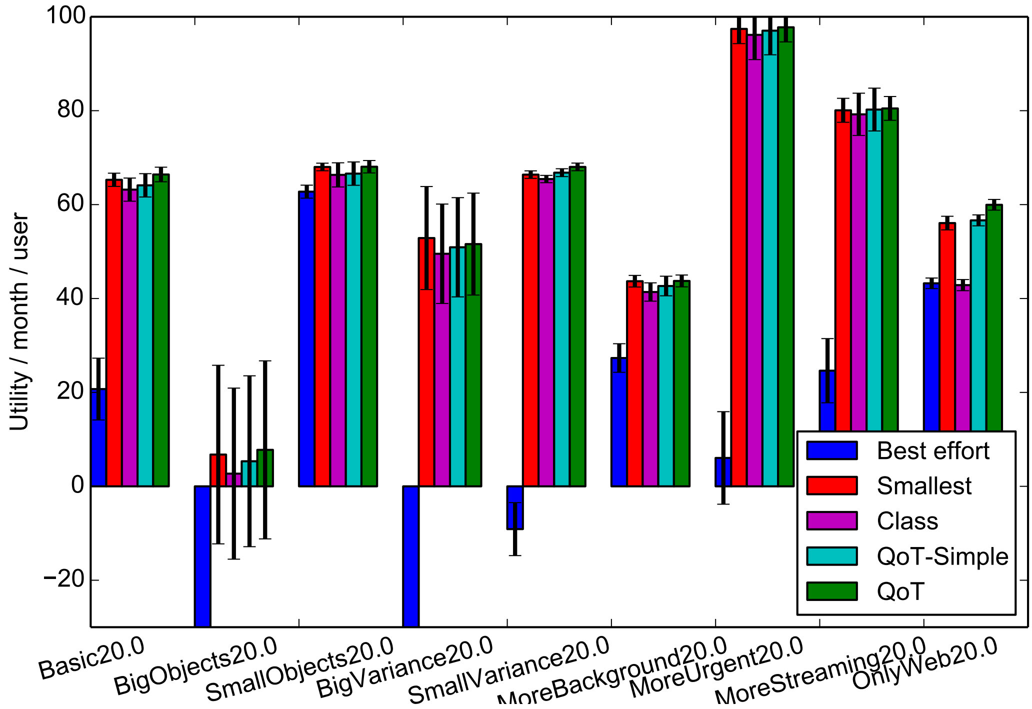


Figure 6.11: Total utility depending on traffic model variants.

different parametrization of the traffic classes represents traffic that has different characteristics regarding the users' requirements, e.g. how the user is going to wait for a given transaction. Since some types of traffic are more demanding than other types, the resulting utility obviously differs for the same available bitrate. At 20 Mbit/s, the traffic model variants with more urgent traffic and more streaming yield near to maximum for all types of schedulers, because the aggregated traffic in these cases is smoother, and cases where the 20 Mbit/s are not sufficient are rare. The traffic model variants with more *Web* traffic and with bigger objects create much more bursty aggregated traffic, with longer periods where the available bitrate is not sufficient. In these cases, none of the schedulers is able to yield acceptable utility.

In scenarios where the resulting utility is in the same range, the available bitrate is different, as in Figure 6.12, depending on the traffic model variant. Again, QoT yields the highest sum utility in most of the cases.

The performance of the class-based approach matches the expectation: in the cases with a high diversity of traffic components, a class-based scheduler yields good results. However, in the cases where one of the traffic class models dominates the traffic mix (*MoreBackground*, *MoreStreaming*, *MoreUrgent*, and *OnlyWeb*) a class-based approach has less diverse transactions in its input. In the extreme case, *OnlyWeb*, all traffic has the same class; therefore, the class-based scheduler has no usable input, and becomes a best-effort scheduler. Therefore, the performance

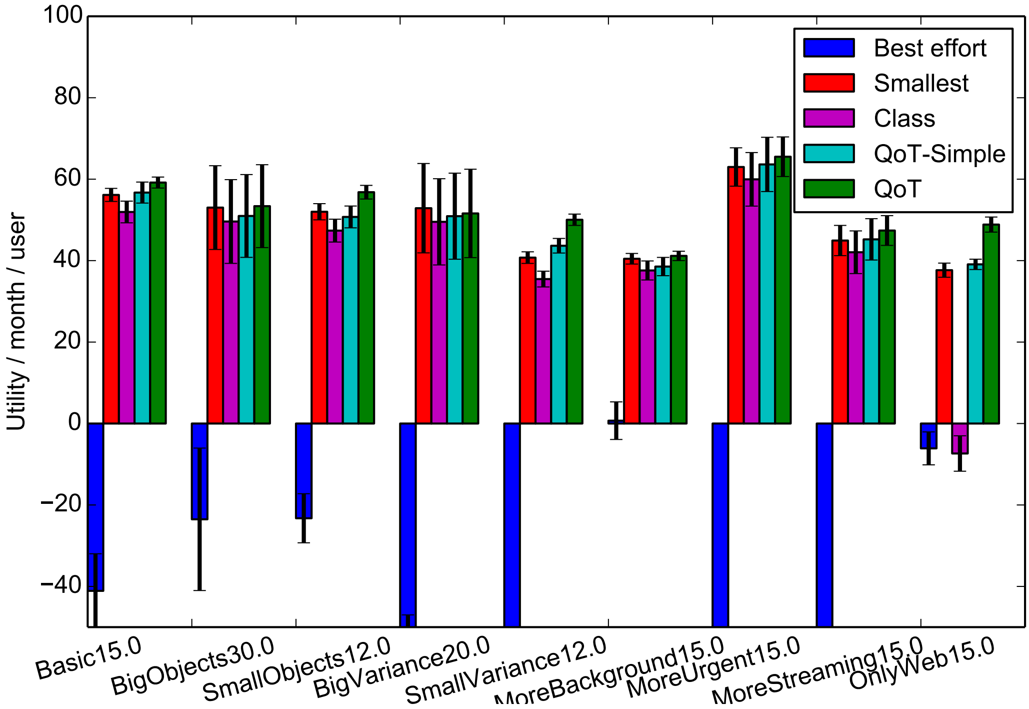


Figure 6.12: Total utility for a network with 10 subscribers, depending on traffic model variants, where the available bitrate is varied according to the traffic model variant.

of the class-based scheduler and a best-effort scheduler are very similar.

In addition to the bar plots in Figure 6.11 and Figure 6.12, plots for the full range of available bitrates can be found in Appendix C.

6.3.3 Utility Variants

Beyond the traffic model itself, it is also important to analyze QoT in different models for the utility values. In Section 5.2.3, three variants are introduced for assigning utility values to transactions: “Basic”, “Proportional”, and “Fixed”. In Section 5.4, the definition of the utility functions for each traffic class model depends on this utility model variant.

Figure 6.13 shows three sub-plots for these three utility variants. On the right-hand side of each plot, the available bitrate of the network increases, and the utilization of the network decreases. Therefore, the utility value converges to its maximum achievable value, which depends only on the traffic and utility model, but not on the scheduler.

It is obvious that the maximum achievable utility value is different for the three utility model variants. This is because the utility model variants do not use the maximum achievable utility in their definition, and there is no calibration to equalize this value.

The plots show that QoT can improve the users’ experience significantly in all cases, and that

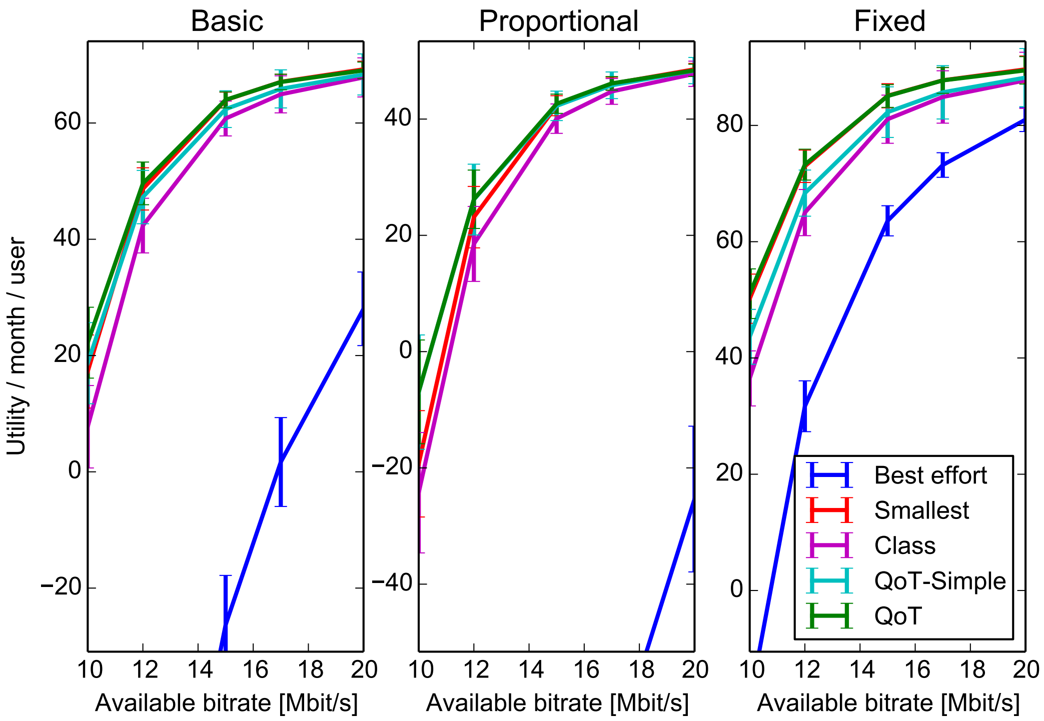


Figure 6.13: Utility model variants: “Basic”, “Proportional”, and “Fixed”

the improvement with QoT is greater than the improvement with the schedulers compared.

With the utility model variant *Fixed*, the utility of a transaction is less dependent on the size of this transaction. This means, that smaller transactions have a greater achievable utility value per Byte, thus the *Smallest* scheduler performs just as well as QoT, this also means that the improvement for the *Best effort* approach is lower, because in this case, all transactions will immediately receive a share of the available bitrate, and therefore, small transactions can be finished relatively early.

In the opposite case, with the utility model variant *Proportional*, the utility of a transaction is more dependent on the size of this transaction. This means that the utility per Byte depends more on the type of transaction than on its size. In this case, the improvement against the *Best effort* approach is even higher than in the *Basic* variant. In this case, the *QoT-Simple* schedule is just as good as QoT.

6.3.4 Additional Variation

In this section, QoT is analyzed with additional variation to the utility model. In Section 6.3.2, QoT was evaluated using traffic models that have more or less variation in its parameters, e.g. regarding the size of a transaction. In this section, two parameters of the utility model are modified to show additional variation:

- The parameter $v^d_{T,t}$ introduces additional variation to all time parameters, e.g. the expected finish-time of a transaction. This parameter is explained in Section 5.2.6, Subsection, *Variation in time parameters*.
- The parameter $v^d_{T,U}$ introduces additional variation to all utility values. This parameter is explained in Section 5.2.6, Subsection *Variation in utility parameters*.

Random values from a logarithmic-normal distribution are multiplicatively added to the original value from the utility model. A variation of $\sigma = 1$ means that, in around 10% of the cases, the resulting value is smaller or bigger than the original value by a factor of 5 or more. A variation of $\sigma = 3$ means that, in around 10% of the cases, the resulting value is smaller or bigger than the original value by a factor of 140 or more.

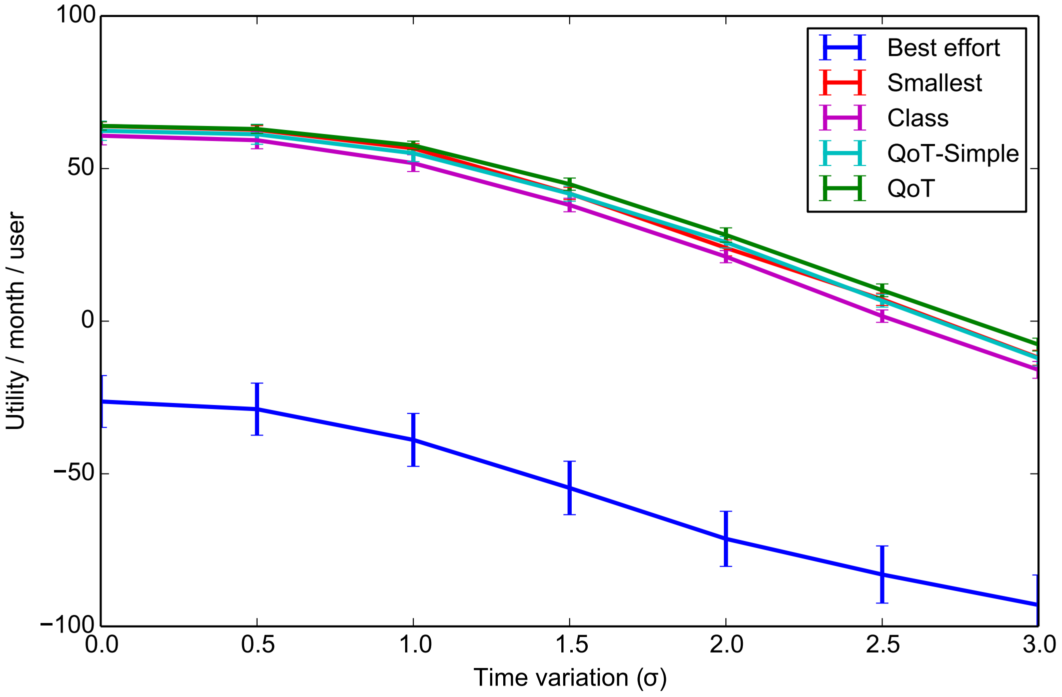


Figure 6.14: Variation of $v^d_{T,t}$.

Figure 6.14 shows the performance of QoT and the comparison schedulers with the additional variation of time parameters.

$v^d_{T,t} = 0$ means that there is no variation and that the expected finish-time of a transaction is deterministic, and only depends on the parameters of the transaction.

When $v^d_{T,t}$ increases, the average utility value decreases for all schedulers, because the additional time for some transactions yields only a small increase in its utility, while the transactions that are expected to be finished earlier will lose a larger part of their utility values. The reason is that it becomes more likely that they cannot be finished in time.

The improvement of QoT and the comparison scheduler is similar in all cases when compared to the *Best effort* approach, and QoT in all cases shows the highest improvement.

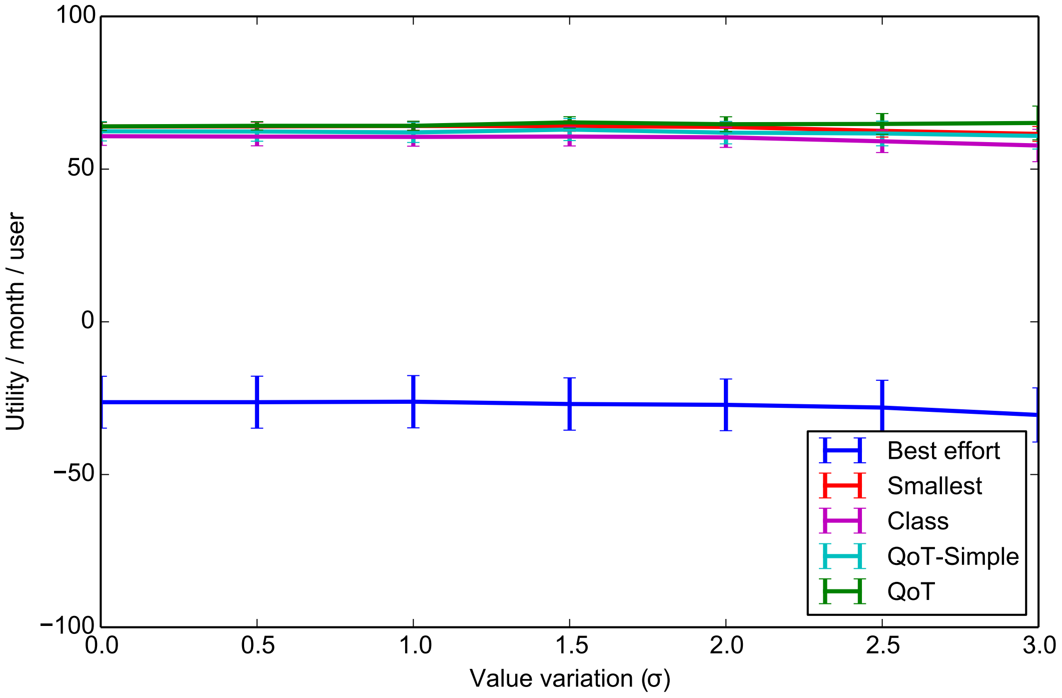


Figure 6.15: Variation of $v^d_{T,U}$.

Figure 6.15 shows the performance of QoT and the comparison schedulers, with the additional variation of utility parameters.

$v^d_{T,U} = 0$ means that there is no variation, which means that the utility value is deterministic, and only depends on the parameters of the transaction.

It can be seen that the resulting utility and the utility improvement with QoT do not differ beyond the confidence interval. With increasing variation, the confidence intervals become wider, because the variance of the utility values of transaction increases.

One could expect that QoT is able to increase its gain with increasing variance. With extreme variation of the expected utility of the transactions on the right hand side of Figure 6.15, a QoT scheduler could use the information about the transactions to cherry-pick a few transactions with exceptional high utility value. Assuming the original utility of each transaction is at a constant value, then there is a heavy-tailed characteristic of the logarithmic-normal distribution with $\sigma = 3$. In this case, only 0.22% of the transactions would carry 50% of the overall utility.

The QoT scheduler examined in Section 4.3.2 is not designed or configured to do such cherry picking. In such a cherry-picking strategy, arbitrarily high gains can be achieved under the following conditions:

- Configuring the QoT scheduler to do such cherry picking by setting $n_f \to \infty$ and $n_e = 1$.
- Reducing the prioritization for RTs and STs by using the total size of the transaction and utility values for the whole transaction.
- Network scenarios where the available bitrate is not sufficient, so that there is a huge gap until the maximum achievable utility sum.

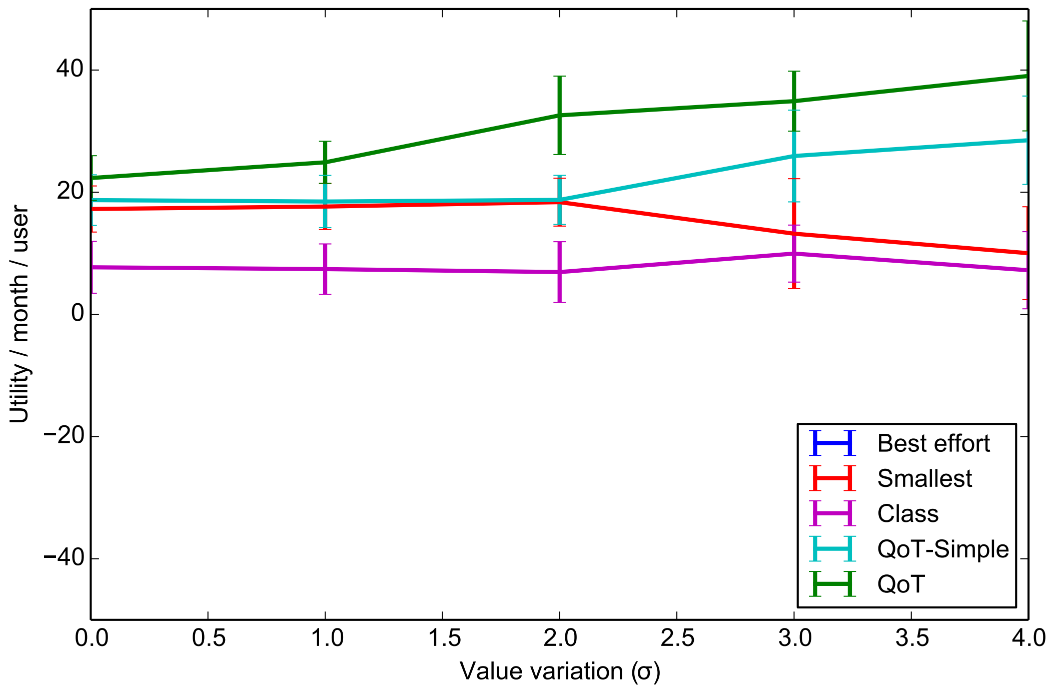


Figure 6.16: Variation of $v^d_{T,U}$ with modified QoT heuristic.

Figure 6.16 shows how the performance gain of QoT can become arbitrarily large with increasing variance of the utility values. The available bitrate of the access network is 10 Mbit/s, so that the gap until the maximum achievable utility sum is sufficiently large.

The QoT heuristic is parametrized $n_f = 10$ and $n_e = 1$. Such a configuration would probably not be useful in real networks, because the strict prioritization would lead to worse results in situations with many transactions of similar importance.

This shows however, that it is possible to improve the heuristic for QoT for given scenarios.

6.3.5 H-Factor

The asymmetry factor h_U introduced in Section 5.2.2 models how users perceive positive and negative experiences differently. It is defined as how many times one bad experience outweighs one good experience.

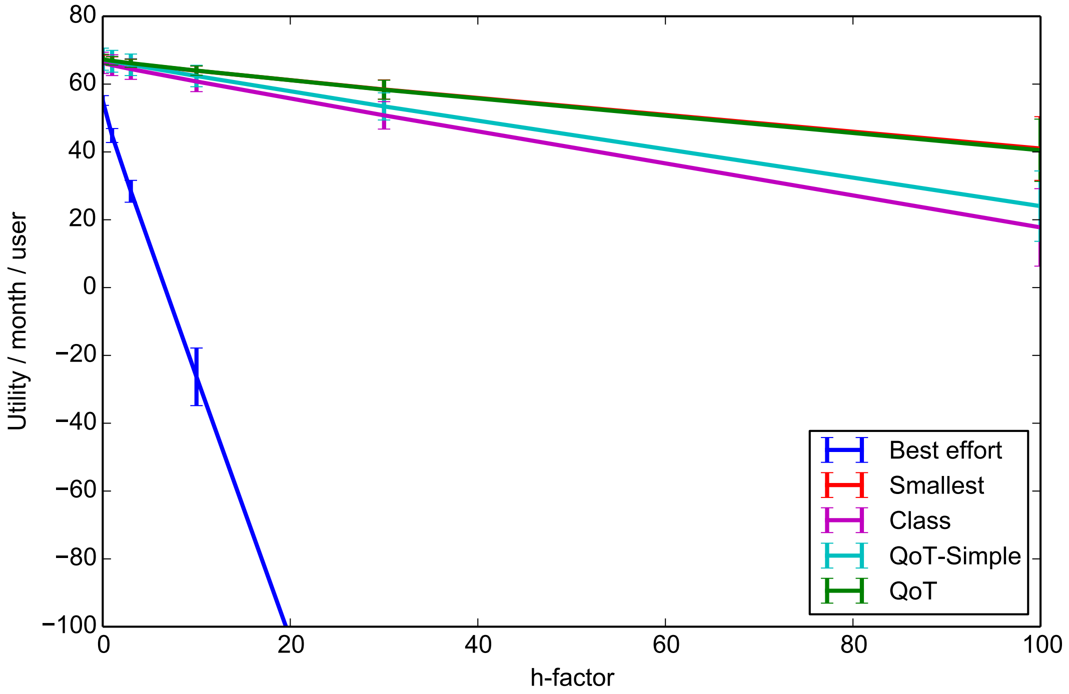


Figure 6.17: Total utility while varying h_U.

In Figure 6.17 the vertical axis shows the different asymmetry factors h_U; all other parameters are kept constant. As increasing h_U magnifies the impact of negative experiences, and the resulting sum of utility values will decrease. The results show that the resulting utility decreases proportionally to h_U for all schedulers. A linear function can be fitted to the resulting utility sums with a perfect coefficient of determination. For best effort, the regression line is $\overline{U_{\text{RR}}} = 53.2 - 6.91 h_U$ with ($R^2 = 99.99\%$) and for QoT, $\overline{U_{\text{QoT}}} = 66.1 - 0.23 h_U$ with ($R^2 = 99.91\%$).

All regression lines meet nearly in one point, at $h_U = -2.0$ with the utility sum $\overline{U} = 67.38 \pm 0.51$.

These results are important, because h_U is a parameter which is subjective, hard to determine, and probably unique for each person. These results show that the asymmetry factor h_U only scales the resulting utility sums; no other interdependencies between the schedulers and h_U is visible. This means that it is not necessary to determine h_U exactly to evaluate scheduling approaches with this utility model. However, utility values can therefore not be regarded as absolute values; instead, utility values are only meaningful as relative values for direct comparisons. This matches the definition of utility values as intuitive comparison metric, see Section 3.2.3.

6.4 Robustness of QoT under Degraded Transaction Knowledge

In this section, the behavior of QoT is analyzed, when the knowledge of the scheduler about the transactions is degraded. The goal of these evaluations is to evaluate the robustness of QoT against inaccurate transaction knowledge. This is important, because the users' requirements can never be obtained accurately. It is also important, because it will show that QoT does not require a high accuracy for the parameters of utility model. Instead, estimations are sufficient for QoT to achieve good results.

Degraded knowledge means that one or multiple parameters that influence the users' requirements and his experience of a transaction are partially unknown to the scheduler. The concept and the distribution of the degradation are introduced in Section 5.2.6. The influence of degraded knowledge about the utility values of transactions is analyzed in Section 6.4.1. The influence of degraded knowledge about the urgency and timing requirements of transactions is analyzed in Section 6.4.2.

The knowledge of the scheduler about the size of each transaction can also be erroneous, which is introduced in Section 5.3.4. The impact of these types of degraded knowledge is evaluated in Section 6.4.3.

As introduced in Section 5.2.6 and Section 5.3.4, the error values are from a logarithmic normal distribution with either $\mu = 0$, or with a mean value 1. The relevant distribution parameter σ determines how far the resulting value differs from the original value. Random values from a logarithmic-normal distribution are multiplicatively added to the original value from the utility model. Again, a variation with $\sigma = 1$ means that for around 10% of the cases, the resulting value is smaller or bigger than the original value by a factor of 5 or more. A variation with $\sigma = 3$ means that in around 10% of the cases, the resulting value is smaller or bigger than the original value by a factor of at least 140. In those cases where the result is larger than the original value, on average, the result is larger by a factor of 180.

6.4.1 Degraded Utility Values

To analyze how QoT is affected by degraded knowledge about the requirements of the user's transactions, the first way is to alter the utility values that the scheduler uses for the scheduling decisions. This will also determine the accuracy required for determining the utility values.

The degradation of utility values in the simulations is achieved by multiplying the utility values the scheduler uses with the factor $e^{d}_{T,U}$, as introduced in Sections 5.2.7 to 5.2.9.

Figure 6.18 shows the utility sum with degraded the utility values available at the scheduler. On the left side of the plot, the utility values are not degraded ($\sigma = 0$), on the right side of the plot, the utility values available at the scheduler are degraded on average by more than two

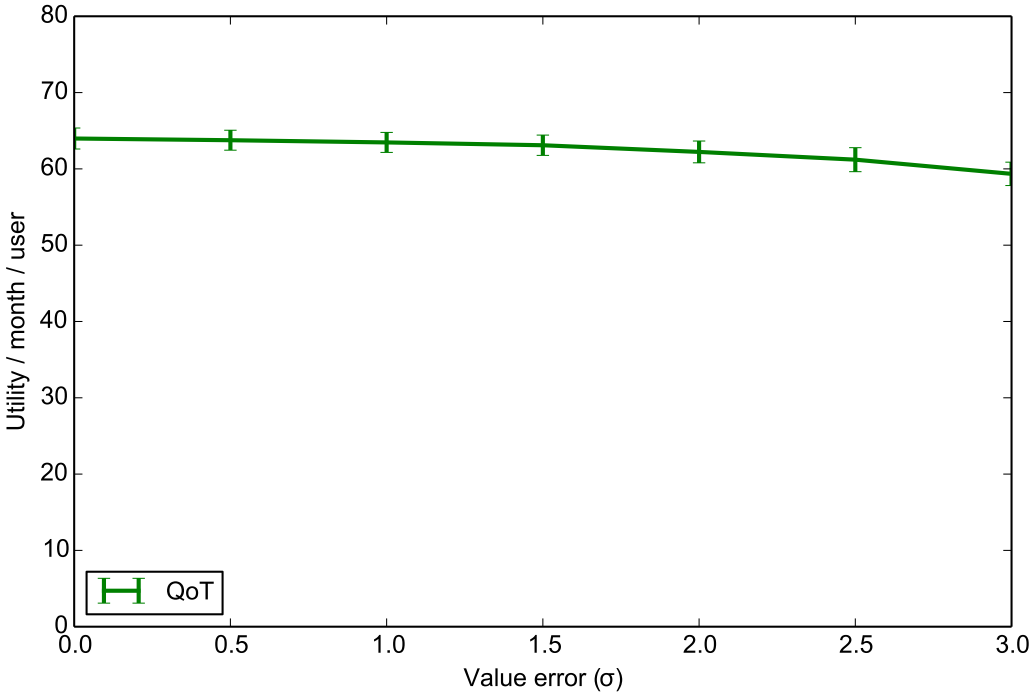


Figure 6.18: Performance of QoT with degraded utility values.

magnitudes ($\sigma = 3$).

The QoT heuristic is the only scheduler that uses this utility value for the scheduling decision. Therefore, only the QoT heuristic is affected by this degradation.

Even with an average error of more than two magnitudes ($\sigma = 3$), QoT is only affected slightly. This shows that the utility values require only low accuracy. This is important, because it is difficult to obtain accurate values from the user, and there are no established utility models for these scenarios, which were already validated to match reality.

6.4.2 Degraded Urgency Information

The second way to analyze how QoT is affected by degraded knowledge about the requirements of the user's transactions is to alter the time values in the formulation of the utility functions. This influences how urgent a transaction seems to be for the scheduler.

Degradation of time values in the simulations is achieved by multiplying all time values the scheduler uses with the factor $e^{d}_{T,t}$, as introduced in Section 5.2.7 to Section 5.2.9.

Figure 6.19 shows the utility sum with the degraded utility values available at the scheduler. On the left side of the plot, the time values are not degraded ($\sigma = 0$), on the right side of the plot, the time values available at the scheduler are degraded on average by more than two magnitudes ($\sigma = 3$).

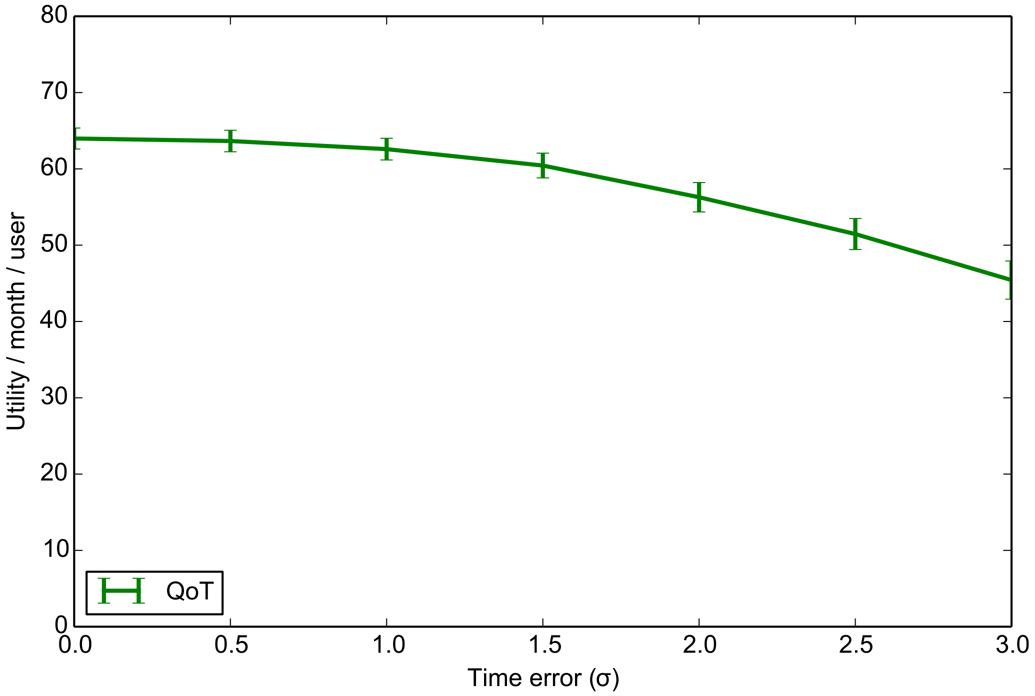


Figure 6.19: Performance of QoT with degraded urgency values

The QoT heuristic is the only scheduler that uses these time values for the scheduling decision. Therefore, other schedulers are again not affected by this degradation.

Even with an average error of more than two magnitudes ($\sigma = 3$), QoT is only affected moderately. The decrease in overall utility is however stronger than with the degraded utility values in the previous Section 6.4.1.

This shows that time information in the utility functions require only low accuracy. A rough estimation of the urgency of a transaction is sufficient.

Analogous to the degradation of utility values, this robustness is important because it is also difficult to obtain accurate urgency information from the user.

6.4.3 Degraded Size Information

In this section, the influence of degraded information about the size of the transactions is analyzed.

The degradation of size information is achieved by multiplying the size information that the scheduler uses with the factor $e^d_{T,s}$, as introduced in Section 5.3.4.

The QoT heuristic, the *Size-and-Class Heuristic* (simple QoT), and the *Smallest-Size Scheduler*, use the estimated size of a transaction for the scheduling decision.

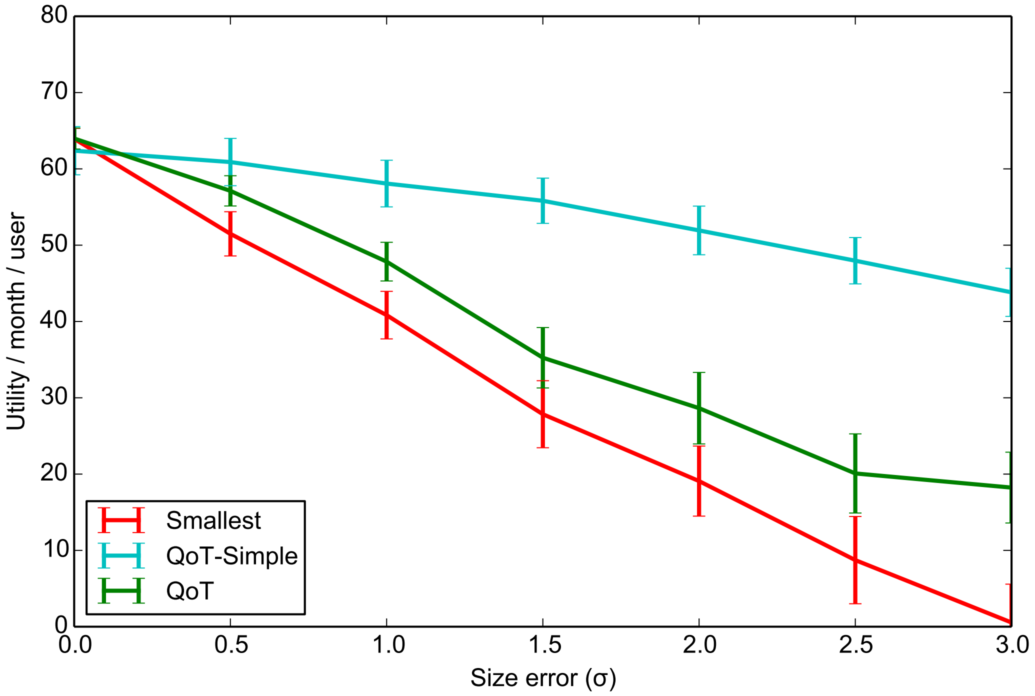


Figure 6.20: Performance of QoT with degraded size values.

Figure 6.20 shows how these three scheduling heuristics perform with degraded size information. The *Size-and-Class Heuristic* primarily uses the traffic class as a key for the scheduling decision. In this case, the size of a transaction is only used as a secondary key for scheduling multiple transactions of the same traffic class. The impact of degradation of size information on the *Size-and-Class Heuristic* is therefore minimal. The *Smallest-Size Scheduler* relies only on the size of transactions as a key for scheduling decisions. Therefore, there is a strong impact of this degradation on the resulting utility. The results for the QoT heuristic are in between these two.

6.5 Summary

In this chapter, the QoT approach is analyzed using an event-driven network simulation. The traffic model from the previous Chapter 5 is used to generate traffic. QoT and the comparison schedulers are used on the access network.

In Section 6.2 different network topologies and network properties are used to analyze how the schedulers perform in different scenarios. These evaluations show that all QoS approaches increase the utility metric significantly over a best-effort approach, and that QoT achieves the highest utility values.

Section 6.3 illustrates that QoT is robust against changes in traffic characteristics. Beyond the varying parameters of the generated traffic, such as the average bitrate each user requests, different variants of the traffic models and the utility models are used. Again, all QoS approaches increase the utility metric significantly over a best-effort approach, and in most of the cases, QoT achieves the highest utility values.

Finally, Section 6.4 shows that QoT is robust against uncertainties in the measurement and the quantification of the metrics. These evaluations show that utility values and urgency information do not need to be accurate for QoT to achieve good results. Even with an average error of more than two magnitudes ($\sigma = 3$), QoT is only affected slightly. This robustness is important, because it is difficult to obtain accurate values from the user and there are no established utility models for these scenarios, which were already validated to match reality.

In general, all QoS approaches increase the utility metric significantly over the best effort approach. All of the evaluated QoS approaches use information about the traffic on both sides of the bottleneck. Bringing information about the traffic from the subscriber to the access network headend on the other side of the bottleneck is one of the key ideas of QoT. These results, therefore, show the usefulness of this QoT feature.

6.6 Outlook

There are more questions regarding the performance of QoT that are worth studying, but are beyond the scope of this work.

The evaluations in this work use only scenarios where all subscribers that share the same access medium request and transmit in average the same amount of traffic. It would be interesting to analyze how QoT behaves when the subscribers on one access network have different traffic characteristics. Such studies require models for what desired outcomes are in such scenarios, and accordingly, the metrics that allow evaluating the results.

It would also be interesting to see how QoT would compare to other QoS approaches; comparisons with existing QoS approaches such as DiffServ or IntServ need to make assumptions on how widespread the support for these approaches might be in the Internet.

A second class of approaches that would be interesting to compare to QoT with are scheduling approaches based on congestion control algorithms. These approaches also do not require support on all intermediate nodes throughout the Internet, and are therefore more likely to reach a critical mass of deployment. One example of such an approach would be an end-to-end congestion control algorithm between the client and the corresponding node, which is able to determine parameters of other traffic that uses the same network resources, or shares the same bottleneck. These approaches estimate the parameters of parallel traffic on the bottleneck by observing characteristics, such as the delay of packets between this client and the respective corresponding node. It is difficult to compare QoT with such an approach, as it is difficult to predict the accuracy of these estimations, because there is no upper bound for these accuracies. By increasingly using a-priori knowledge and assumptions on the parallel traffic in a certain scenario, the quality of these estimations could be progressively improved.

7 Summary, Conclusion and Outlook

In this work, a new approach is developed for improving the service quality for Internet access subscribers. Its design choices are explained, and its performance is evaluated using event-driven network simulation. The new approach, Quality of Transaction (QoT), tries to avoid the problems that prevented greater acceptance of the existing Quality of Service (QoS) approaches, standards, and products.

The first key point of QoT is to focus on improving the scheduling on the access link, because in today's Internet access, the access medium ("last mile") is the bottleneck. As a result, it is significantly easier to deploy QoT than deploying a system that tries to improve the whole Internet. For QoT, only the operator of an access network and the access subscribers need to deploy components. As the access network operator is already in a contractual relationship with his customers, he can provide his customers with QoT enabled devices, firmware images, or applications.

The second key point of QoT is to obtain all context or knowledge about the service requirements from the user, his devices, the operating systems, and the applications on his devices, because the service quality of the access network is experienced and defined by the user. As a result, the user decides which traffic is prioritized by the access network. This is important in the context of the current debate on *Net neutrality*. The scheduling for the downlink direction in the access link is performed by a device of the operator, the headend. Therefore, the users' service requirements need to be signaled to this headend.

The first fundamental concept of the QoT approach is the novel idea of network **transactions**. Such a transaction represents a result of network transmission from a user's view, for example, a website that is displayed. At the same time, transactions are defined from a machine's view as a set of sections from transport layer connections or other transmitted data streams. The users' service requirements are formulated and signaled to the headend, referring to the transactions, while the attribution in existing QoS approaches typically refers to packets or transport layer connections.

The second fundamental concept of QoT is the formulation of the service requirements as time dependent **utility functions**. The end users' applications define the requirements for each transaction, depending on the type of result as a function. The values of this function specify the value of the transaction for the user. The resulting value therefore depends on the scheduling of this transaction. This allows the scheduler to optimize the transmissions based on the current load and network state.

Based on these concepts, an architecture is developed. This architecture specifies the functionality of the required components, and the information that needs to be exchanged between these.

The performance of QoT is evaluated using event-driven network simulation. A system model for Internet access has been developed, along with a new traffic model that generates a traffic mix for Internet access subscribers. This traffic mix contains multiple traffic classes with different behavior and characteristics. The main feature of this traffic model is to generate traffic with transactions and requirements that reflect the application behavior of the corresponding traffic class.

These evaluations show that QoT provides significant improvements in the service quality experienced in comparison to the currently prevailing best effort and fair scheduling disciplines. Analyzing the contribution of the different traffic classes for this improvement, demonstrates that for a wide variety of traffic mixes and network scenarios, an access network using QoT can increase the transmitted bitrate by 60% to 150% while experiencing the same level of service quality.

QoT is also compared to other QoS approaches that transfer some transaction information from the subscriber to the access network headend. These comparison schedulers perform the scheduling on the same nodes that QoT does, but use less transaction information, for example, only the size or only the type of traffic of each transaction. The differences in the utility values between QoT and these comparison schedulers are smaller than between QoT and the best-effort approach. In most cases, QoT shows significantly higher utility values, or allows significantly higher bitrates at the same utility value.

These evaluations show that a simplified procedure that uses the QoT transactions and uses simple metrics instead of utility functions, can obtain nearly the same performance as QoT using utility functions as requirements specifications. Since these metrics are scalar values, this reduced approach would be significantly less complex to implement.

Analyzing the robustness of QoT against any missing or degraded information in the transaction's size or requirements shows that QoT does not depend on the exactness of these values. Even when these values are smaller or larger than the real value by a factor of two, the performance of QoT is only imperceptibly reduced. In these cases, defining and signaling the utility functions as requirements proves beneficial.

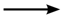
Beyond these evaluations, there are further aspects of QoT that could be evaluated, and further comparisons are possible, as listed in Section 6.6. As mentioned in Section 1.4, the psychological questions for such a system have to be answered by a running prototype, as there is an inherent feedback loop between the user's actions and the systems behavior. With psychophysical studies, the metrics and definition of utilities can be adjusted so that the systems metrics match the user's perception as closely as possible.

It is therefore required, as a next step for QoT, to implement a larger prototype on a real access network. This prototype has to include a sufficient number of adjusted applications to be used in these further studies. In a wider deployment of QoT, the signalization protocol has to be standardized.

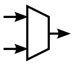
Appendices

A Data Flow Notation

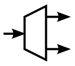
The following symbols are used to depict the dataflow in networks or components.



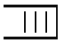
Physical or logical connection to allow data flow.



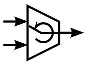
Combining multiple data flows into one data flow, e.g. a multiplexer.



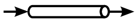
Splitting one data flows into multiple data flows, e.g. a demultiplexer.



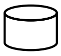
Queue: Stores elements of a data flow until the connected physical or logical connection becomes idle. The queue generally operates in First In – First Out (FIFO) mode. Unless noted otherwise, the size of the queue is infinite.



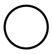
Scheduler: Combining data flows from several queues into one data flow.



Physical line with a constant bitrate and a fixed propagation delay.



Database.



Processor: Process elements of a data flow. The processing time is specified. The processor can only process one element at a time.

B Traffic Model Characteristics

This appendix shows characteristics of the traffic model, introduced in Chapter 5.

Figure B.1, Figure B.2, and Figure B.3 show the distribution of the sizes of all transactions and all traffic chunks generated by the traffic model. While Figure B.2 shows the distribution of transaction and traffic chunk sizes in the downlink direction, Figure B.3 shows this for the uplink direction and Figure B.1 for both directions combined.

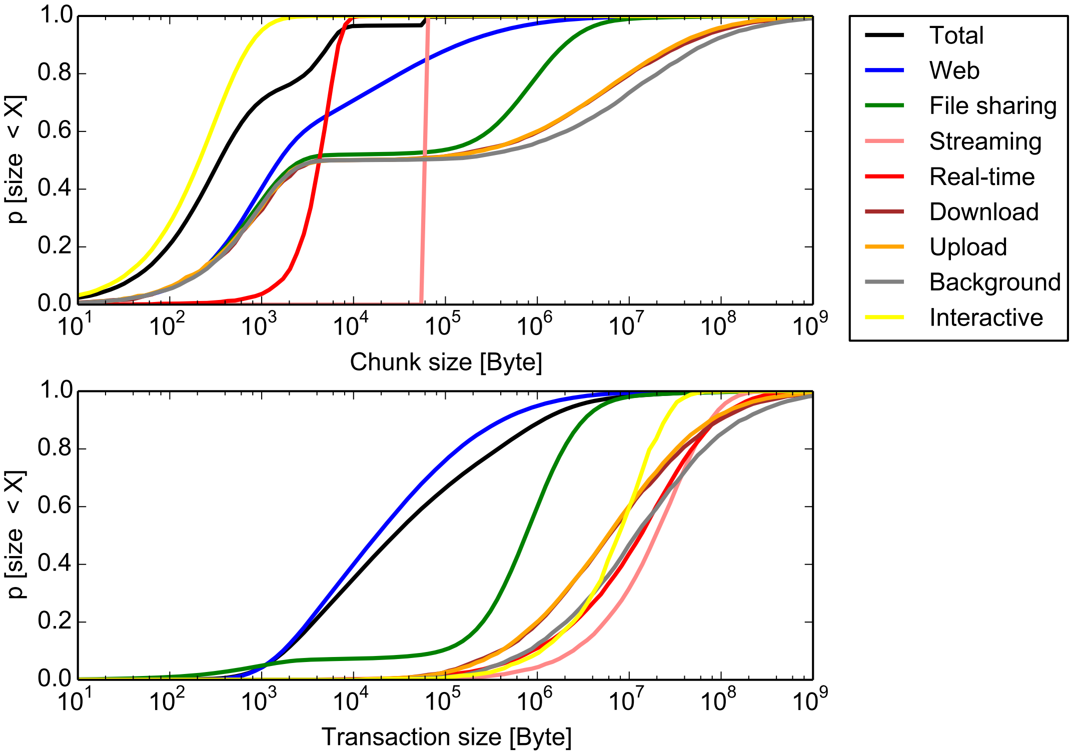


Figure B.1: Distribution of chunk and transaction sizes of the traffic model

The colors of each curve codes the corresponding application class model, while the black curve is the combination of all transactions or of all traffic chunks, regardless of its application class model. Since the horizontal axis is scaled logarithmically, the dominant log-normal Distribution of each application class model can be seen in each of these curves. The step function for the size of traffic chunks of the application class model *Streaming* comes from its fixed 64 KB chunk size.

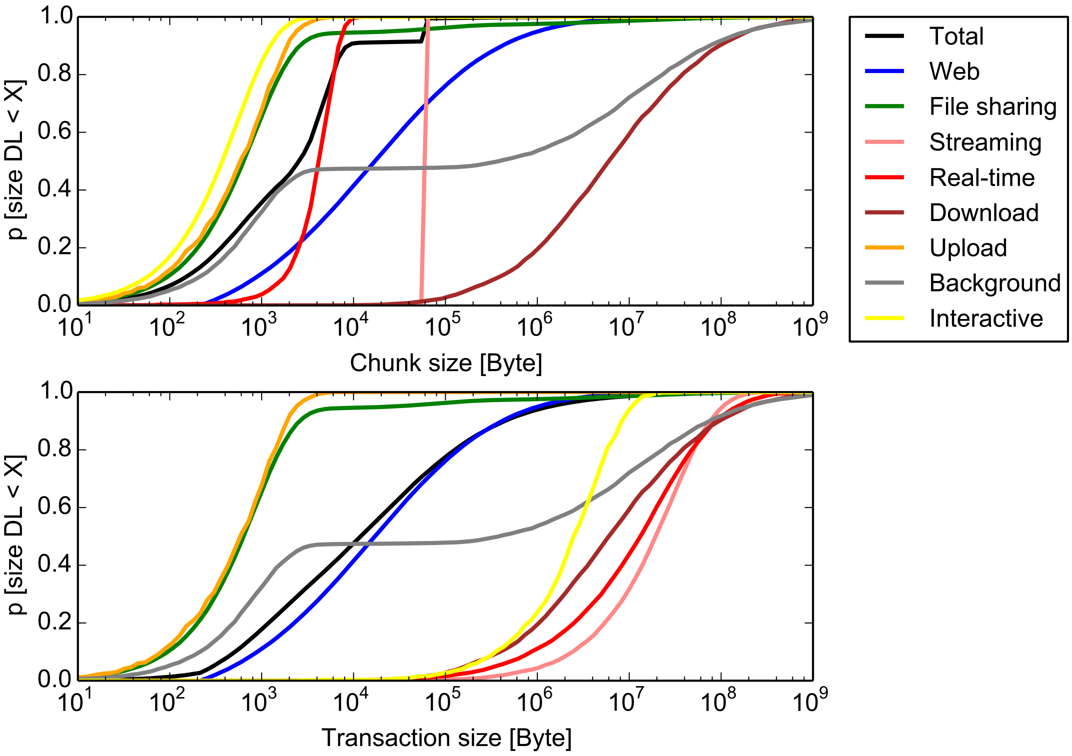


Figure B.2: Distribution of chunk and transaction sizes for downlink

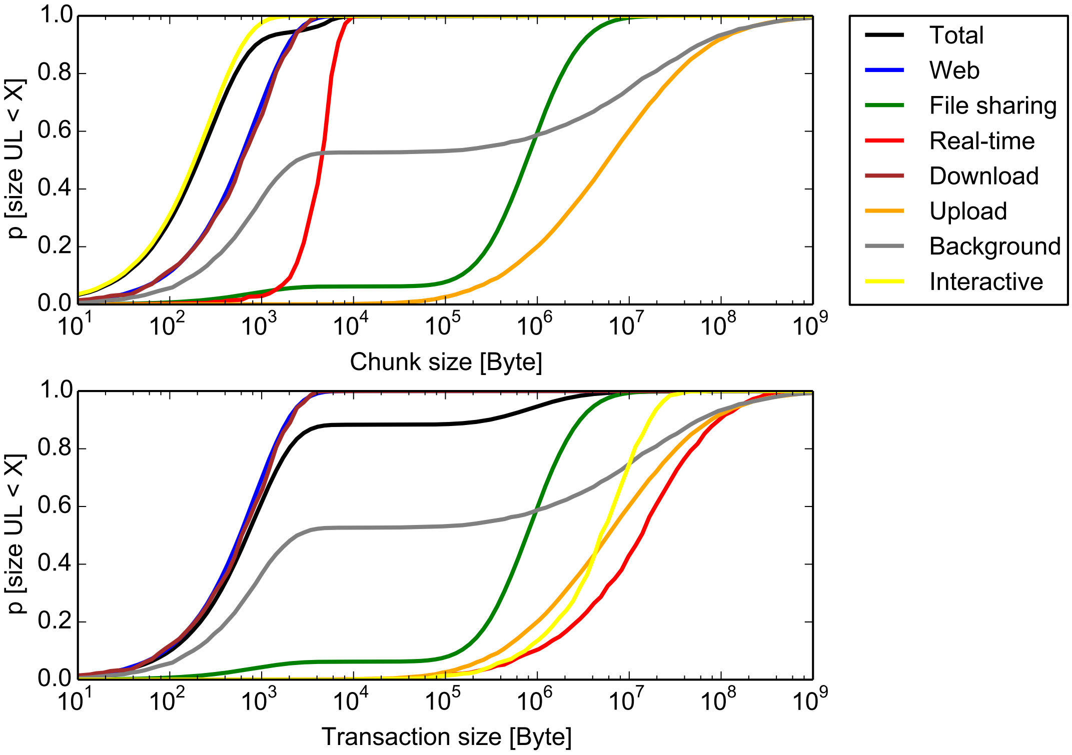


Figure B.3: Distribution of chunk and transaction sizes for uplink

In Figure B.4 the distribution of the size of traffic chunks and transactions for the different traffic model variants can be seen. As expected model variant *BigObjects* produces larger chunks and transaction whereas model variant *SmallObjects* produces traffic with smaller chunks and transactions. The transaction size distribution of the other model variants lie between these two extremes.

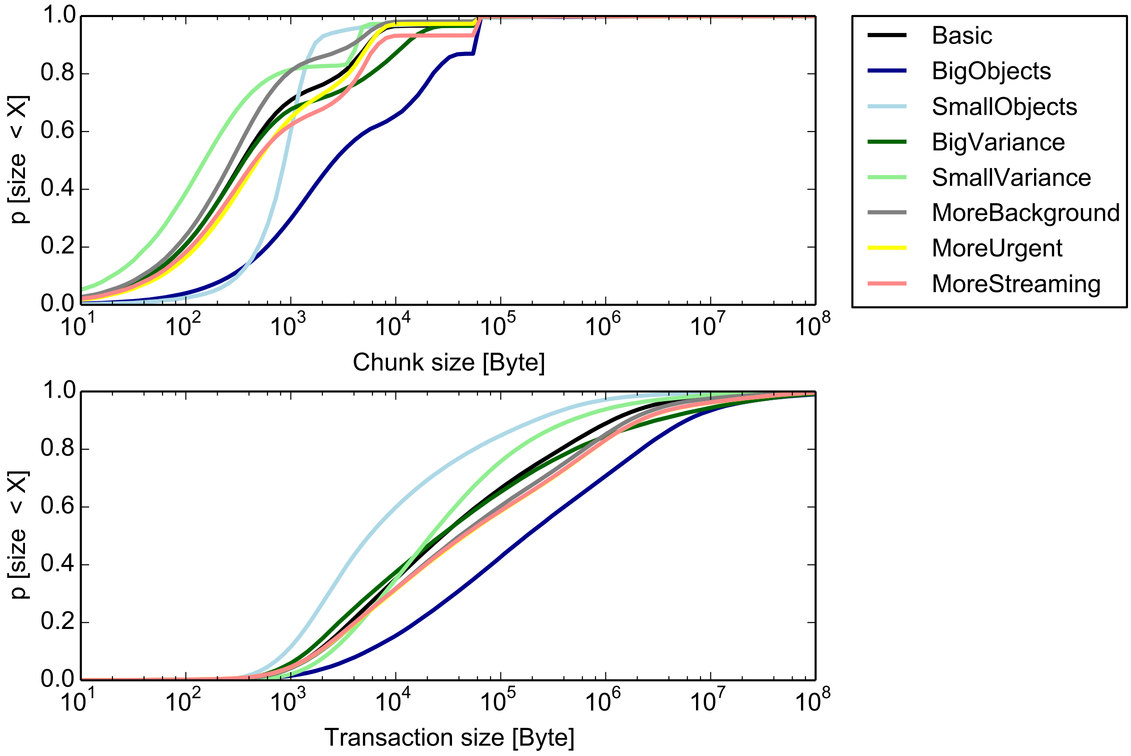


Figure B.4: CDF of chunk and transaction sizes for all model variants

The sizes in such traffic models varies over several orders of magnitude. An important fact to consider with such traffic models is therefore, that a few large transactions account for more data volume than a majority of small transactions. The following Figures therefore show which proportion of the total data volume of the traffic model and of each application class model are caused by traffic chunks or transactions up to a given size. Figure B.6 shows this accumulated data volume for traffic chunks and transactions in downlink direction, Figure B.7 in uplink direction and Figure B.5 for the combination of both directions.

Figure B.8 shows the autocorrelation of the queued transactions in the buffer in downlink direction for two bitrates of the links in the model. The upper plot shows a link bitrate of 50 Mbit/s, which means that the generated traffic is not influenced by the access network, because the available bitrate is always sufficient. In this scenario, this autocorrelation function depends mainly on the characteristics of the traffic model. The queue in the headend is typically empty or contains the single transaction that is transmitted at this time. The average number of transactions in the queue is significantly below 1 (see Table B.1). Therefore, there are significant

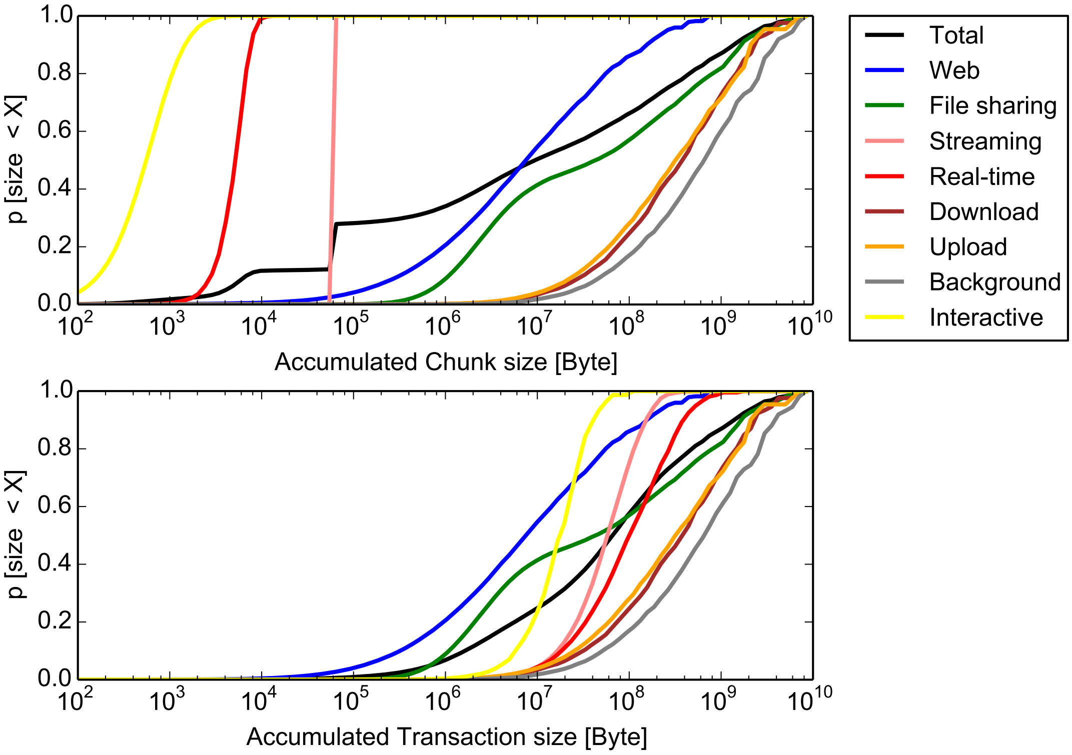


Figure B.5: Amount of traffic in chunks and transactions of below a given size

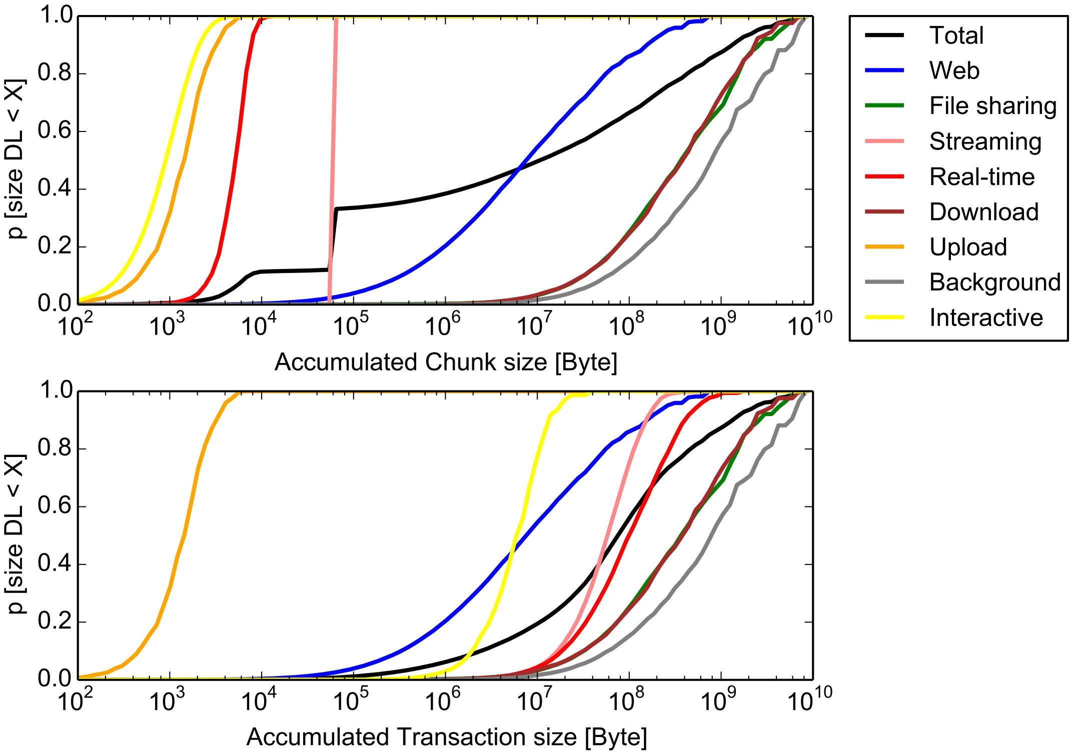


Figure B.6: Amount of downlink traffic in chunks and transactions of below a given size

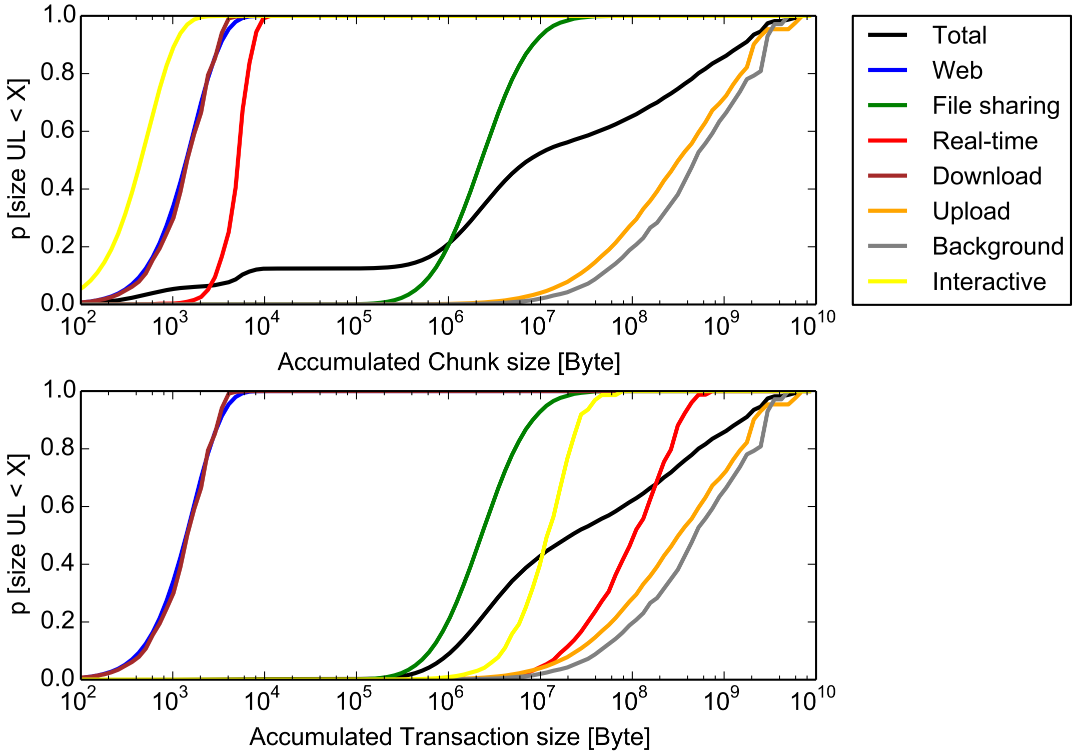


Figure B.7: Amount of uplink traffic in chunks and transactions of below a given size

differences between the traffic model variants. These autocorrelation functions decay faster for those traffic model variants that generate a large number of tiny transactions.

Table B.1: Number of queued transactions

Model Variant	50 Mbit/s	10 Mbit/s
Basic	0.341(±0.031)	5.037(±0.407)
BigObjects	0.498(±0.124)	4.126(±0.824)
SmallObjects	0.290(±0.020)	6.018(±0.372)
BigVariance	0.329(±0.096)	2.871(±0.374)
SmallVariance	0.593(±0.037)	10.290(±0.885)
MoreBackground	0.354(±0.045)	4.401(±0.422)
MoreUrgent	0.285(±0.021)	5.191(±0.523)
MoreStreaming	0.287(±0.024)	5.532(±0.616)

The plot on the bottom of Figure B.8 shows the same autocorrelation functions for an available bitrate of 10 Mbit/s. In this scenario there are influences from the network to the generated traffic, because the available bitrate is often not sufficient. This means that the queue in the headend typically contains multiple transactions (see Table B.1). This queuing of transactions because of insufficient available bitrate is the dominant influence in this scenario. Therefore the

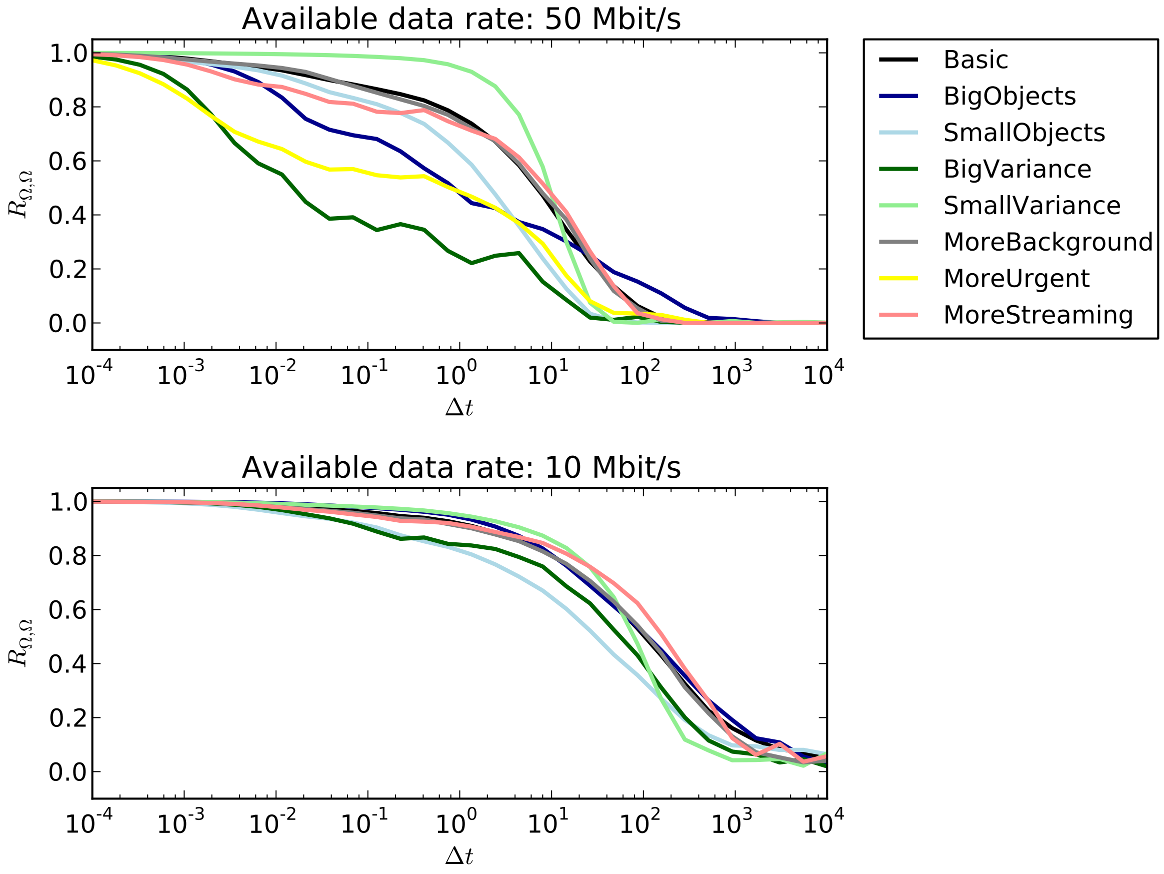


Figure B.8: Autocorrelation of queued transactions

difference between the traffic model variants is less pronounced in this case. The influences of this queuing can also be seen in the shifting of the curves to the right, because queuing is an integrating operation.

In Table B.1 the mean queue length, as well as the 95% confidence intervals, of the downlink buffer can be seen. For configured bitrate of 50 Mbit/s the mean queue length for all model variants is smaller than one. Only for a bitrate of 10 Mbit/s the mean queue length increases to values greater than one.

The utility values generated by the traffic model are analyzed in Figure B.9. This plot shows the utility contribution for all transactions smaller than the size given on the horizontal axis. The traffic class models are depicted by different colors. The total utility of each traffic class model is scaled to 1.

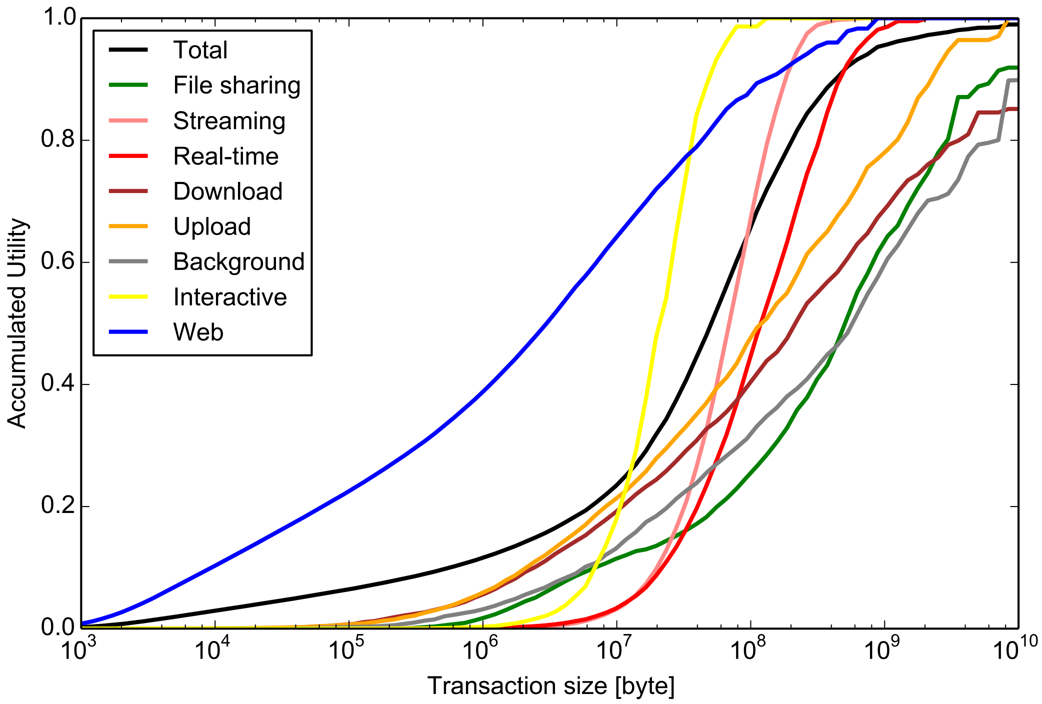


Figure B.9: Contribution to total utility of transactions smaller than the given size, for each traffic class model.

C Traffic Variants

This appendix is an extension to Section 6.3.2. The following Figures C.1 to C.9 show the simulation results for all combinations of traffic model variant and evaluated scheduler.

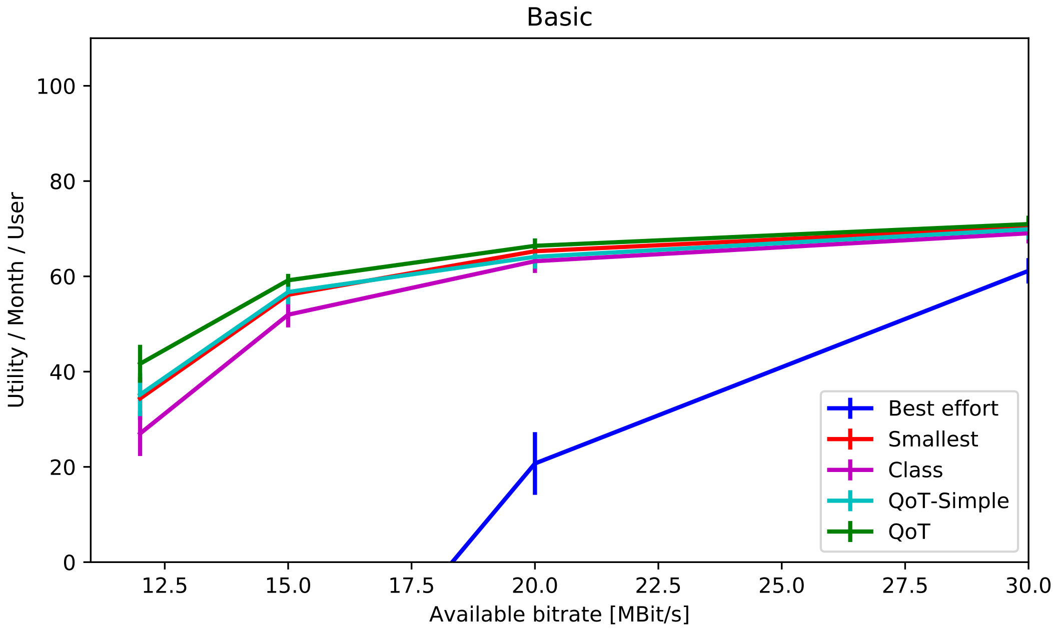


Figure C.1: Sum utility for traffic model variant *Basic*.

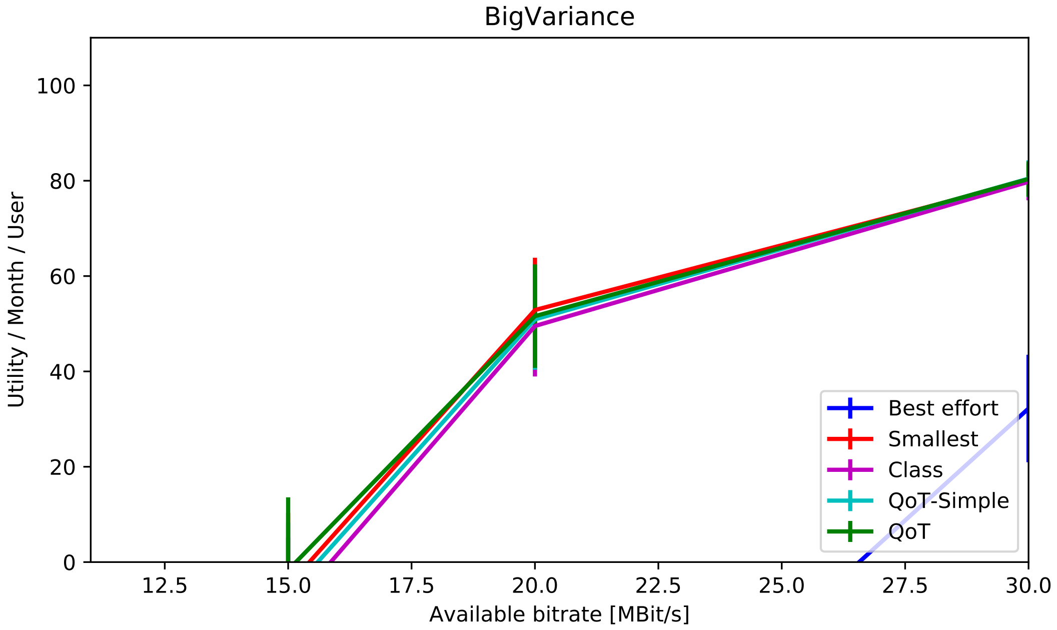


Figure C.2: Sum utility for traffic model variant *BigVariance*.

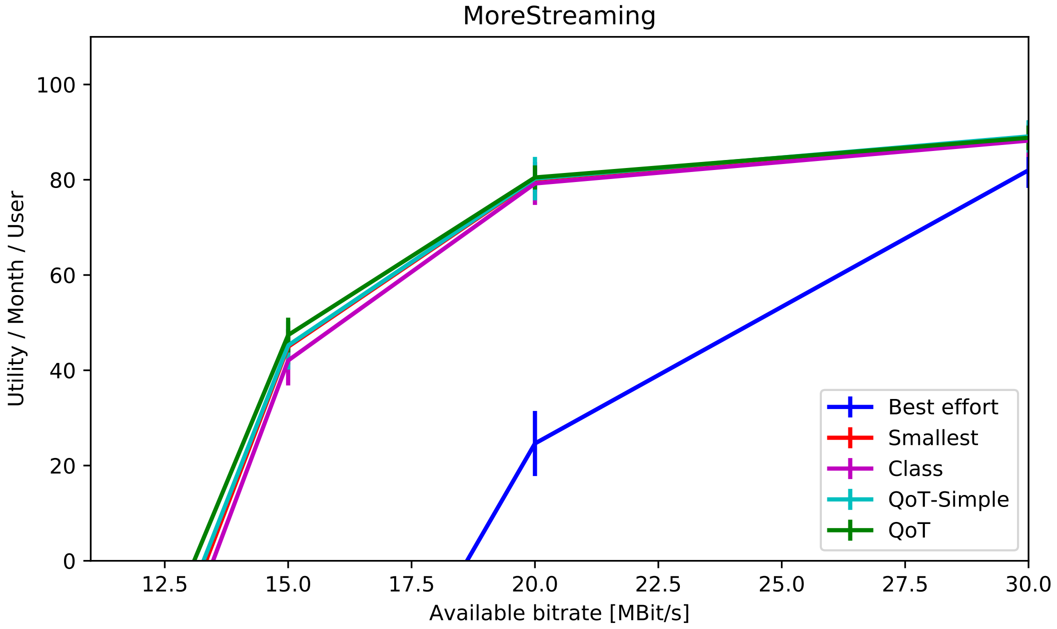


Figure C.3: Sum utility for traffic model variant *MoreStreaming*.

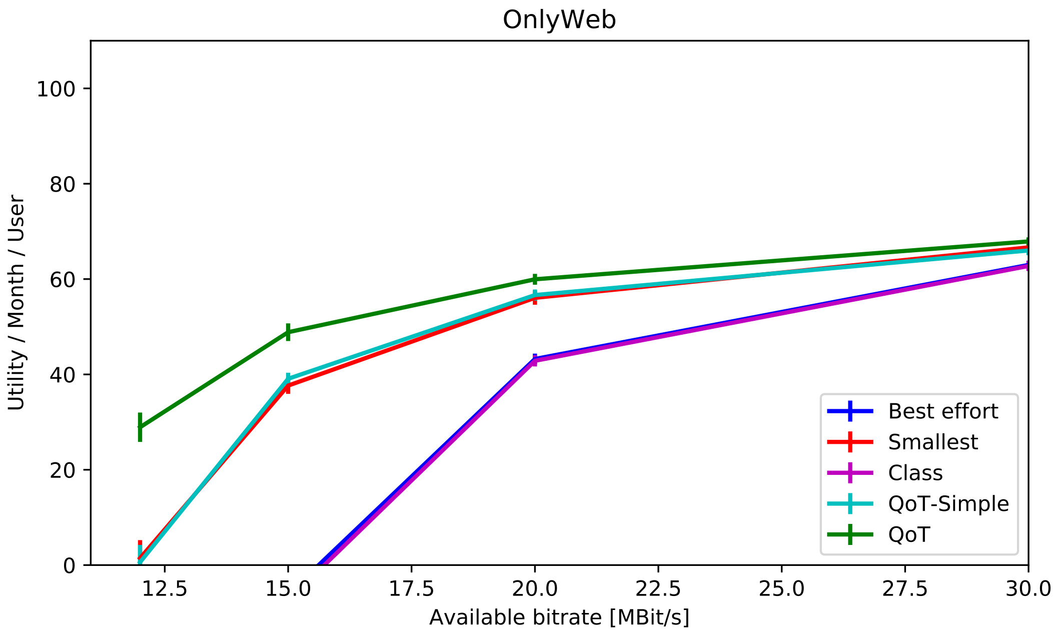


Figure C.4: Sum utility for traffic model variant *OnlyWeb*.

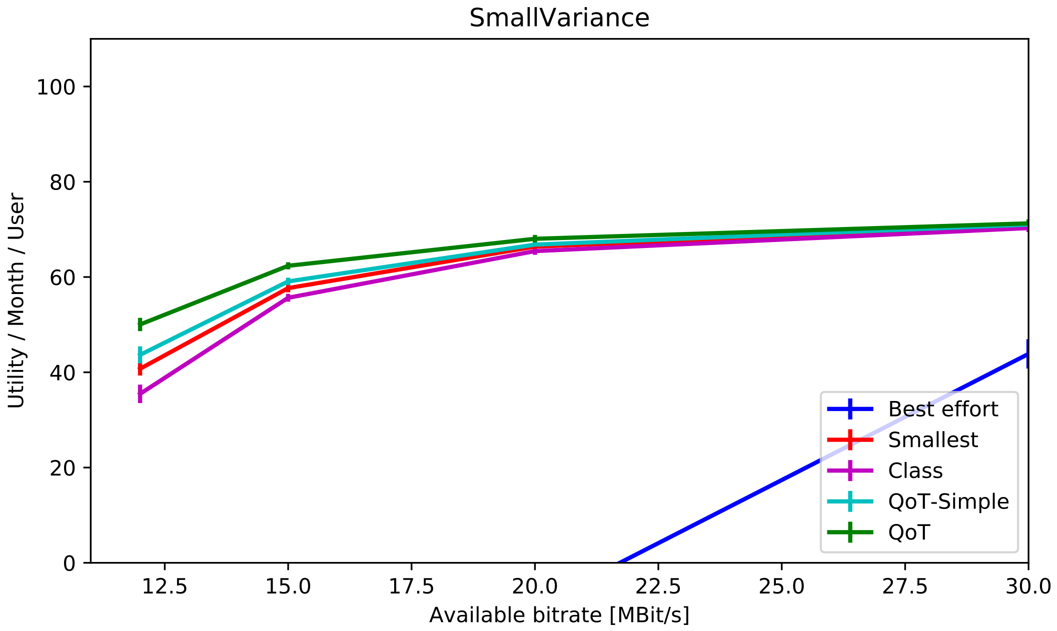


Figure C.5: Sum utility for traffic model variant *SmallVariance*.

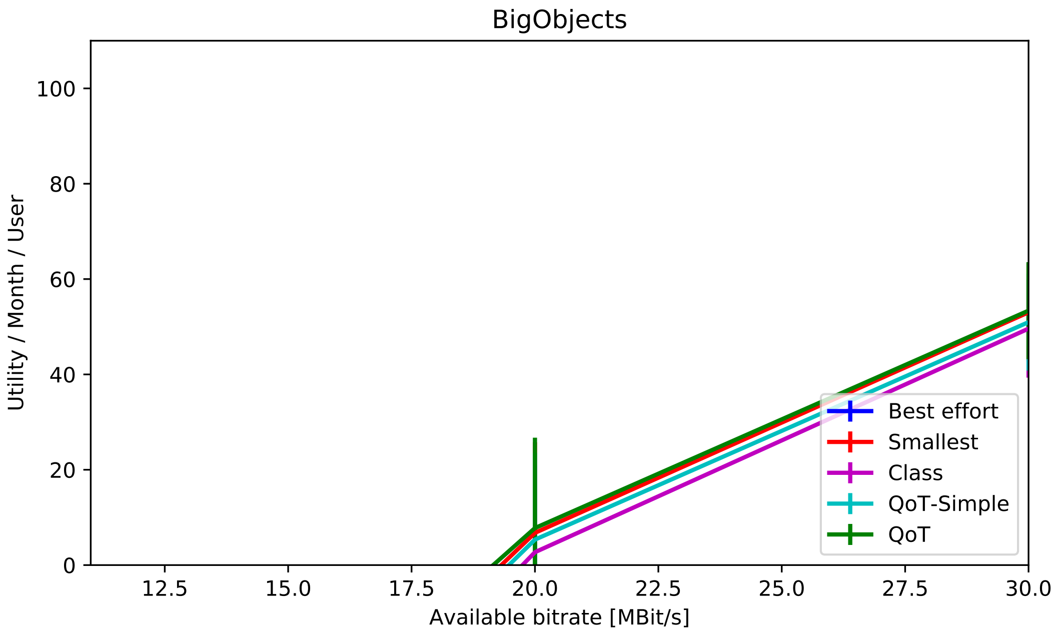


Figure C.6: Sum utility for traffic model variant *BigObjects*.

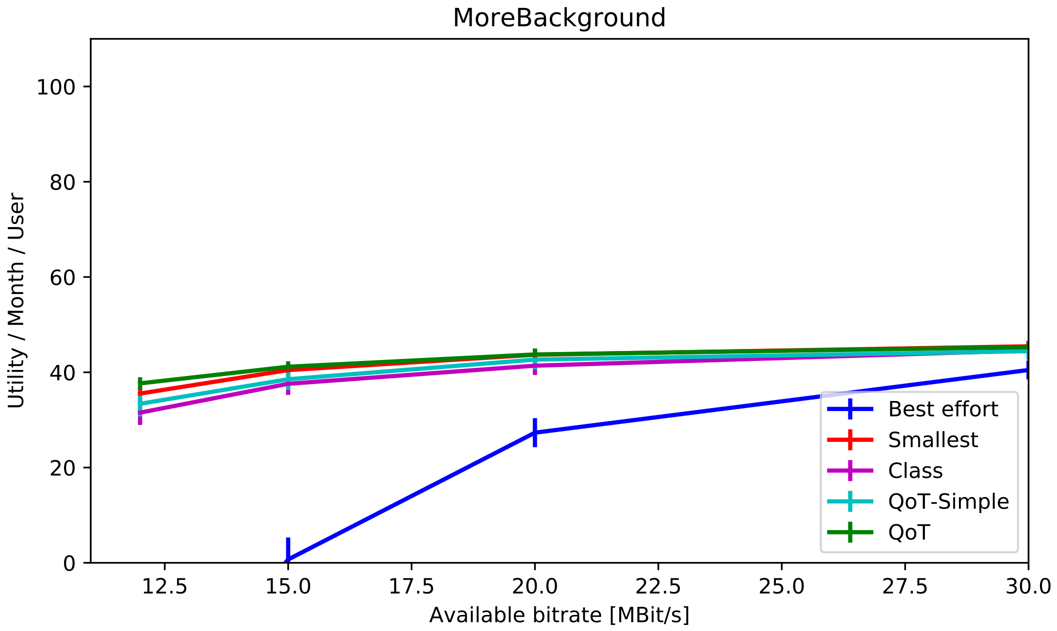


Figure C.7: Sum utility for traffic model variant *MoreBackground*.

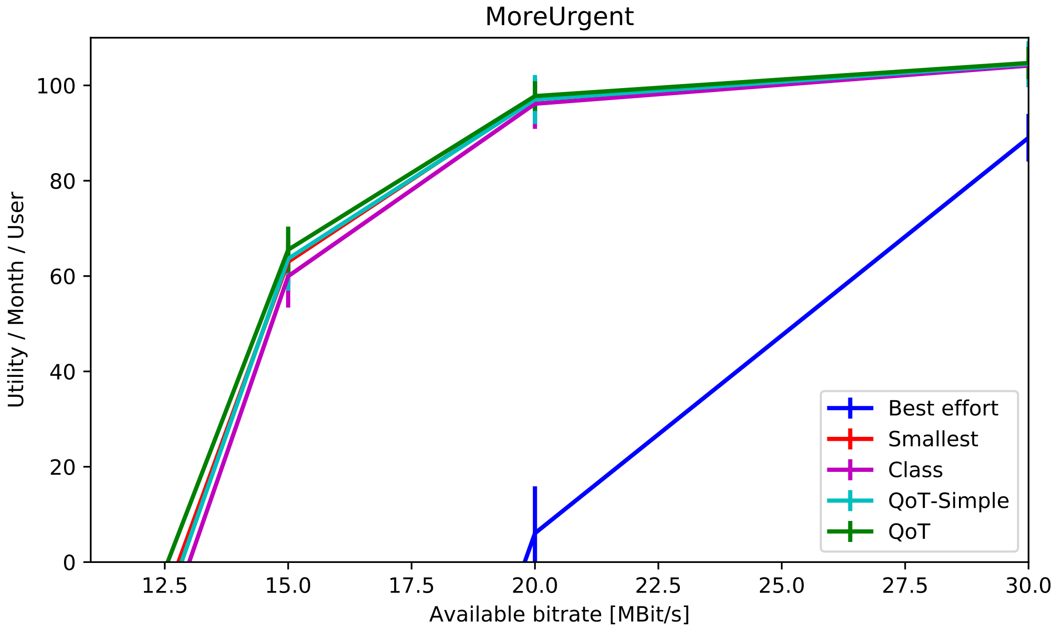


Figure C.8: Sum utility for traffic model variant *MoreUrgent*.

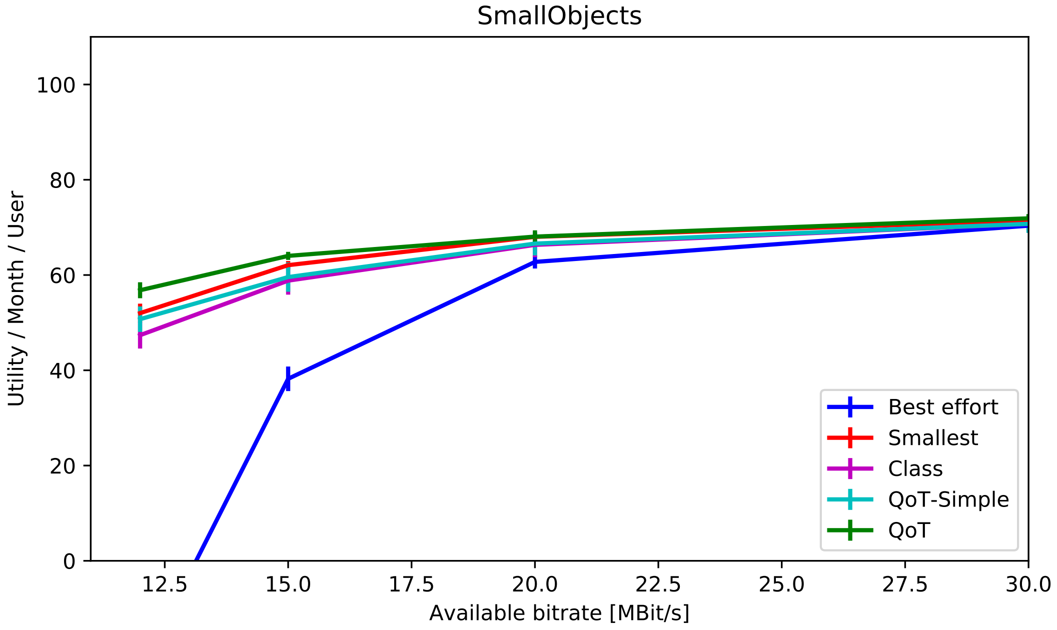


Figure C.9: Sum utility for traffic model variant *SmallObjects*.

D Signalization Protocol

Below is a proposed protocol to signal transaction information. This protocol is plain text and describes the content of a User Datagram Protocol (UDP) datagram using Extended Backus–Naur Form (EBNF) notation.

It is possible, to convert this text protocol to a binary protocol, e.g. using Abstract Syntax Notation One (ASN.1).

```
⟨datagram⟩ = { ⟨transaction⟩ ⟨requirement⟩ };

⟨transaction⟩ = ⟨transaction-id⟩ { ⟨chunk⟩ } :;

⟨transaction-id⟩ = INTEGER;

⟨chunk-id⟩ = INTEGER;

⟨chunk⟩ = { ⟨chunk-id⟩ ⟨filter⟩ [ PART ⟨offset⟩ ⟨length⟩ ] ⟨size-estimate⟩ } ,;

⟨filter⟩ = ⟨direction⟩ ( SOCKET ⟨Socket⟩
  | 5TUPEL ⟨IP-Protokoll⟩ ⟨Src-IP⟩ ⟨Dst-IP⟩ ⟨Src-Port⟩ ⟨Dst-Port⟩
  | LABEL ⟨flow-label⟩
  | 3GPP ⟨bearer⟩
  | ATM ⟨VCI⟩
  | SCTP ⟨stream⟩
  | FILE ⟨local PID⟩ ⟨FD⟩ );

⟨direction⟩ = up
  | down;

⟨length⟩ = INTEGER;

⟨size-estimate⟩ = INTEGER;

⟨offset⟩ = INTEGER;

⟨requirement⟩ = ⟨start-time⟩ { ⟨transaction-ID⟩ } : ⟨utility⟩;
```

⟨*utility*⟩ = RT ⟨*rt-utility*⟩
  | FT ⟨*st-utility*⟩ |
  | ST ⟨*rt-utility*⟩;

⟨*rt-utility*⟩ = (LINEAR ⟨*Start*⟩ ⟨*Regression*⟩
  | SCURVE ⟨*tExp*⟩ ⟨*tDrop*⟩ ⟨*Uexp*⟩ ⟨*Udrop*⟩
  | SPARAM ⟨*scale*⟩ ⟨*Udrop*⟩ ⟨*Timeout*⟩ ⟨*Slope*⟩
  | PIECEWICE { ⟨*t*⟩ ⟨*U*⟩ });

⟨*st-utility*⟩ = (REBUF ⟨*uIdeal*⟩ ⟨*uDrop*⟩ ⟨*uBn*⟩ ⟨*uBt*⟩)

⟨*rt-utility*⟩ = (LOSSCORR ⟨*uIdeal*⟩ ⟨*uDrop*⟩ ⟨*cL*⟩ ⟨*cv*⟩)

The ⟨*transaction-id*⟩ is a numeric identifier for a transaction. It is used to cross-reference transactions. The scope of this identifier is one client, using QoT.

The ⟨*chunk-id*⟩ identifies a chunk of a transaction. The scope of this id is a single transaction. The ⟨*chunk-id*⟩ is used to specify the sequence and ordering of chunks inside a transaction. Chunks with the same ⟨*chunk-id*⟩ are transmitted in parallel or may be transmitted in parallel.

Each chunk is described using a filter that can be used to identify the data of this chunk in the network. One possible filter is to specify the IP 5-tuple of a transport layer connection, and optionally specify the beginning and end of the data within this transport layer connection. Depending on the network and transport layer other filters might be suitable: Internet Protocol Version 6 (IPv6) flow-labels may be identified by ⟨*flow-label*⟩, a radio bearer in 3rd Generation Partnership Project (3GPP) networks may be identified using ⟨*bearer*⟩, an Asynchronous Transfer Mode (ATM) Virtual Channel Identifier (VCI) with ⟨*VCI*⟩, a Stream Control Transmission Protocol (SCTP) stream with ⟨*stream*⟩, a local socket with ⟨*socket*⟩ or with the process ID ⟨*PID*⟩ and the file handle ⟨*FD*⟩ of this process.

The direction 'up' identified data that is transmitted in parallel to the signalization packet, 'down' identifies data transmitted in the opposite direction.

It is also important, to specify the estimated size of each traffic chunk. When the client has an improved estimation for the size of a transaction, this estimation should be updated, by sending a new transaction description for the same transaction.

The requirements of a transaction consists of a utility function description ⟨*utility*⟩ and an optional list of dependencies. Such a dependency is another transactions that has to be finished before this transaction can be started to be played back or to be shown to the user. The utility function consists of the parameters explained in Section 5.2. As explained, there are many ways to formulate such a utility function. The utility functions formulated in Section 5.2.7 corresponds to the variant "SCURVE". The utility formulation for Streaming Transactions (STs) with for the variant “REBUF” corresponds to the description in Section 5.2.8. And the variant “LOSSCORR” can be used to formulate the utility for a Real-time Transaction (RT) as described in Section 5.2.9.

Bibliography

[3GP08a] 3GPP, *Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification*, TS 36.321, 3rd Generation Partnership Project (3GPP), September 2008.

[3GP08b] ______, *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Link Control (RLC) protocol specification*, TS 36.322, 3rd Generation Partnership Project (3GPP), September 2008.

[3GP08c] ______, *Evolved Universal Terrestrial Radio Access (E-UTRA); Radio Resource Control (RRC); Protocol specification*, TS 36.331, 3rd Generation Partnership Project (3GPP), September 2008.

[3GP08d] ______, *IP Multimedia Subsystem (IMS); Stage 2*, TS 23.228, 3rd Generation Partnership Project (3GPP), September 2008.

[3GP09] ______, *cdma2000 Evaluation Methodology*, May 2009.

[3GP12] ______, *Handover procedures*, TS 23.009, 3rd Generation Partnership Project (3GPP), December 2012.

[80204] *IEEE 802.1D: IEEE Standard for Local and metropolitan area networks — Media Access Control (MAC) Bridges*, June 2004.

[AG10] A.J. Abu and S. Gordon, *A dynamic algorithm for stabilising ledbat congestion window*, Computer and Network Technology (ICCNT), 2010 Second International Conference on, April 2010, pp. 157 –161.

[AL06] J. Abley and K. Lindqvist, *Operation of Anycast Services*, RFC 4786 (Best Current Practice), December 2006.

[AMS82] D. Anick, D. Mitra, and M.M. Sondhi, *Stochastic theory of a data-handling system with multiple sources*, The Bell System Technical Journal **61** (1982), no. 8, 1872–1894.

[APY12] R. Alimi, R. Penno, and Y. Yang, *Alto protocol*, Internet Draft draft-ietf-alto-protocol-13, Internet Engineering Task Force, September 2012, Work in progress.

[AVP$^+$03] J. Antoniou, V. Vassiliou, A. Pitsillides, G. Hadjipollas, and N. Jacovides, *A simulation environment for enhanced umts performance evaluation*, Proceedings of the Australian Telecommunications, Networks and Applications Conference (ATNAC 2003), Melbourne, Australia, 2003, pp. 8–10.

[Bac06] Gary R. Bachula, *Testimony of gary r. bachula, vice president, internet2 before the united states senate committee on commerce, science and transportation hearing on net neutrality*, Senate Speech, February 2006.

[BBC$^+$98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, *An Architecture for Differentiated Services*, RFC 2475 (Informational), December 1998, Updated by RFC 3260.

[BBP$^+$14] B. Briscoe, A. Brunstrom, A. Petlund, D. Hayes, D. Ros, I.-J. Tsang, S. Gjessing, G. Fairhurst, C. Griwodz, and M. Welzl, *Reducing internet latency: A survey of techniques and their merits*, Communications Surveys Tutorials, IEEE **PP** (2014), no. 99, 1–1.

[BCD$^+$02] Alexandros Biliris, Chuck Cranor, Fred Douglis, Michael Rabinovich, Sandeep Sibal, Oliver Spatscheck, and Walter Sturm, *Cdn brokering*, Computer Communications **25** (2002), no. 4, 393 – 402.

[BDH$^+$03] L. Burgstahler, K. Dolzer, C. Hauser, J. Jähnert, S. Junghans, C. Macián, and W. Payer, *Beyond technology: the missing pieces for qos success*, Proceedings of the ACM SIGCOMM workshop on Revisiting IP QoS: What have we learned, why do we care? (New York, NY, USA), RIPQoS '03, ACM, 2003, pp. 121–130.

[BMM$^+$07] Dario Bonfiglio, Marco Mellia, Michela Meo, Dario Rossi, and Paolo Tofanelli, *Revealing skype traffic: when randomness plays with you*, SIGCOMM Comput. Commun. Rev. **37** (2007), no. 4, 37–48.

[BMM$^+$08] Naimul Basher, Aniket Mahanti, Anirban Mahanti, Carey Williamson, and Martin Arlitt, *A comparative analysis of web and peer-to-peer traffic*, Proceedings of the 17th international conference on World Wide Web (New York, NY, USA), WWW '08, ACM, 2008, pp. 287–296.

[BP12] Mike Belshe and Roberto Peon, *Spdy protocol*, no. draft-mbelshe-httpbis-spdy-00, Work in progress.

[Bra89] R. Braden, *Requirements for Internet Hosts - Communication Layers*, RFC 1122 (INTERNET STANDARD), October 1989, Updated by RFCs 1349, 4379, 5884, 6093, 6298, 6633.

[Bri08] B. Briscoe, *A fairer, faster internet*, IEEE Spectrum **45** (2008), no. 12, 42 –47.

[BS01] H. Balakrishnan and S. Seshan, *The Congestion Manager*, RFC 3124 (Proposed Standard), June 2001.

[Bun11] Bundesnetzagentur, *Tätigkeitsbericht 2010/2011 telekommunikation*, Bericht gemäß § 121 abs. 1 telekommunikationsgesetz, Bundesnetzagentur für Elektrizität, Gas, Telekommunikation, Post und Eisenbahnen, December 2011.

[BWC12] B. Briscoe, R. Woundy, and A. Cooper, *Congestion Exposure (ConEx) Concepts and Use Cases*, RFC 6789 (Informational), December 2012.

[BZB$^+$97] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*, RFC 2205 (Proposed Standard), September 1997, Updated by RFCs 2750, 3936, 4495, 5946, 6437, 6780.

[CAP10] Tom Callahan, Mark Allman, and Vern Paxson, *A longitudinal view of http traffic*, Passive and Active Measurement (Arvind Krishnamurthy and Bernhard Plattner, eds.), Lecture Notes in Computer Science, vol. 6032, Springer Berlin / Heidelberg, 2010, pp. 222–231.

[CF06] F. Cusack and M. Forssen, *Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)*, RFC 4256 (Proposed Standard), January 2006.

[Cha10] Joachim Charzinski, *Traffic Properties, Client Side Cachability and CDN Usage of Popular Web Sites*, Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance (MMB/DFT'10), Springer, 2010, pp. 136–150.

[Cis07] Cisco Systems, Cisco Systems, Inc., 170 West Tasman Drive, San Jose, CA 95134-1706 USA, *Dynamic subscriber bandwidth selection*, February 2007.

[Cis12] Cisco Systems, Inc., *Visual Networking Index*, Tech. report, 2012.

[Coh02] Bram Cohen, *The bittorrent protocol specification*, oct 2002.

[Cri03] M. Crispin, *INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1*, RFC 3501 (Proposed Standard), March 2003, Updated by RFCs 4466, 4469, 4551, 5032, 5182, 5738, 6186.

[CRJC11] Yuchung Cheng, Sivasankar Radhakrishnan, Arvind Jain, and Jerry Chu, *Tcp fast open*, no. draft-ietf-tcpm-fastopen-03, Work in progress.

[CRM91] Stuart K. Card, George G. Robertson, and Jock D. Mackinlay, *The information visualizer, an information workspace*, Proceedings of the Conference on Human Factors in Computing Systems: Reaching Through Technology (New York, NY, USA), ACM, 1991, pp. 181–186.

[CS06] Inc. Cisco Systems, *What is the maximum number of users per cmts?*, Document ID: 12205, 02 2006.

[DB99] John R. Douceur and William J. Bolosky, *A large-scale study of file-system contents*, SIGMETRICS Perform. Eval. Rev. **27** (1999), no. 1, 59–70.

[DCMP08] L. De Cicco, S. Mascolo, and V. Palmisano, *Skype video responsiveness to bandwidth variations*, Proceedings of the 18th International Workshop on Network and Operating Systems Support for Digital Audio and Video, ACM, 2008, pp. 81–86.

[DH98] S. Deering and R. Hinden, *Internet Protocol, Version 6 (IPv6) Specification*, RFC 2460 (Draft Standard), December 1998, Updated by RFCs 5095, 5722, 5871, 6437, 6564.

[DM06] Nandita Dukkipati and Nick McKeown, *Why flow-completion time is the right metric for congestion control*, SIGCOMM Comput. Commun. Rev. **36** (2006), 59–62.

[Dro97] R. Droms, *Dynamic Host Configuration Protocol*, RFC 2131 (Draft Standard), March 1997, Updated by RFCs 3396, 4361, 5494, 6842.

[Duk07] N. Dukkipati, *Rate control protocol (rcp): Congestion control to make flows complete quickly*, Ph.D. thesis, Stanford University, October 2007.

[ebn96] *ISO/IEC 14977:1996(E) First edition – Information technology – Syntactic metalanguage – Extended BNF*, Tech. report, ISO/IEC, 1996.

[Egg04] L.R. Eggert, *Background use of idle resource capacity*, Ph.D. thesis, University of Southern California, 2004.

[EGS11] Jeffrey Erman, Alexandre Gerber, and Subhabrata Sen, *Http in the home: it is not just about pcs*, SIGCOMM Comput. Commun. Rev. **41** (2011), no. 1, 90–95.

[EIP06] David Erman, Dragos Ilie, and Adrian Popescu, *Bittorrent traffic characteristics*, Computing in the Global Information Technology, 2006. ICCGI '06. International Multi-Conference on, August 2006, p. 42.

[EM92] A.I. Elwalid and D. Mitra, *Fluid models for the analysis and design of statistical multiplexing with loss priorities on multiple classes of bursty traffic*, INFOCOM '92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE, May 1992, pp. 415 –425 vol.1.

[ESR03] Sunil Erevelles, Shuba Srinivasan, and Steven Rangel, *Consumer satisfaction for internet service providers: An analysis of underlying processes*, Information Technology and Management **4** (2003), 69–89 (English).

[ETS06] ETSI, *Telecommunications and Internet converged Services and Protocols for Advanced Networking (TISPAN); NGN Release 1; Release definition*, Tech. Report TR 180 001 V1.1.1, ETSI, 650 Route des Lucioles F-06921 Sophia Antipolis Cedex - FRANCE, March 2006.

[FBC99] Johannes Faerber, Stefan Bodamer, and Joachim Charzinski, *Statistical evaluation and modeling of internet dial-up traffic*, 112–121.

[FGM$^+$99] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee, *Hypertext Transfer Protocol – HTTP/1.1*, RFC 2616 (Draft Standard), June 1999, Updated by RFCs 2817, 5785, 6266, 6585.

[FK03] Sally Floyd and Eddie Kohler, *Internet research needs better models*, SIGCOMM Comput. Commun. Rev. **33** (2003), no. 1, 29–34.

[FML$^+$03] C. Fraleigh, S. Moon, B. Lyles, C. Cotton, M. Khan, D. Moll, R. Rockell, T. Seely, and S.C. Diot, *Packet-level traffic measurements from the sprint ip backbone*, Network, IEEE **17** (2003), no. 6, 6–16.

[For07] WiMAX Forum, *Wimax system evaluation methodology*, Tech. Report Version 1.0, WiMAX Forum, 01 2007.

[GALM07] Phillipa Gill, Martin Arlitt, Zongpeng Li, and Anirban Mahanti, *Youtube traffic characterization: a view from the edge*, Proceedings of the 7th ACM SIGCOMM conference on Internet measurement (New York, NY, USA), IMC '07, ACM, 2007, pp. 15–28.

[GCX$^+$05] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang, *Measurements, analysis, and modeling of bittorrent-like systems*, Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement (Berkeley, CA, USA), IMC '05, USENIX Association, 2005, pp. 4–4.

[GDS$^+$03] Krishna P. Gummadi, Richard J. Dunn, Stefan Saroiu, Steven D. Gribble, Henry M. Levy, and John Zahorjan, *Measurement, modeling, and analysis of a peer-to-peer file-sharing workload*, Proceedings of the nineteenth ACM symposium on Operating systems principles (New York, NY, USA), SOSP '03, ACM, 2003, pp. 314–329.

[Goo11] Google Inc., *Android 4.0 APIs*, Online Reference Manual, oct 2011.

[GPVD99] E. Guttman, C. Perkins, J. Veizades, and M. Day, *Service Location Protocol, Version 2*, RFC 2608 (Proposed Standard), June 1999, Updated by RFC 3224.

[GR06] J. Galbraith and P. Remaker, *The Secure Shell (SSH) Session Channel Break Extension*, RFC 4335 (Proposed Standard), January 2006.

[Gut02] E. Guttman, *Vendor Extensions for Service Location Protocol, Version 2*, RFC 3224 (Proposed Standard), January 2002.

[GV97] A.J. Goldsmith and P.P. Varaiya, *Capacity of fading channels with channel side information*, IEEE Transactions on Information Theory **43** (1997), no. 6, 1986 –1992.

[HFPW03] M. Handley, S. Floyd, J. Padhye, and J. Widmer, *TCP Friendly Rate Control (TFRC): Protocol Specification*, RFC 3448 (Proposed Standard), January 2003, Obsoleted by RFC 5348.

[HGM$^+$02] B. Hubert, T. Graf, G. Maxwell, R. van Mook, M. van Oosterhout, P. Schroeder, J. Spaans, and P. Larroy, *Linux advanced routing & traffic control*, Ottawa Linux Symposium, 2002, p. 213.

[HKLdB05] R. Hancock, G. Karagiannis, J. Loughney, and S. Van den Bosch, *Next Steps in Signaling (NSIS): Framework*, RFC 4080 (Informational), June 2005.

[HLD$^+$05] Oliver Heckmann, Nicolas Liebau, Vasilios Darlagiannis, Axel Bock, Andreas Mauthe, and Ralf Steinmetz, *A peer-to-peer content distribution network*, From Integrated Publication and Information Systems to Information and Knowledge Environments (Matthias Hemmje, Claudia Niederée, and Thomas Risse, eds.), Lecture Notes in Computer Science, vol. 3379, Springer Berlin Heidelberg, 2005, pp. 69–78 (English).

[HLM$^+$04] Ningning Hu, Li Erran Li, Zhuoqing Morley Mao, Peter Steenkiste, and Jia Wang, *Locating internet bottlenecks: Algorithms, measurements, and implications*, vol. 34, ACM, 2004.

[HM03] Dave Hucaby and Steve McQuerry, *Cisco field manual: Catalyst switch configuration*, Cisco Press, 2003.

[HMG$^+$05] Daryl Hlasny, Jack Manbeck, Narm Gadiraju, Stephen Palm, Raj Bopardikar, Michael van Hartskamp, Richard Chen, Richard Bardini, Bruce Fairman, and Amol Bhagwat, *Upnp qos architecture:1.0*, Tech. Report 1.0, UPnP Forum., March 2005.

[HT04] D.P. Hole and F.A. Tobagi, *Capacity of an ieee 802.11b wireless lan supporting voip*, Communications, 2004 IEEE International Conference on, vol. 1, June 2004, pp. 196–201.

[IDC12] IDC, *Idc worldwide mobile phone tracker*, Press release, August 2012.

[Int02] International Telecommunication Union, *Information technology — abstract syntax notation one (asn.1): Specification of basic notation*, ITU-T Recommendation X.680, July 2002.

[Int05] International Telecommunications Union (ITU), *Estimating end-to-end performance in IP networks for data applications (ITU-T Recommendation G.1030)*, November 2005.

[ITU08] ITU, *Definitions of terms related to quality of service* $recommendationitu-te$.800, International Telecommunications Union, ITU, 2008.

[ITU11] ———, *The e-model: a computational model for use in transmission planning* $recommendationitu-tg$.107, International Telecommunications Union, ITU, 2011.

[JBB92] V. Jacobson, R. Braden, and D. Borman, *TCP Extensions for High Performance*, RFC 1323 (Proposed Standard), May 1992.

[JD05] Hao Jiang and Constantinos Dovrolis, *Why is the internet traffic bursty in short time scales?*, Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems (New York, NY, USA), SIGMETRICS '05, ACM, 2005, pp. 241–252.

[JM06] S. Jansen and A. McGregor, *Performance, validation and testing with the network simulation cradle*, Modeling, Analysis, and Simulation of Computer and

Telecommunication Systems, 2006. MASCOTS 2006. 14th IEEE International Symposium on, sept. 2006, pp. 355 – 362.

[KA06] Taehyun Kim and Mostafa H. Ammar, *Receiver buffer requirement for video streaming over tcp*, 2006, pp. 607718–607718–10.

[Kel01] Frank P Kelly, *Mathematical modelling of the internet*, Mathematics unlimited-2001 and beyond (2001), 685–702.

[KHF06] E. Kohler, M. Handley, and S. Floyd, *Datagram Congestion Control Protocol (DCCP)*, RFC 4340 (Proposed Standard), March 2006, Updated by RFCs 5595, 5596, 6335, 6773.

[KHR02] Dina Katabi, Mark Handley, and Charlie Rohrs, *Congestion control for high bandwidth-delay product networks*, Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications (New York, NY, USA), SIGCOMM '02, ACM, 2002, pp. 89–102.

[KK12] S. Kalyani and R. M. Karthik, *Analysis of opportunistic scheduling algorithms in ofdma systems in the presence of generalized fading models*, IEEE Transactions on Wireless Communications **11** (2012), no. 8, 2996 –3005.

[KLL01] A. Klemm, C. Lindemann, and M. Lohmann, *Traffic modeling and characterization for umts networks*, Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE, vol. 3, 2001, pp. 1741 –1746 vol.3.

[KLL$^+$08] Hyun-Jong Kim, Dong Hyeon Lee, Jong Min Lee, Kyoung-Hee Lee, Won Lyu, and Seong-Gon Choi, *The qoe evaluation method through the qos-qoe correlation model*, Networked Computing and Advanced Information Management, 2008. NCM '08. Fourth International Conference on, vol. 2, 2008, pp. 719–725.

[KN04] Sumit Kasera and Nishit Narang, *3g networks*, Tata McGraw-Hill Education, 2004.

[KPS$^+$12] S. Kiesel, S. Previdi, M. Stiemerling, R. Woundy, and Y. Yang, *Application-Layer Traffic Optimization (ALTO) Requirements*, RFC 6708 (Informational), September 2012.

[LC12] K.U.R. Laghari and K. Connelly, *Toward total quality of experience: A qoe model in a communication ecosystem*, Communications Magazine, IEEE **50** (2012), no. 4, 58–65.

[LFJF05] F. Liberal, A. Ferro, J.L. Jodra, and J.O. Fajardo, *Application of general perception-based qos model to find providers & responsibilities. case study: User perceived web service performance.*, Autonomic and Autonomous Systems and International Conference on Networking and Services, 2005. ICAS-ICNS 2005. Joint International Conference on, October 2005, pp. 62–62.

[Lin99] Linux NetKit authors, *telnet — user interface to the telnet protocol*, Linux NetKit (0.17) manual page, August 1999.

[Lin11] Linux man-pages project, *ip - linux ipv4 protocol implementation*, Linux manual page, September 2011.

[Lin12] ______, *socket - linux socket interface*, Linux manual page, July 2012.

[LK00] A.M. Law and W.D. Kelton, *Simulation modeling and analysis*, McGraw-Hill series in industrial engineering and management science, McGraw-Hill, 2000.

[LL06] S. Lehtinen and C. Lonvick, *The Secure Shell (SSH) Protocol Assigned Numbers*, RFC 4250 (Proposed Standard), January 2006.

[LST$^+$10] Jeongkeun Lee, Puneet Sharma, Jean Tourrilhes, Rick McGeer, Jack Brassil, and Andy Bavier, *Network integrated transparent tcp accelerator*, Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, IEEE, 2010, pp. 285–292.

[LTWW94] Will E. Leland, Murad S. Taqqu, Walter Willinger, and Daniel V. Wilson, *On the self-similar nature of ethernet traffic (extended version)*, IEEE/ACM Trans. Netw. **2** (1994), no. 1, 1–15.

[MB12] Matt Mathis and Bob Briscoe, *Congestion exposure (conex) concepts and abstract mechanism*, Internet Draft draft-ietf-conex-abstract-mech-05, Internet Engineering Task Force, July 2012, Work in progress.

[MFPA09] Gregor Maier, Anja Feldmann, Vern Paxson, and Mark Allman, *On dominant characteristics of residential broadband internet traffic*, Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference (New York, NY, USA), IMC '09, ACM, 2009, pp. 90–102.

[Mic10] Microsoft Corp., *PrimeSense Supplies 3-D-Sensing Technology to "Project Natal" for Xbox 360*, Press Release, mar 2010.

[Mil68] Robert B. Miller, *Response time in man-computer conversational transactions*, Proceedings of the December 9-11, 1968, fall joint computer conference, part I (New York, NY, USA), AFIPS '68 (Fall, part I), ACM, 1968, pp. 267–277.

[MKM10] J. Manner, G. Karagiannis, and A. McDonald, *NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling*, RFC 5974 (Experimental), October 2010.

[MSF10] Gregor Maier, Fabian Schneider, and Anja Feldmann, *A first look at mobile hand-held device traffic*, 161–170.

[MW00] Jeonghoon Mo and Jean Walrand, *Fair end-to-end window-based congestion control*, IEEE/ACM Trans. Netw. **8** (2000), no. 5, 556–567.

[NBBB98] K. Nichols, S. Blake, F. Baker, and D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, RFC 2474 (Proposed Standard), December 1998, Updated by RFCs 3168, 3260.

[Nie94] Jakob Nielsen, *Usability engineering*, Morgan Kaufmann Publishers, San Francisco, Calif., 1994.

[NMM98] Masahiko Nabe, Masayuki Murata, and Hideo Miyahara, *Analysis and modeling of world wide web traffic for capacity dimensioning of internet access lines*, Performance Evaluation **34** (1998), no. 4, 249 – 271.

[Nok10] Nokia Corporation, *Qt Reference Documentation - Bearer Management*, Nokia Corporation, `http://doc.qt.nokia.com/4.7-snapshot/bearer-management.html`, September 2010.

[NR08] H.X. Nguyen and M. Roughan, *On the correlation of internet packet losses*, Telecommunication Networks and Applications Conference, 2008. ATNAC 2008. Australasian, dec. 2008, pp. 22 –27.

[NS96] M. Naghshineh and M. Schwartz, *Distributed call admission control in mobile/wireless networks*, IEEE Journal on Selected Areas in Communications **14** (1996), no. 4, 711 –717.

[NUN10] S. Niida, S. Uemura, and H. Nakamura, *Mobile services — user tolerance for waiting time*, IEEE Vehicular Technology Magazine **5** (2010), no. 3, 61–67.

[OPTW07] S. Orlowski, M. Pióro, A. Tomaszewski, and R. Wessäly, *SNDlib 1.0–Survivable Network Design Library*, Proceedings of the 3rd International Network Optimization Conference (INOC 2007), Spa, Belgium, April 2007, http://sndlib.zib.de, extended version accepted in Networks, 2009. (English).

[Orl08] Z. Orlov, *Network-driven adaptive video streaming in wireless environments*, Personal, Indoor and Mobile Radio Communications, 2008. PIMRC 2008. IEEE 19th International Symposium on, September 2008, pp. 1 –6.

[PENUK08] L. Plissonneau, T. En-Najjary, and G. Urvoy-Keller, *Revisiting web traffic from a dsl provider perspective: the case of youtube*, Proc. of the 19th ITC Specialist Seminar, 2008.

[PKV11] M. Proebster, M. Kaschub, and S. Valentin, *Context-aware resource allocation to improve the quality of service of heterogeneous traffic*, Communications (ICC), 2011 IEEE International Conference on, June 2011, pp. 1–6.

[PM95] Venkata N. Padmanabhan and Jeffrey C. Mogul, *Improving http latency*, Computer Networks and ISDN Systems **28** (1995), no. 1–2, 25 – 35, Selected Papers from the Second World-Wide Web Conference.

[PMM93] C. Partridge, T. Mendez, and W. Milliken, *Host Anycasting Service*, RFC 1546 (Informational), November 1993.

[Pos81a] J. Postel, *Internet Control Message Protocol*, RFC 792 (INTERNET STANDARD), September 1981, Updated by RFCs 950, 4884, 6633.

[Pos81b] ———, *Internet Protocol*, RFC 791 (INTERNET STANDARD), September 1981, Updated by RFCs 1349, 2474.

[PR83] J. Postel and J.K. Reynolds, *Telnet Protocol Specification*, RFC 854 (INTERNET STANDARD), May 1983, Updated by RFC 5198.

[PR85] J. Postel and J. Reynolds, *File Transfer Protocol*, RFC 959 (INTERNET STANDARD), October 1985, Updated by RFCs 2228, 2640, 2773, 3659, 5797.

[RCC$^+$11] Sivasankar Radhakrishnan, Yuchung Cheng, Jerry Chu, Arvind Jain, and Barath Raghavan, *Tcp fast open*, Proceedings of the 7th International Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2011.

[Rec96] ITUT Rec, *P. 800: Methods for subjective determination of transmission quality*, August 1996.

[RFB01] K. Ramakrishnan, S. Floyd, and D. Black, *The Addition of Explicit Congestion Notification (ECN) to IP*, RFC 3168 (Proposed Standard), September 2001, Updated by RFCs 4301, 6040.

[RFW06] T. Rahrer, R. Fiandra, and S. Wright, *Technical report tr-126 triple-play services quality of experience (qoe) requirements*, DSL Forum, Tech. Rep, 2006.

[Ros08] J. Rosenberg, *Udp and tcp as the new waist of the internet hourglass*, Internet Draft draft-rosenberg-internet-waist-hourglass-00, Internet Engineering Task Force, February 2008, Work in progress.

[RRO98] D. Reininger, D. Raychaudhuri, and M. Ott, *Market based bandwidth allocation policies for qos control in broadband networks*, Proceedings of the First International Conference on Information and Computation Economies (New York, NY, USA), ICE '98, ACM, 1998, pp. 101–110.

[RSC$^+$02] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261 (Proposed Standard), June 2002, Updated by RFCs 3265, 3853, 4320, 4916, 5393, 5621, 5626, 5630, 5922, 5954, 6026, 6141, 6665.

[RTV10] Dario Rossi, Claudio Testa, and Silvio Valenti, *Yes, we ledbat: Playing with the new bittorrent congestion control algorithm*, Passive and Active Measurement (Arvind Krishnamurthy and Bernhard Plattner, eds.), Lecture Notes in Computer Science, vol. 6032, Springer Berlin Heidelberg, 2010, pp. 31–40.

[SAAF08] Fabian Schneider, Sachin Agarwal, Tansu Alpcan, and Anja Feldmann, *The new web: Characterizing ajax traffic*, Passive and Active Network Measurement (Mark Claypool and Steve Uhlig, eds.), Lecture Notes in Computer Science, vol. 4979, Springer Berlin / Heidelberg, 2008, 10.1007/978-3-540-79232-1_4, pp. 31–40.

[San12a] Sandvine Incorporated, *Global Internet Phenomena Report*, Tech. report, 2012.

[San12b] ———, *Global internet phenomena spotlight europe fixed*, Tech. report, Sandvine Incorporated ULC, April 2012.

[SCUKB07] Matti Siekkinen, Denis Collange, Guillaume Urvoy-Keller, and ErnstW. Biersack, *Performance limitations of adsl users: A case study*, Passive and Active Network Measurement (Steve Uhlig, Konstantina Papagiannaki, and Olivier Bonaventure, eds.), Lecture Notes in Computer Science, vol. 4427, Springer Berlin Heidelberg, 2007, pp. 145–154.

[SH00] Fareena Sultan and Roy B Henrichs, *Consumer preferences for internet services over time: initial explorations*, Journal of Consumer Marketing **17** (2000), no. 5, 386–402.

[SHIK12a] S. Shalunov, G. Hazel, J. Iyengar, and M. Kuehlewind, *Low Extra Delay Background Transport (LEDBAT)*, RFC 6817 (Experimental), December 2012.

[SHIK12b] ———, *Low Extra Delay Background Transport (LEDBAT)*, Internet Draft draft-ietf-ledbat-congestion-10, Internet Engineering Task Force, October 2012, Work in progress.

[SK06] M. Scharf and S. Kiesel, *Nxg03-5: Head-of-line blocking in tcp and sctp: Analysis and measurements*, Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE, 2006, pp. 1–5.

[SM03] A Striegel and G Manimaran, *Dynamic class-based queue management for scalable media servers*, Journal of Systems and Software **66** (2003), no. 2, 119 – 128.

[SMW07] H. Schwarz, D. Marpe, and T. Wiegand, *Overview of the scalable video coding extension of the h.264/avc standard*, Circuits and Systems for Video Technology, IEEE Transactions on **17** (2007), no. 9, 1103 –1120.

[Soc05] IEEE Computer Society, *IEEE 802.11e: Medium Access Control (MAC) Quality of Service Enhancements*, IEEE, EEE 3 Park Avenue New York, NY 10016-5997, USA, ieee standard for information technology— telecommunications and information exchange between systems— local and metropolitan area networks— specific requirements part 11: wireless lan medium access control (mac) and physical layer (phy) specifications ed., November 2005.

[Soc11] ______, *IEEE 802.1q: Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks*, IEEE, IEEE 3 Park Avenue New York, NY 10016-5997 USA, ieee standard for local and metropolitan area networks – 2011 ed., August 2011.

[Spd] *Spdy: An experimental protocol for a faster web.*

[SRC84] J. H. Saltzer, D. P. Reed, and D. D. Clark, *End-to-end arguments in system design*, ACM Trans. Comput. Syst. **2** (1984), no. 4, 277–288.

[STAD10] M. Stiemerling, H. Tschofenig, C. Aoun, and E. Davies, *NAT/Firewall NSIS Signaling Layer Protocol (NSLP)*, RFC 5973 (Experimental), October 2010.

[Ste46] S. S. Stevens, *On the theory of scales of measurement*, Science **103** (1946), no. 2684, 677–680.

[Ste07] R. Stewart, *Stream Control Transmission Protocol*, RFC 4960 (Proposed Standard), September 2007, Updated by RFCs 6096, 6335.

[STK99] S. Sahu, D. Towsley, and J. Kurose, *A quantitative study of differentiated services for the internet*, Global Telecommunications Conference, 1999. GLOBECOM '99, vol. 3, 1999, pp. 1808 –1817 vol.3.

[SUKBEN05] M. Siekkinen, G. Urvoy-Keller, E. W. Biersack, and T. En-Najjary, *Root cause analysis for long-lived tcp connections*, Proceedings of the 2005 ACM conference on Emerging network experiment and technology (New York, NY, USA), CoNEXT ’05, ACM, 2005, pp. 200–210.

[SXM$^+$00] R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang, and V. Paxson, *Stream Control Transmission Protocol*, RFC 2960 (Proposed Standard), October 2000, Obsoleted by RFC 4960, updated by RFC 3309.

[TC05] Fang-Mei Tseng and Yu-Jing Chiu, *Hierarchical fuzzy integral stated preference method for taiwan’s broadband service market*, Omega **33** (2005), no. 1, 55 – 64.

[TFK$^+$11] R. Torres, A. Finamore, Jin Ryong Kim, M. Mellia, M.M. Munafo, and Sanjay Rao, *Dissecting video server selection strategies in the youtube cdn*, Distributed Computing Systems (ICDCS), 2011 31st International Conference on, June 2011, pp. 248 –257.

[THB06] Andrew S. Tanenbaum, Jorrit N. Herder, and Herbert Bos, *File size distribution on unix systems: then and now*, SIGOPS Oper. Syst. Rev. **40** (2006), no. 1, 100–104.

[Tho08] Thompson, *Triple-Play Using IPoE for Voice, PPPoE for Data and Bridged Video on Multiple PVCs (with VLANs)*, Application note, Thompson, April 2008.

[TMW97] Kevin Thompson, Gregory J Miller, and Rick Wilder, *Wide-area internet traffic patterns and characteristics*, Network, iEEE **11** (1997), no. 6, 10–23.

[Tut04] Kurt Tutschku, *A measurement-based traffic profile of the edonkey filesharing service*, Passive and Active Network Measurement (Chadi Barakat and Ian Pratt, eds.), Lecture Notes in Computer Science, vol. 3015, Springer Berlin / Heidelberg, 2004, 10.1007/978-3-540-24668-8_2, pp. 12–21.

[Uni07] International Telecommunications Union, *ITU-T Recommendation H.264 : Advanced video coding for generic audiovisual services*, November 2007.

[Ver00] S. Verdu, *Wireless bandwidth in the making*, Communications Magazine, IEEE **38** (2000), no. 7, 53 –58.

[VZ02] R. Verdone and A. Zanella, *Performance of received power and traffic-driven handover algorithms in urban cellular networks*, IEEE Wireless Communications **9** (2002), no. 1, 60 –71.

[WAHC$^+$06] Michele C. Weigle, Prashanth Adurthi, Félix Hernández-Campos, Kevin Jeffay, and F. Donelson Smith, *Tmix: a tool for generating realistic tcp application workloads in ns-2*, SIGCOMM Comput. Commun. Rev. **36** (2006), no. 3, 65–76.

[Wil41] Roger I. Wilkinson, *The reliability of holding time measurements*, Bell Systems Technical Journal **20** (1941), no. 4.

[WR11] M. Welzl and D. Ros, *A Survey of Lower-than-Best-Effort Transport Protocols*, RFC 6297 (Informational), June 2011.

[YL06a] T. Ylonen and C. Lonvick, *The Secure Shell (SSH) Authentication Protocol*, RFC 4252 (Proposed Standard), January 2006.

[YL06b] ______, *The Secure Shell (SSH) Connection Protocol*, RFC 4254 (Proposed Standard), January 2006.

[YL06c] ______, *The Secure Shell (SSH) Protocol Architecture*, RFC 4251 (Proposed Standard), January 2006.

[YL06d] ______, *The Secure Shell (SSH) Transport Layer Protocol*, RFC 4253 (Proposed Standard), January 2006, Updated by RFC 6668.

[YRK$^+$07] M. Yuksel, K.K. Ramakrishnan, S. Kalyanaraman, J.D. Houle, and R. Sadhvani, *Value of supporting class-of-service in ip backbones*, Fifteenth IEEE International Workshop on Quality of Service, 2007, June 2007, pp. 109 –112.

[YYP04] J. Yang, J. Ye, and S. Papavassiliou, *Enhancing end-to-end qos granularity in diffserv networks via service vector and explicit endpoint admission control*, IEE Proceedings on Communications **151** (2004), no. 1, 77 – 81.

[YZZZ06] Hongliang Yu, Dongdong Zheng, Ben Y. Zhao, and Weimin Zheng, *Understanding user behavior in large-scale video-on-demand systems*, SIGOPS Oper. Syst. Rev. **40** (2006), no. 4, 333–344.

[ZSGK09] Michael Zink, Kyoungwon Suh, Yu Gu, and Jim Kurose, *Characteristics of youtube network traffic at a campus network – measurements, models, and implications*, Computer Networks **53** (2009), no. 4, 501 – 514, Content Distribution Infrastructures for Community Networks.