Network Working Group Michael Welzl
Internet Draft Dimitri Papadimitriou
Editors

Michael Scharf
Bob Briscoe

Expires: February 4, 2009 August 5, 2008

Open Research Issues in Internet Congestion Control

draft-irtf-iccrg-welzl-congestion-control-open-research-02.txt

Status of this Memo

Copyright Notice

Abstract

This document describes some of the open problems in Internet congestion control that are known today. This includes several new

challenges that are becoming important as the network grows, as well as some issues that have been known for many years. These challenges are generally considered to be open research topics that may require more study or application of innovative techniques before Internet-scale solutions can be confidently engineered and deployed.

Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC-2119 [i].

Table of Contents

1. Introduction

This document describes some of the open research topics in the domain of Internet congestion control that are known today. We begin by reviewing some proposed definitions of congestion and congestion control based on current understandings.

Congestion can be defined as the reduction in utility due to overload in networks that support both spatial and temporal multiplexing, but no reservation [Keshav07]. Congestion control is a (typically distributed) algorithm to share network resources among competing traffic sources. Two components of distributed congestion control have been defined in the context of prima-dual modeling [Kelly98]. Primal congestion control refers to the algorithm executed by the traffic sources algorithm for controlling their sending rates or window sizes. This is normally a closed-loop control, where this operation depends on feedback. TCP algorithms fall in this category. Dual congestion control is implemented by the routers through gathering information about the traffic traversing them. A dual congestion control algorithm updates, implicitly or explicitly, a congestion measure and sends it back, implicitly or explicitly, to the traffic sources that use that link. Queue management algorithms such as Random Early Detection (RED) [Floyd93] or Random Exponential Marking (REM) [Ath01] fall in the "dual" category.

Congestion control provides for a fundamental set of mechanisms for maintaining the stability and efficiency of the Internet. Congestion control has been associated with TCP since Van Jacobson's work in 1988, but there is also congestion control outside of TCP (e.g. for real-time multimedia applications, multicast, and router-based mechanisms). The Van Jacobson end-to-end congestion control algorithms [Jacobson88] [RFC2581] are used by the Internet transport protocol TCP [RFC4614]. They have been proven to be highly successful over many years but have begun to reach their limits, as the heterogeneity of both the data link and physical layer and applications are pulling TCP congestion control (which performs poorly as the bandwidth or delay increases) outside of its natural operating regime. A side effect of these deficits is that there is an increasing share of hosts that use non-standardized congestion control enhancements (for instance, many Linux distributions have been shipped with "CUBIC" as default TCP congestion control mechanism).

While the original Jacobson algorithm requires no congestion-related state in routers, more recent modifications have departed from the strict application of the end-to-end principle [Saltzer84]. Active Queue Management (AQM) in routers, e.g., RED and its variants such as xCHOKE [Pan00], RED with In/Out (RIO) [Clark98], improves performance by keeping queues small (implicit feedback via dropped packets), while Explicit Congestion Notification (ECN) [Floyd94] [RFC3168] passes one bit of congestion information back to senders when an AQM would normally drop a packet. These measures do improve performance, but there is a limit to how much can be accomplished without more information from routers. The requirement of extreme scalability together with robustness has been a difficult hurdle to accelerating information flow. Primal-Dual TCP/AQM distributed algorithm stability and equilibrium properties have been extensively studied (cf. [Low02] [Low03]).

Congestion control includes many new challenges that are becoming important as the network grows in addition to the issues that have been known for many years. These are generally considered to be open research topics that may require more study or application of innovative techniques before Internet-scale solutions can be confidently engineered and deployed. In what follows, an overview of some of these challenges is given.

2. Global Challenges

This section describes the global challenges to be addressed in the domain of Internet congestion control.

2.1 Heterogeneity

The Internet encompasses a large variety of heterogeneous IP networks that are realized by a multitude of technologies, which result in a tremendous variety of link and path characteristics: capacity can be either scarce in very slow speed radio links (several kbps), or there may be an abundant supply in high-speed optical links (several gigabit per second). Concerning latency, scenarios range from local interconnects (much less than a millisecond) to certain wireless and satellite links with very large latencies (up to a second). Even higher latencies can occur in interstellar communication. As a consequence, both the available bandwidth and the end-to-end delay in the Internet may vary over many orders of magnitude, and it is likely that the range of parameters will further increase in future.

Additionally, neither the available bandwidth nor the end-to-end delay is constant. At the IP layer, competing cross-traffic, traffic management in routers, and dynamic routing can result in sudden changes of the characteristics of an end-to-end path. Additional dynamics can be caused by link layer mechanisms, such as shared media access (e.g., in wireless networks), changes of links (horizontal/vertical handovers), topology modifications (e.g., in ad-hoc networks), link layer error correction and dynamic bandwidth provisioning schemes. From this follows that path characteristics can be subject to substantial changes within short time frames.

The congestion control algorithms have to deal with this variety in an efficient way. The congestion control principles introduced by Van Jacobson assume a rather static scenario and implicitly target configurations where the bandwidth-delay product is of the order of some dozens of packets at most. While these principles have proved to work well in the Internet for almost two decades, much larger bandwidth-delay products and increased dynamics challenge them more and more. There are many situations where today's congestion control algorithms react in a suboptimal way, resulting in low resource utilization, non-optimal congestion avoidance, or unfairness.

This gave rise to a multitude of new proposals for congestion control algorithms. For instance, since the Additive-Increase Multiplicative Decrease (AIMD) behavior of TCP is too conservative in practical environments when then congestion window is large, several high-speed congestion control extensions have been developed. However, these new algorithms raise fairness issues, and they may be less robust in certain situations for which they have not been designed. Up to now, there is still no common agreement in the IETF on which algorithm and protocol to choose.

It is always possible to tune congestion control parameters based on some knowledge about the environment and the application scenario. However, the fundamental question is whether it is possible to define one congestion control mechanism that operates reasonable well in the whole range of scenarios that exist in the Internet. Hence, it is an important research question how new Internet congestion control mechanisms would have to be designed, which maximum degree of dynamics it could efficiently handle, and whether it could keep the genererality of the existing end-to-end solutions.

Some improvements of congestion control could be realized by simple changes of single functions in end-system or network components. However, new mechanism can also require a fundamental redesign of the overall network architecture, and they may even affect the design of Internet applications. This can imply significant interoperability and backward compatibility challenges and/or create network accessibility obstacles. In particular, networks and/or applications that do not use or support a new congestion control mechanism could be penalized by a significantly worse performance compared to what they would get if everybody used the existing mechanisms (cf. the discussion on fairness in Section 2.3). [RFC5033] defines several criteria to evaluate the appropriateness of a new congestion control mechanism. However, the fundamental question is how much performance deterioration is acceptable for "legacy" applications. This tradeoff between performance and cost has to be very carefully examined for all new congestion control schemes.

2.2 Stability

Control theory, which is a mathematical tool for describing dynamic systems, lends itself to modeling congestion control - TCP is a perfect example of a typical "closed loop" system that can be described in control theoretic terms. In control theory, there is a mathematically defined notion of system stability. In a stable system, for any bounded input over any amount of time, the output will also be bounded. For congestion control, what is actually meant with stability is typically asymptotic stability: a mechanism should converge to a certain state irrespective of the initial state of the network.

Control theoretic modeling of a realistic network can be quite difficult, especially when taking distinct packet sizes and heterogeneous RTTs into account. It has therefore become common practice to model simpler cases and leave the more complicated (realistic) situations for simulations. Clearly, if a mechanism is not stable in a simple scenario, it is generally useless; this method therefore helps to eliminate faulty congestion control candidates at an early stage.

Some fundamental facts, which are known from control theory are useful as guidelines when designing a congestion control mechanism. For instance, a controller should only be fed a system state that reflects its output. A (low-pass) filter function should be used in order to pass only states to the controller that are expected to last long enough for its action to be meaningful [Jain88]. Action should be carried out whenever such feedback arrives, as it is a fundamental principle of control that the control frequency should be equal to the feedback frequency. Reacting faster leads to oscillations and instability while reacting slower makes the system tardy [Jain90].

TCP stability can be attributed to two key aspects which were introduced in [Jacobson88]: the AIMD control law during congestion avoidance, which is based on a simple, vector based analysis of two controllers sharing one resource with synchronous RTTs [Chiu89], and the "conservation of packets principle", which, once the control has reached "steady state", tries to maintain an equal amount of packets in flight at any time by only sending a packet into the network when a packet has left the network (as indicated by an ACK arriving at the sender). The latter aspect has guided many decisions regarding changes that were made to TCP over the years.

The reasoning in [Jacobson88] assumes all senders to be acting at the same time. The stability of TCP under more realistic network conditions has been investigated in a large number of ensuing works, leading to no clear conclusion that TCP would also be asymptotically stable under arbitrary network conditions. The stability impact of

Slow Start (which can be significant as short-lived HTTP flows often never leave this phase) is also not entirely clear.

2.3 Fairness

Recently, the way the Internet community reasons about fairness has been called into deep questioning [Bri07]. Much of the community has taken fairness to mean approximate equality between the rates of flows (flow rate fairness) that experience equivalent path congestion as with TCP [RFC2581] and TFRC [RFC3448]. [RFC3714] depicts the resulting situation as "The Amorphous Problem of Fairness".

A parallel tradition has been built on [Kelly98] where, as long as each user is accountable for the cost their rate causes to others [MKMV95], the set of rates that everyone chooses is deemed fair (cost fairness) - because with any other set of choices people would lose more value than they gained overall.

In comparison, the debate between max-min, proportional and TCP fairness is about mere details. These three all share the assumption that equal flow rates are desirable; they merely differ in the second order issue of how to share out excess capacity in a network of many bottlenecks. In contrast, cost fairness should lead to extremely unequal flow rates by design. Equivalently, equal flow rates would typically be considered extremely unfair.

The two traditional approaches are not protocol options that can each be followed in different parts of a network. They result in research agendas and issues that are different in their respective objectives resulting in different set of open issues.

If we assume TCP-friendliness as a goal with flow rate as the metric, open issues would be:

- Should rate fairness depend on the packet rate or the bit rate?
- Should the flow rate depend on RTT (as in TCP) or should only flow dynamics depend on RTT (e.g. as in Fast TCP [Jin04])?
- How to estimate whether a particular flow start strategy is fair, or whether a particular fast recovery strategy after a reduction in rate due to congestion is fair?
- How should we judge what is reasonably fair if an application needs, for example, even smoother flows than TFRC, or it needs to burst occasionally, or with any other application behavior?
- During brief congestion bursts (e.g. due to new flow arrivals) how to judge at what point it becomes unfair for some flows to continue at a smooth rate while others reduce their rate?
- Which mechanism(s) should be used to enforce approximate flow rate fairness?
- How can we introduce some degree of fairness that takes account of

  flow duration?
- How to judge the fairness of applications using a large number of flows over separate paths (e.g., via an overlay)?

If we assume cost fairness as a goal with congestion volume as the metric, open issues would be:

- Can one application’s sensitivity to instantaneous congestion really be protected by longer-term accountability of competing applications?
- Which protocol mechanism(s) should give accountability for causing congestion?
- How to design one or two generic transport protocols (such as to TCP, UDP, etc.) with the addition of application policy control?
- Which policy enforcement should be used by networks and which interactions between application policy and network policy enforcement?
- How to design a new policy enforcement framework that will appropriately compete with existing flows aiming for rate equality

  (e.g. TCP)?

The question of how to reason about fairness is a pre-requisite to agreeing on the research agenda. However, that question does not require more research in itself, it is merely a debate that needs to be resolved by studying existing research and by assessing how bad fairness problems could become if they are not addressed rigorously.

3. Detailed Challenges

3.1 Challenge 1: Router Support

Routers can be involved in congestion control in two ways: first, they can implicitly optimize their functions, such as queue management and scheduling strategies, in order to support the operation of an end-to-end congestion control. Second, routers can participate in congestion control via explicit notification mechanisms.

In the first category, various approaches have been proposed and also deployed, such as different AQM techniques. Even though these implicit techniques are known to improve network performance during congestion phases, they are still only partly deployed in the Internet. This may be due to the fact that finding optimal and robust parameterizations for these mechanisms is a non-trivial problem. Indeed, the problem with various AQM schemes is the difficulty to identify correct values of the parameter set that affects the performance of the queuing scheme (due to variation in the number of sources, the capacity and the feedback delay) [Fioriu00] [Hollot01] [Zhang03]. Many AQM schemes (RED, REM, BLUE, PI-Controller but also

Adaptive Virtual Queue (AVQ)) do not define a systematic rule for setting their parameters.

By using explicit feedback from the network, connection endpoints can obtain more accurate information about the current network characteristics on the path. This allows endpoints to make more precise decisions that can better prevent packet loss and that can also improve fairness among different flows. Examples for explicit router feedback include Explicit Congestion Notification (ECN) [RFC3168], Quick-Start [RFC4782], the eXplicit Control Protocol (XCP) [Katabi02] [Falk07], the Rate Control Protocol (RCP) [Dukk06], and CADPC/PTP [Welzl03].

Explicit router feedback can address some of the inherent shortcomings of TCP. For instance, XCP has been developed to overcome the inefficiency, unfairness and instability that TCP suffers from when the per-flow bandwidth-delay product increases. By decoupling resource utilization/congestion control from fairness control, XCP achieves fair bandwidth allocation, high utilization, a small standing queue size, and near-zero packet drops, with both steady and highly varying traffic. Importantly, XCP does not maintain any per-flow state in routers and requires few CPU cycles per packet, hence making it potentially applicable in high-speed routers. However, XCP is still subject to research: as [Andrew05] has pointed out, XCP is locally stable but globally unstable when the maximum RTT of a flow is much larger than the mean RTT. This instability can be removed by changing the update strategy for the estimation interval, but this makes the system vulnerable to erroneous RTT advertisements. The authors of [PAP02] have shown that, when flows with different RTTs are applied, XCP sometimes discriminates among heterogeneous traffic flows, even if XCP is generally fair to different flows. [Low05] provides for a complete characterization of the XCP equilibrium properties.

Several other explicit router feedback schemes have been developed with different design objectives. For instance, RCP uses a per-packet feedback similar to XCP. Different to XCP, RCP focuses on the reduction of flow completion times and therefore tolerates larger instantaneous queue sizes [Dukk06].

Both implicit and explicit router support should be considered in the context of the end-to-end argument [Saltzer84], which is one of the key design principles of the Internet. It suggests that functions that can be realized both in the end-systems and in the network should be implemented in the end-systems. This principle ensures that the network provides a general service and that remains as simple as possible (any additional complexity is placed above the IP layer, i.e., at the edges) so as to ensure reliability and robustness. In particular, this means that Internet protocols should not rely on the maintenance of applicative state (i.e., information about the state of the end-to-end communication) inside the network [RFC1958] and that the network state (e.g. routing state) maintained by the Internet shall minimize its interaction with the states maintained at the end-points/hosts.

However, as discussed for instance in [Moors02], congestion control cannot be realized as a pure end-to-end function only. Congestion is an inherent network phenomenon and can only be resolved efficiently by some cooperation of end-systems and the network. Congestion control in today's Internet protocols follows the end-to-end design principle insofar as only minimal feedback from the network is used (e. g., packet loss and delay). The end-systems only decide how to react and how to avoid congestion. The crux is that, on the one hand, there would be substantial benefit by further assistance from the network, but, on the other hand, such router support could lead to duplication of functions, which might even harmfully interact with end-to-end protocol mechanisms. The different requirements of applications (cf. the fairness discussion in Section 2.3) call for a variety of different congestion control approaches, but putting such application-specific behavior inside the network should be avoided, as such design would clearly be at odds with the end-to-end design principle.

The end-to-end argument is generally regarded as a key ingredient for ensuring a scalable network design. In order to ensure that new congestion control mechanisms are scalable, violating this principle must therefore be avoided.

In general, router support raises many issues that have not been completely solved yet.

3.1.1 Performance and robustness

Congestion control is subject to some tradeoffs: on one hand, it must allow high link utilizations and fair resource sharing but on the other hand, the algorithms must also be robust and conservative in particular during congestion phases.

Router support can help to improve performance and fairness, but it can also result in additional complexity and more control loops. This requires a careful design of the algorithms in order to ensure stability and avoid e.g. oscillations. A further challenge is the fact that information may be imprecise. For instance, severe congestion can delay feedback signals. Also, the measurement of parameters such as RTTs or data rates may contain estimation errors. Even though there has been significant progress in providing fundamental theoretical models for such effects, research has not completely explored the whole problem space yet.

Open questions are:

- How much can routers theoretically improve performance in the complete range of communication scenarios that exists in the Internet without damaging or impacting end-to-end mechanisms already in place?

- Is it possible to design robust mechanisms that offer significant benefits without additional risks?

- What is the minimum support that is needed from routers in order to achieve significantly better performance than with end-to-end mechanisms?

3.1.2 Granularity of router functions

There are several degrees of freedom concerning router involvement, ranging from some few additional functions in network management procedures one the one end, and additional per packet processing on the other end of the solution space. Furthermore, different amounts of state can be kept in routers (no per-flow state, partial per-flow state, soft state, hard state). The additional router processing is a challenge for Internet scalability and could also increase end-to-end latencies.

There are many solutions that do not require per-flow state and thus do not cause a large processing overhead. However, scalability issues could also be caused, for instance, by synchronization mechanisms for state information among parallel processing entities, which are e. g. used in high-speed router hardware designs.

Open questions are:

- What granularity of router processing can be realized without affecting Internet scalability?

- How can additional processing efforts be kept at a minimum?

3.1.3 Information acquisition

In order to support congestion control, routers have to obtain at least a subset of the following information. Obtaining that information may result in complex tasks.

1. Capacity of (outgoing) links

Link characteristics depend on the realization of lower protocol layers. Routers do not necessarily know the link layer network topology and link capacities, and these are not always constant (e.g., on shared wireless links). Depending on the network technology, there can be queues or bottlenecks that are not directly visible at the IP layer. Difficulties also arise when using IP-in-IP tunnels [RFC2003] or MPLS [RFC3031] [RFC3032]. In these cases, link information could be determined by cross-layer information exchange, but this requires link layer technology specific interfaces. An alternative could be online measurements, but this can cause significant additional network overhead.

2. Traffic carried over (outgoing) links

Accurate online measurement of data rates is challenging when traffic is bursty. For instance, measuring a "current link load" requires defining the right measurement interval/ sampling interval. This is a challenge for proposals that require knowledge e.g. about the current link utilization.

3. Internal buffer statistics

Some proposals use buffer statistics such as a virtual queue length to trigger feedback. However, routers can include multiple distributed buffer stages that make it difficult to obtain such metrics.

Open questions are: Can and should this information be made available, e.g., by additional interfaces or protocols?

3.1.4 Feedback signaling

Explicit notification mechanisms can be realized either by in-band signaling (notifications piggybacked along with the data traffic) or by out-of-band signaling. The latter case requires additional protocols and can be further subdivided into path-coupled and path-decoupled approaches.

Open questions concerning feedback signaling include:

- At which protocol layer should the feedback signaling occur (IP/network layer assisted, transport layer assisted, hybrid solutions, shim layer, intermediate sub-layer, etc.) ?

- What is the optimal frequency of feedback (only in case of congestion events, per RTT, per packet, etc.)?

3.2 Challenge 2: Corruption Loss

It is common for congestion control mechanisms to interpret packet loss as a sign of congestion. This is appropriate when packets are dropped in routers because of a queue that overflows, but there are other possible reasons for packet drops. In particular, in wireless networks, packets can be dropped because of corruption, rendering the typical reaction of a congestion control mechanism inappropriate.

TCP over wireless and satellite is a topic that has been investigated for a long time [Krishnan04]. There are some proposals where the congestion control mechanism would react as if a packet had not been dropped in the presence of corruption (cf. TCP HACK [Balan01]), but discussions in the IETF have shown that there is no agreement that this type of reaction is appropriate. For instance, it has been said that congestion can manifest itself as corruption on shared wireless links, and it is questionable whether a source that sends packets that are continuously impaired by link noise should keep sending at a high rate.

Generally, two questions must be addressed when designing congestion control mechanism that takes corruption into account:

1. How is corruption detected?

2. What should be the reaction?

In addition to question 1 above, it may be useful to consider detecting the reason for corruption, but this has not yet been done to the best of our knowledge.

Corruption detection can be done using an in-band or out-of-band signaling mechanism, much in the same way as described for Challenge 1. Additionally, implicit detection can be considered: link layers sometimes retransmit erroneous frames, which can cause the end-to-end delay to increase - but, from the perspective of a sender at the transport layer, there are many other possible reasons for such an effect.

Header checksums provide another implicit detection possibility: if a checksum only covers all the necessary header fields and this checksum does not show an error, it is possible for errors to be found in the payload using a second checksum. Such error detection is possible with UDP-Lite and DCCP; it was found to work well over a GPRS network in a study [Chester04] and poorly over a WiFi network in another study [Rossi06] [Welzl08]. Note that, while UDP-Lite and DCCP enable the detection of corruption, the specifications of these protocols do not foresee any specific reaction to it for the time being.

The idea of having a transport endpoint detect and accordingly react to corruption poses a number of interesting questions regarding cross-layer interactions. As IP is designed to operate over arbitrary link layers, it is therefore difficult to design a congestion control mechanism on top of it, which appropriately reacts to corruption - especially as the specific data link layers that are in use along an end-to-end path are typically unknown to entities at the transport layer.

While the IETF has not yet specified how a congestion control mechanism should react to corruption, proposals exist in the literature. For instance, TCP Westwood sets the congestion window equal to the measured bandwidth at time of congestion in response to three DupACKs or a timeout. This measurement is obtained by counting and filtering the ACK rate. This setting provides a significant goodput improvement in noisy channels because the "blind" by half window reduction of standard TCP is avoided, i.e. the window is not reduced by too much [Mascolo01].

Open questions concerning corruption loss include:

- How should corruption loss be detected?

- How should a source react when it is known that corruption has occurred?

3.3 Challenge 3: Small Packets

Over past years, the performance of TCP congestion avoidance algorithms has been extensively studied. The well known "square root formula" provides the performance of the TCP congestion avoidance algorithm for TCP Reno [RFC2581]. [Padhye98] enhances the model to account for timeouts, receiver window, and delayed ACKs.

For the sake of the present discussion, we will assume that the TCP throughput is expressed using the simplified formula. Using this formula, the TCP throughput is proportional to the packet size and inversely proportional to the RTT and the square root of the drop probability:

$$B \sim C \frac{MSS}{RTT} \frac{1}{sqrt(p)}$$

where,

MSS is the TCP segment size (in bytes)
RTT is the end-to-end round trip time of the TCP connection (in seconds)
p is the packet drop probability

Neglecting the fact that the TCP rate linearly depends on it, choosing the ideal packet size is a trade-off between high throughput (the larger a packet, the smaller the relative header overhead) and low delay (the smaller a packet, the shorter the time that is needed until it is filled with data). Observing that TCP is not suited for applications such as streaming media (since reliable in-order delivery and congestion control can cause arbitrarily long delays), this trade-off has not usually been considered for TCP applications, and the influence of the packet size on the sending rate is not typically seen as a significant issue.

The situation is different for the Datagram Congestion Control Protocol (DCCP) [RFC4340], which has been designed to enable unreliable but congestion-controlled datagram transmission, avoiding the arbitrary delays associated with TCP. DCCP is intended for applications such as streaming media that can benefit from control over the tradeoffs between delay and reliable in-order delivery.

DCCP provides for a choice of modular congestion control mechanisms. DCCP uses Congestion Control Identifiers (CCIDs) to specify the congestion control mechanism. Three profiles are currently specified:

- DCCP Congestion Control ID 2 (CCID 2) [RFC4341]:
  TCP-like Congestion Control. CCID 2 sends data using a close variant of TCP’s congestion control mechanisms, incorporating a variant of SACK [RFC2018, RFC3517]. CCID 2 is suitable for senders who can adapt to the abrupt changes in congestion window typical of TCP’s AIMD congestion control, and particularly useful for senders who would like to take advantage of the available bandwidth in an environment with rapidly changing conditions.
- DCCP Congestion Control ID 3 (CCID 3) [RFC4342]:
  TCP-Friendly Rate Control (TFRC) [RFC3448bis] is a congestion control mechanism designed for unicast flows operating in a best-effort Internet environment. It is reasonably fair when competing for bandwidth with TCP flows, but has a much lower variation of throughput over time compared with TCP, making it more suitable for applications such as streaming media where a relatively smooth sending rate is of importance. CCID 3 is appropriate for flows that would prefer to minimize abrupt changes in the sending rate, including streaming media applications with small or moderate receiver buffering before playback.
- DCCP Congestion Control ID 4 [draft-ietf-ccid4-02.txt]:
  TFRC Small Packets (TFRC-SP) [RFC4828], a variant of TFRC mechanism has been designed for applications that exchange small packets. The objective of TFRC-SP is to achieve the same bandwidth in bps (bits per second) as a TCP flow using packets of up to 1500 bytes. TFRC-SP enforces a minimum interval of 10 ms between data packets to prevent a single flow from sending small packets arbitrarily frequently. TFRC is a congestion control mechanism for unicast flows operating in a best-effort Internet environment, and

is designed for DCCP that controls the sending rate based on a stochastic Markov model for TCP Reno. CCID 4 has been designed to be used either by applications that use a small fixed segment size, or by applications that change their sending rate by varying the segment size. Because CCID 4 is intended for applications that use a fixed small segment size, or that vary their segment size in response to congestion, the transmit rate derived from the TCP throughput equation is reduced by a factor that accounts for packet header size, as specified in [RFC4828].

The resulting open questions are:

- How does TFRC-SP operate under various network conditions?
- How to design congestion control so as to scale with packet size (dependency of congestion algorithm on packet size)? Early assessment shows that packet size dependency should remain at the transport layer.

Today, many network resources are designed so that packet processing cannot be overloaded even for incoming loads at the maximum bit-rate of the line. If packet processing can handle sustained load r [packet per second] and the minimum packet size is h [bit] (i.e. packet headers with no payload), then a line rate of x [bit per second] will never be able to overload packet processing as long as x <= r.h. However, realistic equipment is often designed to only cope with a near-worst-case workload with a few larger packets in the mix, rather than the worst-cast of all minimum size packets. In this case, x = r.(h + e) for some small value of e.

Therefore, it is likely that most congestion seen on today's Internet is due to an excess of bits rather than packets, although packet-congestion is not impossible for runs of small packets (e.g. TCP ACKs or DoS attacks with small UDP datagrams).

This observation raises additional open issues:

- Will bit congestion remain prevalent?

Being able to assume that congestion is generally due to excess bits not excess packets is a useful simplifying assumption in the design of congestion control protocols. Can we rely on this assumption into the future?

Over the last three decades, performance gains have mainly been through increased packet rates, not bigger packets. But if bigger maximum segment sizes become more prevalent, tiny segments (e.g. ACKs) will still continue to be widely used - a widening range of packet sizes.

The open question is thus whether or not packet processing rates (r) will keep up with growth in transmission rates (x). A superficial look at Moore's Law type trends would suggest that processing (r) will continue to outstrip growth in transmission (x). But predictions based on actual knowledge of technology futures would be useful. Another open question is whether there are likely to be more small packets in the average packet mix. If the answers to either of these questions predict that packet congestion could become prevalent, congestion control protocols will have to be more complicated.

- Confusable Causes of Drop

There is a considerable body of research on how to distinguish whether packet drops are due to transmission corruption or to congestion. But the full list of confusable causes of drop is longer and includes transmission loss, congestion loss (bit congestion and packet congestion), and policing loss.

If congestion is due to excess bits, the bit rate should be reduced. If congestion is due to excess packets, the packet rate can be reduced without reducing the bit rate - by using larger packets. However, if the transport cannot tell which of these causes led to a specific drop, its only safe response is to reduce the bit rate. This is why the Internet would be more complicated if packet congestion were prevalent, as reducing the bit rate normally also reduces the packet rate, while reducing the packet rate doesn't necessarily reduce the bit rate.

Given distinguishing between transmission loss and congestion is already an open issue (Section 3.2), if that problem is ever solved, a further open issue would be whether to standardize a solution that distinguishes all the above causes of drop, not just two of them.

Nonetheless, even if we find a way for network equipment to explicitly distinguish which sort of drop has occurred, we will never be able to assume that such a smart AQM solution is deployed at every congestible resource throughout the Internet - at every higher layer device like firewalls, proxies, servers and at every lower layer device like low-end home hubs, DSLAMs, WLAN cards, cellular base-stations and so on. Thus, transport protocols will always have to cope with drops due to unguessable causes, so we should always treat AQM smarts as an optimization, not a given.

- What does a congestion notification on a packet of a certain size mean?

The open issue here is whether a loss or explicit congestion mark should be interpreted as a single congestion event irrespective of the size of the packet lost or marked, or whether the strength of the congestion notification is weighted by the size of the packet. This issue is discussed at length in [Bri08], along with other aspects of packet size and congestion control.

[Bri08] makes the strong recommendation that network equipment should drop or mark packets with a probability independent of each specific packet's size, while congestion controls should respond to dropped or marked packets in proportion to the packet's size. This issue is deferred to the Transport Area Working Group.

- Packet Size and Congestion Control Protocol Design

If the above recommendation is correct - that the packet size of a congestion notification should be taken into account when the transport reads, not when the network writes the notification - it opens up a significant program of protocol engineering and re-engineering. Indeed, TCP does not take packet size into account when responding to losses or ECN. At present this is not a pressing problem because use of 1500B data segments is very prevalent for TCP and the range of alternative segment sizes is not large. However, we should design the Internet's protocols so they will scale with packet size, so an open issue is whether we should evolve TCP, or expect new protocols to take over.

As we continue to standardize new congestion control protocols, we must then face the issue of how they should take account of packet size. If we determine that TCP was incorrect in not taking account of packet size, even if we don't change TCP, we should not allow new protocols to follow TCP's example in this respect. For example, as explained here above, the small-packet variant of TCP-friendly rate control (TFRC-SP [RFC4828]) is an experimental protocol that aims to take account of packet size. Whatever packet size it uses, it ensures its rate approximately equals that of a TCP using 1500B segments. This raises the further question of whether TCP with 1500B segments will be a suitable long-term gold standard, or whether we need a more thoroughgoing review of what it means for a congestion control to scale with packet size.

3.4 Challenge 4: Pseudo-Wires

Pseudowires (PW) may carry non-TCP data flows (e.g. TDM traffic). Structure Agnostic TDM over Packet (SATOP) [RFC4553], Circuit Emulation over Packet Switched Networks (CESoPSN), TDM over IP, are not responsive to congestion control in a TCP-friendly manner as prescribed by [RFC2914]. Moreover, it is not possible to simply reduce the flow rate of a TDM PW when facing packet loss.

Carrying TDM PW over an IP network poses a real problem. Indeed, providers can rate control corresponding incoming traffic but it may not be able to detect that a PW carries TDM traffic. This can be illustrated with the following example.

```
              ...........       ............
             .           .     .
      S1 --- E1 ---      .     .
             .     |     .     .
             .      === E5 === E7 ---
             .     |     .     .     |
      S2 --- E2 ---      .     .     |
             .           .     .     |      |
              ...........      .     |      v
                               .      ----- R --->
              ...........      .     |      ^
             .           .     .     |      |
      S3 --- E3 ---      .     .     |
             .     |     .     .     |
             .      === E6 === E8 ---
             .     |     .     .
      S4 --- E4 ---      .     .
             .           .     .
              ...........       ............

             \---- P1 ---/     \---------- P2 -----
```

Sources S1, S2, S3 and S4 are originating TDM over IP traffic. P1 provider edges E1, E2, E3, and E4 are rate limiting such traffic. The SLA of provider P1 with transit provider P2 is such that the latter assumes a BE traffic pattern and that the distribution shows the typical properties of common BE traffic (elastic, non-real time, non-interactive).

The problem arises for transit provider P2 that is not able to detect that IP packets are carrying constant-bit rate service traffic that is by definition unresponsive to any congestion control mechanisms.

Assuming P1 providers are rate limiting BE traffic, a transit P2 provider router R may be subject to serious congestion as all TDM PWs cross the same router. TCP-friendly traffic would follow TCP's AIMD algorithm of reducing the sending rate in half in response to each packet drop. Nevertheless, the TDM PWs will take all the available capacity, leaving no room for any other type of traffic. Note that the situation may simply occur because S4 suddenly turns up a TDM PW.

As it is not possible to assume that edge routers will soon have the ability to detect the type of the carried traffic, it is important for transit routers (P2 provider) to be able to apply a fair, robust, responsive and efficient congestion control technique in order to prevent impacting normally behaving Internet traffic. However, it is still an open question how the corresponding mechanisms in the data and control planes have to be designed.

3.5 Challenge 5: Multi-domain Congestion Control

Transport protocols such as TCP operate over the Internet that is divided into autonomous systems. These systems are characterized by their heterogeneity as IP networks are realized by a multitude of technologies. The variety of conditions and their variations leads to correlation effects between policers that regulate traffic against certain conformance criteria.

With the advent of techniques allowing for early detection of congestion, packet loss is no longer the sole metric of congestion. ECN (Explicit Congestion Notification) marks packets - set by active queue management techniques - to convey congestion information trying to prevent packet losses (packet loss and the number of packets marked gives an indication of the level of congestion). Using TCP ACKs to feed back that information allows the hosts to realign their transmission rate and thus encourage them to efficiently use the network. In IP, ECN uses the two unused bits of the TOS field [RFC2474]. Further, ECN in TCP uses two bits in the TCP header that were previously defined as reserved [RFC793].

ECN [RFC3168] is an example of a congestion feedback mechanism from the network toward hosts, while the policer must sit at every potential point of congestion. The congestion-based feedback scheme however has limitations when applied on an inter-domain basis. Indeed, the same congestion feedback mechanism is required along the entire path for optimal control at end-systems.

Another solution in a multi-domain environment may be the TCP rate controller (TRC), a traffic conditioner which regulates the TCP flow at the ingress node in each domain by controlling packet drops and RTT of the packets in a flow. The outgoing traffic from a TRC controlled domain is shaped in such a way that no packets are dropped at the policer. However, the TRC depends on the end-to-end TCP model, and thus the diversity of TCP implementations is a general problem.

Security is another challenge for multi-domain operation. At some domain boundaries, an increasing number of application layer gateways (e. g., proxies) are deployed, which split up end-to-end connections and prevent end-to-end congestion control.

Furthermore, authentication and authorization issues can arise at domain boundaries whenever information is exchanged, and so far the Internet does not have a single general security architecture that could be used in all cases. Many autonomous systems also only exchange some limited amount of information about their internal state (topology hiding principle), even though having more precise information could be highly beneficial for congestion control. The future evolution of the Internet inter-domain operation has to show whether more multi-domain information exchange can be realized.

3.6 Challenge 6: Precedence for Elastic Traffic

Traffic initiated by so-called elastic applications adapt to the available bandwidth using feedback about the state of the network. For all these flows the application dynamically adjusts the data generation rate. Examples encompass short-lived elastic traffic including HTTP and instant messaging traffic as well as long file transfers with FTP. In brief, elastic data applications can show extremely different requirements and traffic characteristics.

The idea to distinguish several classes of best-effort traffic types is rather old, since it would be beneficial to address the relative delay sensitivities of different elastic applications. The notion of traffic precedence was already introduced in [RFC791], and it was broadly defined as "An independent measure of the importance of this datagram."

For instance, low precedence traffic should experience lower average throughput than higher precedence traffic. Several questions arise here: what is the meaning of "relative"? What is the role of the Transport Layer?

The preferential treatment of higher precedence traffic with appropriate congestion control mechanisms is still an open issue that may, depending on the proposed solution, impact both the host and the network precedence awareness, and thereby congestion control. [RFC2990] points out that interactions between congestion control and DiffServ [RFC2475] have yet to be addressed, and this statement is still valid at the time of writing.

There is also still work to be performed regarding lower precedence traffic - data transfers which are useful, yet not important enough to significantly impair any other traffic. Examples of applications that could make use of such traffic are web caches and web browsers (e.g. for pre-fetching) as well as peer-to-peer applications. There are proposals for achieving low precedence on a pure end-to-end basis (e.g. TCP-LP [Kuzmanovic03]), and there is a specification for achieving it via router mechanisms [RFC3662]. It seems, however, that such traffic is still hardly used, and sending lower precedence data is not yet a common service on the Internet.

3.7 Challenge 7: Misbehaving Senders and Receivers

In the current Internet architecture, congestion control depends on parties acting against their own interests. It is not in a receiver's interest to honestly return feedback about congestion on the path, effectively requesting a slower transfer. It is not in the sender's interest to reduce its rate in response to congestion if it can rely on others to do so. Additionally, networks may have strategic reasons to make other networks appear congested.

Numerous strategies to divert congestion control have already been identified. The IETF has particularly focused on misbehaving TCP receivers that could confuse a compliant sender into assigning excessive network and/or server resources to that receiver (e.g. [Sav99], [RFC3540]). But, although such strategies are worryingly powerful, they do not yet seem common.

A growing proportion of Internet traffic comes from applications designed not to use congestion control at all, or worse, applications that add more forward error correction the more losses they experience. Some believe the Internet was designed to allow such freedom so it can hardly be called misbehavior. But others consider that it is misbehavior to abuse this freedom [RFC3714], given one person's freedom can constrain the freedom of others (congestion represents this conflict of interests). Indeed, leaving freedom unchecked might result in congestion collapse in parts of the Internet. Proportionately, large volumes of unresponsive voice traffic could represent such a threat, particularly for countries with less generous provisioning [RFC3714]. More recently, Internet video on demand services are becoming popular that transfer much greater data rates without congestion control (e.g. the peer-to-peer Joost service currently streams media over UDP at about 700kbps downstream and 220kbps upstream).

Note that the problem is not just misbehavior driven by a selfish desire for more bandwidth (see Section 4).

Open research questions resulting from these considerations are:

- By design, new congestion control protocols need to enable one end to check the other for protocol compliance.
- Provide congestion control primitives that satisfy more demanding applications (smoother than TFRC, faster than high speed TCPs), so that application developers and users do not turn off congestion control to get the rate they expect and need.

Note also that self-restraint is disappearing from the Internet. So, it may no longer be sufficient to rely on developers/users voluntarily submitting themselves to congestion control. As main consequence, mechanisms to enforce fairness (see Section 2.3) need to have more emphasis within the research agenda.

3.8 Other challenges

This section provides additional challenges and open research issues that are not (at this point in time) deemed very large or of different nature compared to the main challenges depicted since so far.

Note that this section may be complemented in future release of this document by topics discussed during the last ICCRG meeting, co-located with PFLDNet 2008 International Workshop. Topics of interest include multipath congestion control, and congestion control for multimedia codecs that only support certain set of data rates.

3.8.1 RTT estimation

Several congestion control schemes have to precisely know the round-trip time (RTT) of a path. The RTT is a measure of the current delay on a network. It is defined as the delay between the sending of a packet and the reception of a corresponding response, which is echoed back immediately by receiver upon receipt of the packet. This corresponds to the sum of the one-way delay of the packet and the (potentially different) one-way delay of the response. Furthermore, any RTT measurement also includes some additional delay due to the packet processing in both end-systems.

There are various techniques to measure the RTT: Active measurements inject special probe packets to the network and then measure the response time, using e.g. ICMP. In contrast, passive measurements determine the RTT from ongoing communication processes, without sending additional packets.

The connection endpoints of reliable transport protocols such as TCP, SCTP, and DCCP, as well as several application protocols, keep track of the RTT in order to dynamically adjust protocol parameters such as the retransmission timeout (RTO). They can implicitly measure the RTT on the sender side by observing the time difference between the sending of data and the arrival of the corresponding acknowledgements. For TCP, this is the default RTT measurement procedure, in combination with Karn's algorithm that prohibits RTT measurements from retransmitted segments [RFC2988]. Traditionally, TCP implementations take one RTT measurement at a time (i. e., about once per RTT). As alternative, the TCP timestamp option [RFC1323] allows more frequent explicit measurements, since a sender can safely obtain an RTT sample from every received acknowledgment. In principle, similar measurement mechanisms are used by protocols other than TCP.

Sometimes it would be beneficial to know the RTT not only at the sender, but also at the receiver. A passive receiver can deduce some information about the RTT by analyzing the sequence numbers of received segments. But this method is error-prone and only works if the sender permanently sends data. Other network entities on the path can apply similar heuristics in order to approximate the RTT of a connection, but this mechanism is protocol-specific and requires per-connection state. In the current Internet, there is no simple and safe solution to determine the RTT of a connection in network entities other than the sender.

As outlined earlier in this document, the round-trip time is typically not a constant value. For a given path, there is theoretical minimum value, which is given by the minimum transmission, processing and propagation delay on that path. However, additional variable delays might be caused by congestion, cross-traffic, shared mediums access control schemes, recovery procedures, or other sub-IP layer mechanisms. Furthermore, a change of the path (e. g., route flipping, handover in mobile networks) can result in completely different delay characteristics.

Due to this variability, one single measured RTT value is hardly sufficient to characterize a path. This is why many protocols use RTT estimators that derive an averaged value and keep track of a certain history of previous samples. For instance, TCP endpoints derive a smoothed round-trip time (SRTT) from an exponential weighted moving average [RFC2988]. Such a low-pass filter ensures that measurement noise and single outliers do not significantly affect the estimated RTT. Still, a fundamental drawback of low-pass filters is that the averaged value reacts slower to sudden changes of the measured RTT. There are various solutions to overcome this effect: For instance, the standard TCP retransmission timeout calculation considers not only the SRTT, but also a measure for the variability of the RTT measurements [RFC2988]. Since this algorithm is not well-suited for frequent RTT measurements with timestamps, certain implementations modify the weight factors (e.g., [SK02]). There are also proposals for more sophisticated estimators, such as Kalman filters or estimators that utilize mainly peak values.

However, open questions concerning RTT estimation in the Internet remain:

- Optimal measurement frequency: Currently, there is no common understanding of the right time scale of RTT measurement. In particular, the necessity of rather frequent measurements (e.g., per packet) is not well understood. There is some empirical evidence that such frequent sampling may not have a significant benefit [Allman99].

- Filter design: A closely related question is how to design good filters for the measured samples. The existing algorithms are known to be robust, but they are far from being perfect. The fundamental problem is that there is no single set of RTT values that could characterize the Internet as a whole, i.e., it is hard to define a design target.

- Default values: RTT estimators can fail in certain scenarios, e. g., when any feedback is missing. In this case, default values have to be used. Today, most default values are set to conservative values that may not be optimal for most Internet communication. Still, the impact of more aggressive settings is not well understood.

- Clock granularities: RTT estimation depends on the clock granularities of the protocol stacks. Even though there is a trend towards higher precision timers, the limited granularity may still prevent highly accurate RTT estimations.

3.8.2 Malfunctioning devices

There is a long history of malfunctioning devices harming the deployment of new and potentially beneficial functionality in the Internet. Sometimes, such devices drop packets when a certain mechanism is used, causing users to opt for reliability instead of performance and disable the mechanism, or operating system vendors to disable it by default. One well-known example is ECN, whose deployment was long hindered by malfunctioning firewalls, but there are many other examples (e.g. the Window Scaling option of TCP).

As new congestion control mechanisms are developed with the intention of eventually seeing them deployed in the Internet, it would be useful to collect information about failures caused by devices of this sort, analyze the reasons for these failures, and determine whether there are ways for such devices to do what they intend to do without causing unintended failures. Recommendation for vendors of these devices could be derived from such an analysis. It would also be useful to see whether there are ways for failures caused by such devices to become more visible to endpoints, or for those failures to become more visible to the maintainers of such devices.

4. Security Considerations

Misbehavior may be driven by pure malice, or malice may in turn be driven by wider selfish interests, e.g. using distributed denial of service (DDoS) attacks to gain rewards by extortion [RFC4948]. DDoS attacks are possible both because of vulnerabilities in operating systems and because the Internet delivers packets without requiring congestion control.

To date, compliance with congestion control rules and being fair requires end points to cooperate. The possibility of uncooperative behavior can be regarded as a security issue; its implications are discussed throughout these documents in a scattered fashion.

Currently the focus of the research agenda against denial of service is about identifying attack packets, attacking machines and networks hosting them, with a particular focus on mitigating source address spoofing. But if mechanisms to enforce congestion control fairness were robust to both selfishness and malice [Bri06] they would also naturally mitigate denial of service, which can be considered (from the perspective of well-behaving Internet user) as a congestion control enforcement problem.

5. Contributors

This document is the result of a collective effort to which the following people have contributed:

Dimitri Papadimitriou <dimitri.papadimitriou@alcatel-lucent.be>
Michael Welzl <michael.welzl@uibk.ac.at>
Wesley Eddy <weddy@grc.nasa.gov>
Bela Berde <bela.berde@gmx.de>
Paulo Loureiro <loureiro.pjg@gmail.com>
Chris Christou <christou_chris@bah.com>
Michael Scharf <michael.scharf@ikr.uni-stuttgart.de>

6. References

6.1 Normative References

[RFC791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC793, September 1981.

[RFC896] Nagle, J., "Congestion Control in IP/TCP", RFC 896, January 1984.

[RFC1323] Jacobson, V., Braden, R., Borman, D., "TCP Extensions for High Performance", RFC 1323, May 1992.

[RFC1958] B. Carpenter, Ed., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC2309] Braden, B., et al., "Recommendations on queue management and congestion avoidance in the Internet", RFC 2309, April 1998.

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 1633, October 1996.

[RFC2474] Nichols, K., Blake, S. Baker, F. and D. Black, "Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers", RFC 2474, December 1998.

[RFC2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Services", RFC 2475, December 1998.

[RFC2581] Allman, M., Paxson, V., and W. Stevens, "TCP Congestion Control", RFC 2581, April 1999.

[RFC2914] Floyd, S., "Congestion Control Principles", BCP 41, RFC 2914, September 2000.

[RFC2988] Paxson, V. and Allman, M., "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[RFC2990] Huston, G., "Next Steps for the IP QoS Architecture", RFC 2990, November 2000.

[RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, September 2001.

[RFC3448] Handley, M., Floyd, S., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 3448, January 2003.

[RFC3540] N. Spring, D. Wetherall, "Robust Explicit Congestion Notification (ECN) Signaling with Nonces", RFC 3540, June 2003.

[RFC3662] Roland Bless, Kathleen Nichols, Klaus Wehrle, "A Lower Effort Per-Domain Behavior for Differentiated Services", RFC 3662, December 2003.

[RFC3714] S. Floyd, Ed., J. Kempf, Ed. "IAB Concerns Regarding Congestion Control for Voice Traffic in the Internet", RFC 3714, March 2004.

[RFC3985] Bryant, S. and P. Pate, "Pseudo Wire Emulation Edge-to-

Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4340] Kohler, E., Handley, M., and S. Floyd, "Datagram Congestion Control Protocol (DCCP)", RFC 4340, March 2006.

[RFC4341] Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 2: TCP-like Congestion Control", RFC 4341, March 2006.

[RFC4342] Floyd, S., Kohler, E., and J. Padhye, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC)", RFC 4342, March 2006.

[RFC4553] Vainshtein, A. and Y. Stein, "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", RFC 4553, June 2006.

[RFC4614] Duke, M., R. Braden, R., Eddy, W., and Blanton, E., "A Roadmap for Transmission Control Protocol (TCP) Specification Documents", RFC 4614, September 2006.

[RFC4782] Floyd, S., Allman, M., Jain, A., and P. Sarolahti, "Quick-Start for TCP and IP", RFC 4782, Jan. 2007.

[RFC4948] L. Andersson, E. Davies, L. Zhang, "Report from the IAB workshop on Unwanted Traffic March 9-10, 2006", RFC 4948, August 2007.

[RFC5033] S. Floyd, M. Allman, "Specifying New Congestion Control Algorithms", RFC 5033, Aug. 2007.

6.2 Informative References

[Allman99] Allman, M. and V. Paxson, "On Estimating End-to-End Network Path Properties", Proc. SIGCOMM, Sept. 99.

[Andrew00] L. Andrew, B. Wydrowski and S. Low, "An Example of Instability in XCP", Manuscript available at <http://netlab.caltech.edu/maxnet/XCP_instability.pdf>

[Ath01] S. Athuraliya, S. Low, V. Li, and Q. Yin, "REM: Active queue management", IEEE Network Magazine, vol.15, no.3, pp.48-53, May 2001.

[Balan01] Balan, R. K., Lee, B.P., Kumar, K.R.R., Jacob, L., Seah, W.K.G., Ananda, A.L., "TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links", Proceedings of IEEE Infocom, Anchorage, Alaska, April 2001.

[Bonald00] T. Bonald, M. May, and J.-C. Bolot, "Analytic Evaluation of RED Performance", Proceedings of IEEE INFOCOM, Tel Aviv, Israel, March 2000.

[Bri07] Bob Briscoe, "Flow Rate Fairness: Dismantling a Religion" ACM SIGCOMM Computer Communication Review 37(2) 63--74 (April 2007).

[Bri06] Bob Briscoe, "Using Self-interest to Prevent Malice; Fixing the Denial of Service Flaw of the Internet," Workshop on the Economics of Securing the Information Infrastructure (Oct 2006) <http://wesii.econinfosec.org/draft.php?paper_id=19>

[Chester04] Chesterfield, J., Chakravorty, R., Banerjee, S., Rodriguez, P., Pratt, I. and Crowcroft, J., "Transport level optimisations for streaming media over wide-area wireless networks", WIOPT'04, March 2004.

[Chiu89] D. M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN Systems, vol. 17, pp.1-14, 1989.

[Clark98] D. Clark and W. Fang, "Explicit Allocation of Best-Effort Packet Delivery Service", IEEE/ACM Transactions on Networking, vol.6, no.4, pp.362-373, August 1998.

[Dukki06] N. Dukkipati and N. McKeown, "Why Flow-Completion Time is the Right Metric for Congestion Control", ACM SIGCOMM Computer Communication Review Volume 36, issue 1, Jan. 2006.

[Floyd93] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Trans. on Networking, vol.1, no.4, pp.397-413, Aug. 1993.

[Falk07] A. Falk et al "Specification for the Explicit Control Protocol (XCP)", Work in Progress, draft-falk-xcp-spec-03.txt, July 2007.

[Firoiu00] V. Firoiu and M. Borden, "A Study of Active Queue Management for Congestion Control," Proceedings of IEEE INFOCOM, Tel Aviv, Israel, March 2000.

[Floyd94] S. Floyd, "TCP and Explicit Congestion Notification", ACM Computer Communication Review, vol.24, no.5, October 1994, pp.10-23.

[Hollot01] C. Hollot, V. Misra, D. Towsley, and W.-B. Gong, "A Control Theoretic Analysis of RED", Proceedings of IEEE INFOCOM, Anchorage, Alaska, April 2001.

[Jacobson88] V. Jacobson, "Congestion Avoidance and Control", Proc. of the ACM SIGCOMM'88 Symposium, pp.314-329, August 1988.

[Jain88] R. Jain and K. Ramakrishnan, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals, and Methodology", Proceedings of IEEE Computer Networking Symposium: proceedings, Sheraton National Hotel, Washington, DC area, April 11-13, 1988.

[Jain90] R. Jain, "Congestion Control in Computer Networks: Trends and Issues", IEEE Network, May 1990, pp.24-30, ISSN 0890-8044.

[Jin04] Chen Jin, David X. Wei and Steven Low "FAST TCP: Motivation, Architecture, Algorithms, Performance", In Proc. IEEE Conference on Computer Communications Infocomm'04, March 2004.

[Katabi02] D. Katabi, M. Handley, and C. Rohr, "Internet Congestion Control for Future High Bandwidth-Delay Product Environments", Proceedings of the ACM SIGCOMM'02 Symposium, pp.89-102, August 2002.

[Kelly98] F. Kelly, A. Maulloo, and D. Tan, "Rate control in communication networks: shadow prices, proportional fairness, and stability", Journal of the Operational Research Society, vol.49, pp.237-252, 1998.

[Keshav07] S. Keshav, "What is congestion and what is congestion control", Presentation at IRTF ICCRG Workshop, Pfldnet 2007, (Los Angeles), California, February 2007.

[Krishnan04] R. Krishnan, J. Sterbenz, W. Eddy, C. Partridge, and M. Allman, "Explicit Transport Error Notification (ETEN) for Error-Prone Wireless and Satellite Networks", Computer Networks, vol.46, no.3, October 2004.

[Kuzmanovic03] A.Kuzmanovic and E.W.Knightly, "TCP-LP: A Distributed Algorithm for Low Priority Data Transfer", Proceedings of IEEE INFOCOM 2003, San Francisco, CA, April 2003.

[Low05] S. Low, L. Andrew and B. Wydrowski. "Understanding XCP: equilibrium and fairness", Proceedings of IEEE Infocom, Miami, USA, March 2005.

[Low03.2] S. Low, F. Paganini, J. Wang, and J. Doyle, "Linear stability of TCP/RED and a scalable control", Computer Networks Journal, vol.43, no.5, pp.633-647, December 2003.

[Low03.1] S. Low, "A duality model of TCP and queue management algorithms", IEEE/ACM Trans. on Networking, vol.11, no.4, pp.525-536, August 2003.

[Low02] S. Low, F. Paganini, J. Wang, S. Adlakha, and J. C. Doyle, "Dynamics of TCP/RED and a Scalable Control", Proceedings of IEEE Infocom, New York, USA, June 2002.

[Mascolo01] Saverio Mascolo, Claudio Casetti, Mario Gerla, M. Y. Sanadidi, Ren Wang, "TCP westwood: Bandwidth estimation for enhanced transport over wireless links", Proceedings of MOBICOM 2001, pp.287-297.

[Moors02] T. Moors, "A critical review of End-to-end arguments in system design", Proc. International Conference on Communications (ICC), Apr./May 2002.

[MKMV95] MacKie-Mason, J. and H. Varian, "Pricing Congestible Network Resources", IEEE Journal on Selected Areas in Communications, Advances in the Fundamentals of Networking' 13(7)1141--1149, 1995, <http://www.sims.berkeley.edu/~hal/Papers/pricing-congestible.pdf>.

[Padhye98] Padhye, J., Firoiu, V., Towsley, D., Kurose, J., Modeling TCP Throughput: A Simple Model and Its Empirical Validation, UMASS CMPSCI Tech Report TR98-008, Feb. 1998.

[Pan00] R. Pan, B. Prabhakar, and K. Psounis, "CHOKe: a stateless AQM scheme for approximating fair bandwidth allocation", In Proceedings of IEEE Infocom, Tel Aviv, Israel, March 2000.

[Rossi06] Rossi, M., "Evaluating TCP with Corruption Notification in an IEEE 802.11 Wireless LAN", master thesis, University of Innsbruck, November 2006. Available from http://www.welzl.at/research/projects/corruption/

[Sarola02] Sarolahti, P. and Kuznetsov, A., "Congestion Control in Linux TCP", Proc. USENIX Annual Technical Conference, June 2002.

[Savage99] Savage, S., Wetherall, D., and T. Anderson, "TCP Congestion Control with a Misbehaving Receiver", ACM SIGCOMM Computer Communication Review (1999).

[Saltzer84] Saltzer, J., Reed, D., and Clark, D. D. "End-to-end arguments in system design", ACM Transactions on Computer Systems 2, 4 (Nov. 1984).

[TRILOGY] "Trilogy Project", European Commission Seventh Framework Program Contract Number: INFSO-ICT-216372 <http://www.trilogy-project.org>

[Welzl03] M. Welzl, "Scalable Performance Signalling and Congestion Avoidance", Springer, August 2003. ISBN 1-4020-7570-7.

[Welzl08] M. Welzl, M. Rossi, A. Fumagalli, and M. Tacca, "TCP/IP over IEEE 802.11b WLAN: the Challenge of Harnessing Known-Corrupt Data", In Proceedings of IEEE ICC 2008, 19-23 May 2008, Beijing, China.

[Zhang03] H. Zhang, C. Hollot, D. Towsley, and V. Misra. "A Self-Tuning Structure for Adaptation in TCP/AQM Networks", SIGMETRICS'03, June 10-14, 2003, San Diego, California, USA.

Acknowledgments

The authors would like to thank the following people whose feedback and comments contributed to this document: Keith Moore, Jan Vandenabeele.

Larry Dunn (his comments at the Manchester ICCRG and discussions with him helped with the section on packet-congestibility). Bob Briscoe's contribution was partly funded by [TRILOGY], a research project supported by the European Commission.

Author's Addresses

Michael Welzl
University of Innsbruck
Technikerstr 21a
A-6020 Innsbruck, Austria
Phone: +43 (512) 507-6110

Email: michael.welzl@uibk.ac.at

Dimitri Papadimitriou
Alcatel-Lucent
Copernicuslaan, 50
B-2018 Antwerpen, Belgium
Phone: +32 3 240 8491
Email: dimitri.papadimitriou@alcatel-lucent.be

Michael Scharf
University of Stuttgart
Pfaffenwaldring 47
D-70569 Stuttgart
Germany
Phone: +49 711 685 69006
Email: michael.scharf@ikr.uni-stuttgart.de

Bob Briscoe
BT & UCL
B54/77, Adastral Park
Martlesham Heath
Ipswich IP5 3RE
UK
Email: bob.briscoe@bt.com

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).