Speeding up the 3D Web: A Case for Fast Startup Congestion Control

Michael Scharf[1], Mike Eissele[2], Christian Mueller[1], Thomas Ertl[2]
University of Stuttgart
[1] Institute of Communication Networks and Computer Engineering (IKR)
[2] Visualization and Interactive Systems Group (VIS)
michael.scharf@ikr.uni-stuttgart.de

ABSTRACT

More and more Web applications interactively display three dimensional (3D) real or virtual worlds. These applications typically require broadband network connectivity, since voluminous content must be transported over the Internet in a timely fashion. This paper studies how transport protocols can be optimized to support such network-demanding 3D Web applications. A review of the state-of-the-art shows that they differ from traditional Web applications. We discuss typical characteristics of these applications and highlight where and how they can benefit from enhancements of the Transmission Control Protocol (TCP). Our main conclusion is that 3D visualization applications can be an important use case for new fast startup congestion control mechanisms, such as the Quick-Start TCP extension and the Jump-Start proposal. Our claim is backed by experiments that combine a prototypical 3D visualization application with fast startup congestion control schemes implemented in the Linux network stack. As expected, our measurement results confirm significant performance benefits of Jump-Start and Quick-Start TCP.

1. INTRODUCTION

In the last years new 3D applications have emerged, which provide rich visualizations and graphical renderings of real or virtual worlds. Applications such as Google Earth or Microsoft's Virtual Earth foreshadow a new class of networked, multi-purpose, interactive 3D applications. Many ongoing research and development activities are driven by the vision that it will be possible to create a new "World Wide Space" [1] or "3D Web" [2] that offers ubiquitous access to detailed representations of the real [3] or virtual world [4] and that combine 3D city models with other data sources, location-based services, and potentially even live data.

Several problems have still to be solved in order to realize such a 3D Web. One challenge are the communication characteristics: The interactive 3D visualization usually does not solely operate on *offline* content, but instead retrieves data *online* in a streaming-like fashion over the Internet. Due to the data volumes of the 3D models and the interactivity of the applications, this communication is network-demanding and typically requires broadband Internet connectivity.

The timely and reliable transport of large volumes of data imposes challenges for the Internet transport protocols, in particular the Transmission Control Protocol (TCP). This paper analyzes the transport layer implications of interactive 3D applications, which have hardly been surveyed systematically so far. We discuss whether new protocol mechanisms would be useful, in particular new Internet congestion control mechanisms [5]. Specifically, we show that the emerging 3D Web would be a relevant use case for new fast startup congestion mechanisms, such as the Quick-Start TCP extension [6, 7] or Jump-Start [8]. Our motivation is to study these enhancements from the perspective of such new types of applications, given that TCP's Slow-Start is known to perform reasonably well for many current Web usage scenarios [9]. Our experimental results have been obtained with a prototypical interactive 3D visualization application and our Linux network stack implementations of fast startup congestion control schemes.

The rest of this paper is structured as follows. Section 2 reviews 3D Web technologies. Section 3 surveys related work concerning transport mechanisms for 3D content. In Section 4 we analyze the requirements of interactive 3D visualization and discuss the implications on transport protocols. Section 5 presents experimental results that show the benefit of fast startup congestion control for an interactive 3D visualization tool. Finally, Section 6 concludes the paper.

2. TOWARDS THE 3D WEB

2.1 3D Visualization Technologies

3D content plays an important role in several emerging applications. On modern desktop systems, high-speed 3D hardware acceleration is state-of-the-art and extensively used, in particular by computer games. Mobile devices have more limitations in screen sizes, computation, electrical power, and memory. Still, more and more mobile devices have enough resources to locally perform 3D rendering tasks by using hardware-accelerated 3D graphics. An example for a visualization on a mobile device can be seen in Fig. 1. Most visualization applications are *interactive*, i. e., users can change the visualized scenery, move around, rotate objects, or zoom in/out.

Figure 1: High-quality rendering of a 3D city model on an O2 XDA Flame. The textured model has 23k polygons and renders at 5 frames per second.

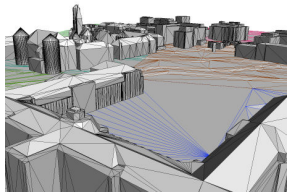

Figure 2: Same 3D model without texture images

There are different standards to describe and store 3D geometrical models. Descriptive textual formats to encode geometric 3D models are widely used, in particular based on Extensible Markup Language (XML) formats [3]. Examples include X3D, GML/CityGML, KML, or Collada. The main advantages of the XML formats are the ease of use and the extensibility. They are therefore preferred for data exchange in heterogeneous environments. However, their flexibility comes at the cost of huge data sizes. The description of physical scenarios also typically covers several *levels of detail* (cf. Figures 1 and 2).

In contrast to the standardized 3D *storage* formats, the *transport* of 3D models between servers and clients is often realized with proprietary, application-specific, or even encrypted formats and protocols. 3D content has also been considered by video standardization bodies [10], envisioning an integration of video streams and 3D virtual worlds. But this has not gained much acceptance so far. Similar to multi-layer video codecs, 3D models could also be streamed by using the concept of progressive meshes [11]. The idea of progressive meshes is to first transport a coarse object description, which allows an initial rendering. As more data becomes available, the visualization is refined.

2.2 Application and System Architectures

3D Web applications are typically realized as a client-server solution with a viewer application on the client and a back-end server infrastructure [2]. The rendering of 3D models is a computationally complex process and can be realized by two different approaches: Local rendering and remote rendering [12]. Traditionally, 3D graphics pipelines are realized as *local rendering*. The client device generates images of the 3D scenery and therefore must have sufficient graphic processing power. All content to be visualized has to be available at the client, i. e., it must be requested from the server first and temporally stored on the client. Once all required data is available on the client, it can be visualized in a highly interactive and flexible way.

An alternative solution is *remote rendering*. In this case, the complex rendering of 3D scenes is performed on a server. Any user interaction is sent to the server, triggers a new rendering, and the result is transmitted back to the client and is displayed there. This communication can be realized either by a download of a still image or by video streaming. Actually, remote rendering is quite similar to video streaming, except that the content is interactively created on the server. The advantage of remote rendering is that even complex 3D scenarios can be visualized on simple devices without 3D hardware acceleration. However, this comes at the cost of high server-side processing demands and an inherent delay lag between user interactions and the visualization. It depends on the content and user activity patterns whether the network traffic compared to local rendering is reduced or increased. This trade-off is highly application-specific.

3. SURVEY OF EXISTING APPROACHES

3.1 Classification

A multitude of interactive 3D visualization applications exists in many application areas. Concerning the realization of the data transport, two different classes can be distinguished: *Web-like* schemes basically use the World Wide Web (WWW) technology with Hypertext Transfer Protocol (HTTP) request-response communication using one or multiple parallel TCP connections. Its main advantage is the ubiquitous availability. In contrast, other solutions are close to linear multimedia *streaming*, which is often realized by specific protocols on top of the User Datagram Protocol (UDP), in particular if timely arrival of data is more important than reliable transport.

An interesting aspect of 3D virtual worlds is that they combine characteristics of traditional Web and multimedia streaming. A survey of different solutions in Table 1 illustrates that both communication paradigms are used by applications dealing with 3D models. This table focuses in particular on 3D city models and is by far not a comprehensive list of all related work. For instance, application types with similar characteristics exist in the field of telemedicine or massively multiplayer online games, too.

3.2 Usage of Transport Protocols

Almost all transport protocol usage patterns have been proposed or are in use by 3D visualization applications: For instance, Google Earth is one out of several popular virtual worlds that visualize high-resolution aerial images in combination with other data, such as 3D city models. The application is realized as a typical client-server Web solution, i. e., the content is retrieved by HTTP over multiple TCP connections, using ZIP compression. Related approaches, such as Microsoft's Virtual Earth or NASA World Wind, have similar characteristics. There are also ongoing efforts to realize 3D city model visualizations on mobile devices. One example is the m-LOMA project [13], which addresses the rather low bandwidth of cellular networks by a server-side preprocessing of content, sophisticated caching schemes in the client, and a view-dependent, hierarchical streaming of partial 3D models in a proprietary binary format over a single TCP connection. The Nexus project at the University of Stuttgart [1] has also developed similar visualization applications, e. g., for 3D city models (cf. Section 5.2).

Table 1: Comparison of selected approaches to transport and visualize 3D worlds

Example	Target environment	System architecture	Transport protocol usage
Web-like			
Google Earth, Microsoft Virtual Earth, etc.	Standalone clients or browser plugins	Local rendering e. g. of Keyhole Markup Language (KML) content	HTTP over multiple TCP conn. ZIP data compression
m-LOMA project [13]	Smart phones with OpenGL ES support	Local rendering with server-side preprocessing	One TCP connection Compact binary transport format
Nexus example application (see Section 5.2)	Standalone client	Local rendering with server-side preprocessing	One TCP connection Binary transport format
Streaming-like			
Remote visualization (e. g., [14])	Mobile devices (PDAs)	Remote rendering	MPEG-2 stream over UDP Separate TCP control channel
IP multimedia subsystem (IMS) usage (e. g., [15])	IMS-enabled mobile devices	Local rendering plus video content Textures with different qualities	HTTP over TCP transport QoS negotiation by SIP signaling
Communication middleware (e. g., [16])	Mobile devices	Local rendering Progressive mesh encoding	Middleware over TCP and UDP Intelligent selection of TCP/UDP

Concerning streaming-like solutions, several research projects have developed remote visualization solutions, which mostly use UDP transport (see [12, 14] and references therein). Given the similarity to video streaming, the usage of network Quality-of-Service (QoS) mechanisms has been investigated. For instance, [15] demonstrates that 3D visualization applications could use bandwidth reservations in combination with application-level adaptation, including textures with different levels of detail. There is also work on middlewares that dynamically select whether to use a reliable transport protocol (TCP) or an unreliable one (UDP) [16]. These mechanisms could be used in combination with progressive meshes, since then certain parts of the 3D content could be transported unreliably.

4. TRANSPORT LAYER IMPLICATIONS

4.1 3D Application Requirements

Online 3D visualization applications are network-demanding: On the one hand, the user activities can frequently trigger downloads of content. As visualization lags can be immediately noticed by users, the responsiveness of the application is a key usability requirement. In the best case, new content is displayed almost instantaneously. On the other hand, encoding of complex 3D structures and texture images can sum up to large amounts of data, in particular if XML descriptions are used.

Thus, the vision of the 3D Web implies new non-linear multimedia applications. They are adaptive, but they require that rather large amounts of data are transported with small delays. There are two differences to other multimedia types like audio or video streams: First, the traffic is not a continuous stream, but can instead be bursty, since data retrieval is mainly triggered by user actions. And second, the data transport of complex 3D world descriptions usually has to be reliable, in particular when XML-encoded content is locally rendered at the client. In contrast, multimedia codecs are often loss-tolerant.

4.2 Workload Characteristics

The description of a complex 3D scenery can be orders of magnitude larger than a typical Web site. The part of a 3D city model shown in Fig. 1 consists of about 23,000 polygons. A compact binary representation of the model, including the compressed textures, requires 16 MB of storage. The size and complexity of a 3D model can usually be reduced at cost of quality. For example, a significant size reduction can be achieved by omitting textures, as shown in Fig. 2. The same scene without textures requires only 950 kB. A further reduction would be possible by reducing the level of detail of the 3D structures. Still, even then the scenery would be much larger than most of today's Web sites. A similar effect has also been observed for 2D maps [17].

The delay-sensitive transport of large amounts of data requires broadband connectivity. Of course, there are several well-known mechanisms to reduce the bandwidth demand and to minimize communication. One mechanism is caching at the client. Another one is to use optimized encoding schemes. Plain-text XML is widely used to encode 3D content, but the XML representation of floating-point numbers is space-consuming. Plain text 3D formats are 4–14 times larger than binary representations. To some extent, lossless compression techniques can reduce the size of models and textures. Still, browsing interactively through complex, high-quality 3D scenarios retrieved from the Internet will always result in large and bursty data transfers.

4.3 A Case for TCP Enhancements?

The voluminous and time-critical communication raises the question whether the existing transport protocols offer optimal solutions for this class of applications. The various UDP-based application protocols indicate that TCP might not always be a perfect fit. From an application developer's perspective, 3D visualization applications could benefit from several features that the existing transport protocol implementations do not provide:

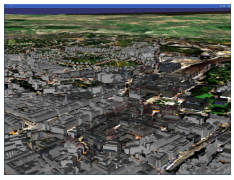

Figure 3: Screenshot of the Nexus application, showing a detailed 3D city model of Stuttgart

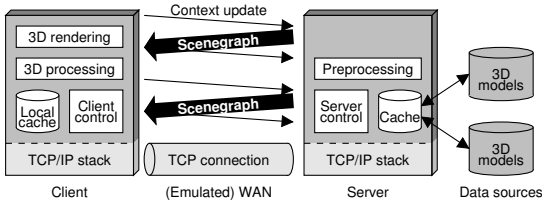

Figure 4: Client-server application architecture

Cross-layer adaptation interfaces: 3D visualization applications have a multitude of possibilities to control their communication behavior. They can present less level of detail (simpler 3D structures, smaller visibility range, etc.), or adapt the presentation (simpler textures, smaller frame rates, less requests for dynamic content, etc.). The adaptation decisions inside the application require knowledge about the network characteristics, such as the available bandwidth or an estimate how long the transport of a certain amount of data will probably last [18]. Such information might be available in the network stack, but the realization of corresponding application interfaces [19] is still an open issue.

Controlled reliability: Most parts of a 3D scenery is described by complex data structures that require reliable data delivery [18]. However, for dynamic status updates in-time delivery is more important than reliability. Enhanced encoding schemes do not necessarily require reliable transport of certain optional model parts. This could be a use case for transport protocols with partial reliability [20].

Advanced congestion control: Interactive 3D visualization applications could benefit from more flexible congestion control mechanisms. The required bandwidth easily reaches the operational area of state-of-the-art high-speed congestion control algorithms. Furthermore, more fine-grained differentiation or prioritization mechanisms among different flows between the same application instances could be used [18]. Congestion control protocols with some form of short-term predictability could be useful. And, last but not least, minimizing transport delays is a highly desirable feature, too. In this context, the Slow-Start of TCP's congestion control has been identified as an issue, e. g., by [13].

Some of these features can be realized by extensions of TCP, whereas others require other transport protocols. In the remainder of this paper we focus on the question how the congestion control can be optimized for the 3D Web.

5. FAST STARTUP SCHEMES

5.1 Fast Startup: What and Why?

The objective of fast startup congestion control is to use almost instantaneously the available bandwidth of a path. This requires a more aggressive behavior than the existing Slow-Start in the TCP congestion control, which may require a substantial amount of time until a path is fully utilized. Several fast startup schemes have been proposed. They can be roughly subdivided into end-to-end mechanisms that mainly affect the sender side and network-assisted congestion control mechanisms that assume additional processing inside routers. An example for the former class is Jump-Start [8]. In contrast, Quick-Start [6, 7] is an experimental TCP extension that uses additional network feedback. Due to the resulting deployment challenges, Quick-Start is currently not intended for the global Internet [6]. Both schemes have been evaluated extensively by simulations [7, 8]. They have also been implemented and compared by synthetic experiments (see [9] and references therein).

It is well-known that a significant speedup compared to Slow-Start is possible for "mid-sized" transfer sizes in the range of few dozens of kilobytes to several megabytes [7]. However, it has been questioned whether this specific range indeed represents relevant use cases, given that most Internet traffic either consists of mice or elephant flows. Only few studies with real applications have indeed demonstrated the theoretically predicted speedups. One purpose of this paper is to substantiate that such "mid-sized" transfer sizes are common if 3D models are incrementally transported.

5.2 Example of an 3D Application

As example we use an interactive 3D visualization application that has been developed within the Nexus project (see [1]). A screenshot is shown in Fig. 3. This application has the advantage that it is Linux-based and that it offers full access to the source code of both client and server, which facilitates instrumentation and experiments in controlled environments. The client application can visualize 3D city models using Open GL graphics acceleration. In order to retrieve the model data, the client periodically synchronizes its context with the server. If the user changes the visualization perspective, missing 3D model parts are retrieved. The data transport is realized over a single persistent TCP connection using a binary encoding format. The overall application architecture is sketched in Fig. 4. The architecture is open to other data sources, as the server can query different external databases to retrieve various 3D models that are available in XML formats.

5.3 Measurement Methodology

Both client and server run on state-of-the-art computers with an Ubuntu Linux and kernel version 2.6.24, which has been patched to implement either Jump-Start or Quick-Start TCP. We use basically the default stack configuration with the "Cubic" congestion control. The socket buffer sizes have been increased to 8 MB in order to avoid limitations by the TCP flow control. Client and server are interconnected by a 1 Gbit/s Ethernet segment. This represents a capacity over-provisioning scenario, since the applications cannot process the content at this speed. We use the Linux network emulation "NetEm" to enforce a minimum round-trip-time (RTT) of 200 ms. Such a large RTT could easily occur if the servers providing the 3D models are distributed around the world.

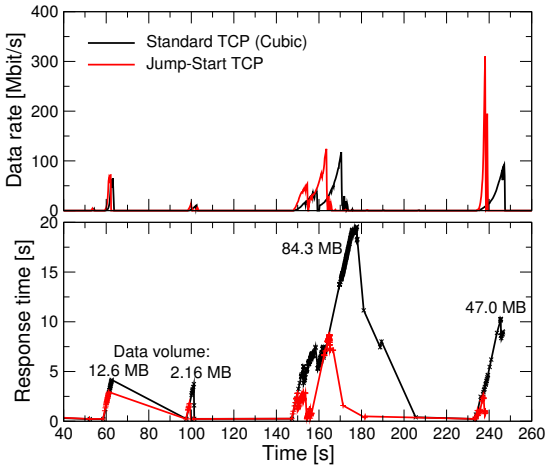

Figure 5: Traces for Jump-Start

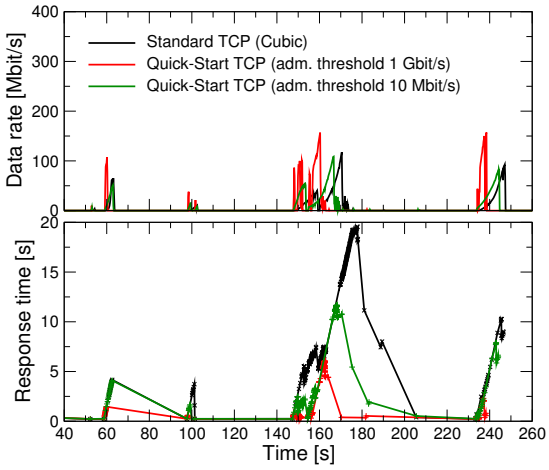

Figure 6: Traces for Quick-Start

We have instrumented client and server to measure the response times, being defined as the time between the sending of a context update of the client and the complete reception of the corresponding new scenegraphs as illustrated in Fig. 4. Furthermore, we analyze the communication by capturing "tcpdump" traces. In order to reproduce the results we measure with a recorded sequence of user interactions that browses through a couple of different locations where 3D city models are available. At the beginning of each measurement, the client cache is empty, i. e., the complete 3D scenery must be retrieved from the server.

5.4 Performance Results

Figure 5 shows one example of a resulting trace. The upper part of the diagram depicts the observed data rate between server and client as a function of the time in the recorded sequence of user interactions. The lower part reports the response times measured by the client application. Each measurement point refers to the delay between the complete download of a new 3D scenegraph object and the client message that triggered that transfer. Obviously, as long as the 3D scenery does not change, no data transfers occur. Due to the size of the displayed 3D models, the total amount of data exchanged during one measurement run is large. For instance, after about 150 s, a city model of Frankfurt is loaded and displayed, which includes many buildings. All objects of this model part sum up to more than 80 MB. Of course, this communication pattern is somehow futuristic since the large bandwidth required for interactive usage is hardly available end-to-end in the current Internet.

In Fig. 5 one can observe the typical TCP Slow-Start behavior, i. e., the sender starts to send with a small data rate, which then ramps up exponentially. As shown in the lower part, it can last up to 20 s until all parts of a new 3D model are retrieved, even though the link has a vast amount of free capacity. For this communication pattern, a congestion control with Slow-Start is apparently sub-optimal.

The performance of Jump-Start is also included in Fig. 5. With the selected parametrization, depending on the queued amount of application data, up to 65,535 bytes are played out during the first round-trip time [9]. Our implementation also automatically reactivates the Jump-Start mechanism after long idle times. In this experiment, the usage of Jump-Start significantly reduces the response times whenever data is transferred. In the best case, the model loading delay is reduced by over 10 s, which is a very significant reduction.

We have also repeated the same experiment with the Quick-Start TCP extension. Different to end-to-end schemes such as Jump-Start, Quick-Start is only activated if there is free capacity on the path, which reduces the risk of unfairness due to a fast startup [9]. Fig. 6 shows results for two configurations: In the first case it is assumed that the total link capacity is available to Quick-Start requests, while in the second case the "target algorithm" admission control [7] only approves Quick-Start requests up to an admission threshold of 10 Mbit/s. In both setups, the server asks for an initial sending rate of 82 Mbit/s, which is a reasonable value to transfer even large models within few seconds.

If the admission threshold is high, all Quick-Start requests throughout the experiment are approved, i. e., the initial sending rate is always of the order of 100 Mbit/s. Fig. 6 shows that in this case the content download times are indeed reduced to few seconds only and even smaller than the Jump-Start results. If the admission threshold is only 10 Mbit/s, the Quick-Start requests are reduced in the network stack. They could even be denied if there was cross-traffic. The corresponding results in Fig. 6 still show a performance improvement, i. e., it is sufficient to grant only a certain share of the link capacity to Quick-Start requests. But, of course, the maximum achievable speedup is then smaller. In the given scenario, the performance with a small admission threshold is worse than the performance with Jump-Start.

We also performed experiments with other high-speed congestion control algorithms. They revealed very similar results as long as the standard Slow-Start algorithm is used. Furthermore, we also repeated the experiment with smaller RTTs, such as 50 ms. As to be expected, the potential response time reduction by a fast startup scheme is then much smaller and hardly exceeds one second.

5.5 Discussion

One could argue that RTTs of the order of 200 ms may not be an important scenario for 3D virtual worlds, since large RTTs could be avoided by hosting the content inside a Content Delivery Network (CDN). However, it is currently unclear whether CDNs will indeed be able to deliver complex and potentially dynamic 3D models, in particular once they get synchronized with the real world. One could also consider the use of multiple parallel TCP connections, which is a well-known technique to circumvent the Slow-Start limitations. So far, our application does not use multiple connections, and our measurement results are therefore somehow provisional. We certainly do not claim that our results are universally valid for all types of online 3D visualization applications. Still, they probably illustrate typical communication patterns. Our experiments are also rudimentary in the sense that the interaction with application adaptation mechanisms and an intelligent application control of the fast startup activation are not considered so far. For example, request rates could be selected as a function of object sizes, but this is left for further study.

6. CONCLUSIONS AND OUTLOOK

3D Web applications exhibit communication patterns that are distinct from other application classes such as classic Web or video streaming applications. Their bandwidth requirements can be very high and latency is a critical parameter directly affecting usability. This paper studies the implications of such network-challenging applications. We identify a couple of transport protocol features that would be beneficial for online 3D visualization. As a specific example we study fast startup congestion control schemes, such as Jump-Start and Quick-Start TCP. We show with a prototype tool that applications dealing with 3D content can have characteristics that make fast startup schemes indeed useful. Depending on the scenario, speedups of up to 10 s can be achieved. Our results highlight that fast startup mechanisms should be considered in the future evolution of the Internet congestion control.

The main intention of this paper is to raise awareness of the demands of emerging 3D Web applications and to discuss their potential implications on transport protocols. It neither claims to be a complete survey of all communication issues concerning 3D worlds, nor to be a universal evaluation of the usefulness and risks of fast startup congestion control. Further work is required in both fields.

7. ACKNOWLEDGMENT

This work is funded by the German Research Foundation (DFG) through the Center of Excellence "Nexus – Spatial World Models for Mobile Context-Aware Applications".

8. REFERENCES

[1] R. Lange, N. Cipriani, L. Geiger, M. Großmann, H. Weinschrott, A. Brodt, M. Wieland, S. Rizou, and K. Rothermel, "Making the world wide space happen: New challenges for the Nexus context platform," in *Proc. IEEE PerCom*, Mar. 2009.

[2] N. Leavitt, "Browsing the 3D Web," *Computer*, vol. 39, no. 9, pp. 18–21, 2006.

[3] A. Altmaier and T. H. Kolbe, "Applications and solutions for interoperable 3D geo-visualization," in *Proc. Photogrammetric Week*, 2003.

[4] S. Kumar, J. Chhugani, C. Kim, D. Kim, A. Nguyen, P. Dubey, C. Bienia, and Y. Kim, "Second life and the new generation of virtual worlds," *Computer*, vol. 41, no. 9, pp. 46–53, 2008.

[5] M. Welzl, D. Papadimitriou, M. Scharf, and B. Briscoe, "Open research issues in Internet congestion control," IRTF Internet Draft, Apr. 2009.

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

[7] P. Sarolahti, M. Allman, and S. Floyd, "Determining an appropriate sending rate over an underutilized network path," *Computer Networks*, vol. 51, no. 7, pp. 1815–1832, 2007.

[8] D. Liu, M. Allman, S. Jin, and L. Wang, "Congestion control without a startup phase," in *Proc. PFLDnet*, Feb. 2007.

[9] M. Scharf, "Work in progress: Performance evaluation of fast startup congestion control schemes," in *Proc. Networking 2009, LNCS 5550*, May 2009, pp. 716–727.

[10] M. Hosseini and N. D. Georganas, "MPEG-4 BIFS streaming of large virtual environments and their animation on the Web," in *Proc. ACM Web3D*, 2002, pp. 19–25.

[11] H. Hoppe, "Progressive meshes," in *Proc. ACM SIGGRAPH*, 1996, pp. 99–108.

[12] S. Stegmaier, J. Diepstraten, M. Weiler, and T. Ertl, "Widening the remote visualization bottleneck," in *Proc. ISPA*, 2003, pp. 1–6.

[13] A. Nurminen, "Mobile, hardware-accelerated urban 3D maps in 3G networks," in *Proc. ACM Web3D*, 2007, pp. 7–16.

[14] F. Lamberti and A. Sanna, "A streaming-based solution for remote visualization of 3D graphics on mobile devices," *IEEE Trans. on Visualization and Computer Graphics*, vol. 13, no. 2, pp. 247–260, 2007.

[15] L. Skorin-Kapov, M. Mosmondor, O. Dobrijevic, and M. Matijasevic, "Application-level QoS negotiation and signaling for advanced multimedia services in the IMS," *IEEE Communications Magazine*, vol. 45, no. 87, pp. 108–116, 2007.

[16] H. Li, M. Li, and B. Prabhakaran, "Middleware for streaming 3D progressive meshes over lossy networks," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 2, no. 4, pp. 282–317, 2006.

[17] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann, "The new Web: Characterizing AJAX traffic," in *Proc. Passive and Active Network Measurement, Springer LNCS 4979*, 2008, pp. 31–40.

[18] I. M. Boier-Martin, "Adaptive graphics," *IEEE Comp. Graphics and Appl.*, vol. 23, no. 1, pp. 6–10, 2003.

[19] L. Eggert and W. M. Eddy, "Towards more expressive transport-layer interfaces," in *Proc. ACM/IEEE MobiArch*, 2006, pp. 71–74.

[20] K.-J. Grinnemo, J. Garcia, and A. Brunstrom, "Taxonomy and survey of retransmission-based partially reliable transport protocols," *Computer Communications*, vol. 27, no. 15, pp. 1441–1452, 2004.