Internet Research Task Force (IRTF)
Request for Comments: 6077
Category: Informational
ISSN: 2070-1721

D. Papadimitriou, Ed.
Alcatel-Lucent
M. Welzl
University of Oslo
M. Scharf
University of Stuttgart
B. Briscoe
BT & UCL

February 2011

Open Research Issues in Internet Congestion Control

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document, the result of the Internet Congestion Control Research Group (ICCRG), 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 a state or condition that occurs when network resources are overloaded, resulting in impairments for network users as objectively measured by the probability of loss and/or delay. The overload results in the reduction of utility 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 primal-dual modeling [Kelly98]. Primal congestion control refers to the algorithm executed by the traffic sources 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 or congestion rate 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 into 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) [RFC5783]. The Van Jacobson end-to-end congestion control algorithms [Jacobson88] [RFC2581] [RFC5681] 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 the data link and physical layer on the one hand, and of applications on the other, are pulling TCP congestion control beyond its natural operating regime. This is because it performs poorly as the bandwidth or delay increases. A side effect of these deficiencies is that an increasing share of hosts use non-standardized congestion control enhancements (for instance, many Linux distributions have been shipped with "CUBIC" [Ha08] as the default TCP congestion control mechanism).

While the original Van 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] in order to avoid congestion collapse. Active Queue Management (AQM) in routers, e.g., RED and some of its variants such as Adaptive RED (ARED), 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. It is to be noted that other variants of RED built on AQM, such as Weighted RED (WRED) and RED with In/Out (RIO) [Clark98] are for quality enforcement, whereas Stabilized RED (SRED), and CHOKe [Pan00] and its extensions such as XCHOKe [Chhabra02], are flow policers. In [Bonald00], authors analytically evaluated RED performance.

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 for acceleration of this information flow. Primal-dual TCP/AQM distributed algorithm stability and equilibrium properties have been extensively studied (cf. [Low02], [Low03.1], [Low03.2], [Kelly98], and [Kelly05]).

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 or over a second). Even higher latencies can occur in space 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 the 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 in 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 to new links due to mobility (horizontal/vertical handovers), topology modifications (e.g., in ad hoc or meshed networks), link layer error correction, and dynamic bandwidth provisioning schemes. From this, it follows that path characteristics can be subject to substantial changes within short time frames.

Congestion control algorithms have to deal with this variety in an efficient and stable 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 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, among other things, in low resource utilization.

This has resulted in 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 the congestion window is large, several high-speed congestion control extensions have been developed. However, these new algorithms may be less robust or starve legacy flows in certain situations for which they have not been designed. At the time of writing, there is no common agreement in the IETF on which algorithm(s) and protocol(s) to choose.

It is always possible to tune congestion control parameters based on some knowledge of the environment and the application scenario. However, the interaction between multiple congestion control techniques is not yet well understood. The fundamental challenge is whether it is possible to define one congestion control mechanism that operates reasonably well in a whole range of scenarios that exist in the Internet. Hence, important research questions are how new Internet congestion control mechanisms would have to be designed, which maximum degree of dynamics they can efficiently handle, and whether they can keep the generality of the existing end-to-end solutions.

Some improvements to congestion control could be realized by simple changes to single functions in end-systems or optimizations of network components. However, new mechanism(s) might 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, a key issue 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 is a mathematical tool for describing dynamic systems. It 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. However, control theory has had to be extended to model the interactions between multiple control loops in a network [Vinnic02]. 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 by global stability is typically asymptotic stability: a mechanism should converge to a certain state irrespective of the initial state of the network. Local stability means that if the system is perturbed from its stable state it will quickly return toward the locally stable state.

Some fundamental facts 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 to the controller only states 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 ideally be equal to the feedback frequency. Reacting faster leads to oscillations and instability, while reacting more slowly makes the system tardy [Jain90].

Control theoretic modeling of a realistic network can be quite difficult, especially when taking distinct packet sizes and heterogeneous round-trip times (RTTs) into account. It has therefore become common practice to model simpler cases and to 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. However, a mechanism that is found to be stable in simulations can still not be safely deployed in real networks, since simulation scenarios make simplifying assumptions.

TCP stability can be attributed to two key aspects that 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. On the other hand, research has concluded that stability can be assured with constraints on dynamics that are less stringent than the "conservation of packets principle". From control theory, only rate increase (not the target rate) needs to be inversely proportional to RTT (whereas window-based control converges on a target rate inversely proportional to RTT). A congestion control mechanism can therefore converge on a rate that is independent of RTT as long as its dynamics depend on RTT (e.g., FAST TCP [Jin04]).

In the stability analysis of TCP and of these more modern controls, the impact of slow-start on stability (which can be significant as short-lived HTTP flows often never leave this phase) is not entirely clear.

2.3. Fairness

Recently, the way the Internet community reasons about fairness has been called deeply into question [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] [RFC5681] and TCP-Friendly Rate Control (TFRC) [RFC5348]. [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 [MacK95], 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 an internetwork. They lead to research agendas that are different in their respective objectives, resulting in a different set of open issues.

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

- Should flow fairness depend on the packet rate or the bit rate?

- Should the target flow rate depend on RTT (as in TCP) or should only flow dynamics depend on RTT (e.g., as in FAST TCP [Jin04])?

- How should we 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?

- 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 should we judge at what point it becomes unfair for some flows to continue at a smooth rate while others reduce their rate?

- Which mechanism(s) could be used to enforce approximate flow rate fairness?

- Should we introduce some degree of fairness that takes into account different users' flow activity over time?

- How should we 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) are needed to give accountability for causing congestion?

- How might we design one or two weighted transport protocols (such as TCP, UDP, etc.) with the addition of application policy control over the weight?

- Which policy enforcement might be used by networks, and what are the interactions between application policy and network policy enforcement?

- How should we 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 prerequisite to agreeing on the research agenda. If the relevant metric is flow rate, it places constraints at protocol design time, whereas if the metric is congestion-volume, the constraints move to run-time while design-time constraints can be relaxed [Bri08]. 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, and whether we can rely on trust to maintain approximate fairness without requiring policing complexity [RFC5290]. The latter points may themselves lead to additional research. However, it is also accepted that more research will not necessarily lead to convincing either side to change their opinions. More debate would be needed. It seems also that if the architecture is built to support cost fairness, then equal instantaneous cost rates for flows sharing a bottleneck result in flow-rate fairness; that is, flow-rate fairness can be seen as a special case of cost fairness. One can be used to build the other, but not vice-versa.

3. Detailed Challenges

3.1. Challenge 1: Network Support

This challenge is perhaps the most critical to get right. Changes to the balance of functions between the endpoints and network equipment could require a change to the per-datagram data plane interface between the transport and network layers. Network equipment vendors need to be assured that any new interface is stable enough (on decade timescales) to build into firmware and hardware, and operating-system vendors will not use a new interface unless it is likely to be widely deployed.

Network components 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 end-to-end congestion control. Second, network components can participate in congestion control via explicit signaling mechanisms. Explicit signaling mechanisms, whether in-band or out-of-band, require a communication between network components and end-systems. Signals realized within or over the IP layer are only meaningful to network components that process IP packets. This always includes routers and potentially also middleboxes, but not pure link layer devices. The following section distinguishes clearly between the term "network component" and the term "router"; the term "router" is used whenever the processing of IP packets is explicitly required. One fundamental challenge of network-supported congestion control is that typically not all network components along a path are routers (cf. Section 3.1.3).

The first (optimizing) category of implicit mechanisms can be implemented in any network component that processes and stores packets. 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 in identifying correct values of the parameters that affect the performance of the queuing scheme (due to variation in the number of sources, the capacity, and the feedback delay) [Firoiu00] [Hollot01] [Zhang03]. Many AQM schemes (RED, REM, BLUE, and PI-Controller, but also Adaptive Virtual Queue (AVQ)) do not define a systematic rule for setting their parameters.

The second class of approaches uses explicit signaling. 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 control congestion.

Explicit feedback techniques fall into three broad categories:

- Explicit congestion feedback: one-bit Explicit Congestion Notification (ECN) [RFC3168] or proposals for more than one bit [Xia05];

- Explicit per-datagram rate feedback: the eXplicit Control Protocol (XCP) [Katabi02] [Falk07], or the Rate Control Protocol (RCP) [Dukki05];

- Explicit rate feedback: by means of in-band signaling, such as by Quick-Start [RFC4782], or by means of out-of-band signaling, e.g., Congestion Avoidance with Distributed Proportional Control/Performance Transparency Protocol (CADPC/PTP) [Welzl03].

Explicit router feedback can address some of the inherent shortcomings of TCP. For instance, XCP was developed to overcome the inefficiency 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 equal 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 generally equalizes rates among 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 per-packet feedback similar to XCP. But unlike XCP, RCP focuses on the reduction of flow completion times [Dukki06], taking an optimistic approach to flows likely to arrive in the next RTT and tolerating larger instantaneous queue sizes [Dukki05]. XCP, on the other hand, gives very poor flow completion times for short flows.

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 it remains as simple as possible (any additional complexity is placed above the IP layer, i.e., at the edges) so as to ensure evolvability, reliability, and robustness. Furthermore, the fate-sharing principle ([Clark88], "Design Philosophy of the DARPA Internet Protocols") mandates that an end-to-end Internet protocol design should not rely on the maintenance of any per-flow state (i.e., information about the state of the end-to-end communication) inside the network and that the network state (e.g., routing state) maintained by the Internet shall minimize its interaction with the states maintained at the endpoints/hosts [RFC1958].

However, as discussed in [Moors02] for instance, 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 network 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 per-flow behavior inside the network should be avoided, as such a design would clearly be at odds with the end-to-end and fate-sharing design principles.

The end-to-end and fate-sharing principles are generally regarded as the key ingredients for ensuring a scalable and survivable network design. In order to ensure that new congestion control mechanisms are scalable, violating these principles must therefore be avoided.

For instance, protocols like XCP and RCP seem not to require flow state in the network, but this is only the case if the network trusts i) the receiver not to lie when feeding back the network's delta to the requested rate; ii) the source not to lie when declaring its rate; and iii) the source not to cheat when setting its rate in response to the feedback [Katabi04].

Solving these problems for non-cooperative environments like the public Internet requires flow state, at least on a sampled basis. However, because flows can create new identifiers whenever they want, sampling does not provide a deterrent -- a flow can simply cheat until it is discovered and then switch to a whitewashed identifier [Feldman04], and continue cheating until it is discovered again ([Bri09], S7.3).

However, holding flow state in the network only seems to solve these policing problems in single autonomous system settings. A multi-domain system would seem to require a completely different protocol structure, as the information required for policing is only seen as packets leave the internetwork, but the networks where packets enter will also want to police compliance.

Even if a new protocol structure were found, it seems unlikely that network flow state could be avoided given the network's per-packet flow rate instructions would need to be compared against variations in the actual flow rate, which is inherently not a per-packet metric. These issues have been outstanding ever since integrated services (IntServ) was identified as unscalable in 1997 [RFC2208]. All subsequent attempts to involve network elements in limiting flow rates (XCP, RCP, etc.) will run up against the same open issue if anyone attempts to standardize them for use on the public Internet.

In general, network support of congestion control raises many issues that have not been completely solved yet.

3.1.1. Performance and Robustness

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

Router support can help to improve performance, 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 feedback information may be imprecise. For instance, severe congestion can delay feedback signals. Also, in-network 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 network elements theoretically improve performance in the complete range of communication scenarios that exist in the Internet without damaging or impacting end-to-end mechanisms already in place?

- Is it possible to design robust congestion control mechanisms that offer significant benefits with minimum additional risks, even if Internet traffic patterns will change in the future?

- What is the minimum support that is needed from the network in order to achieve significantly better performance than with end-to-end mechanisms and the current IP header limitations that provide at most unary ECN signals?

3.1.2. Granularity of Network Component Functions

There are several degrees of freedom concerning the involvement of network entities, ranging from some few additional functions in network management procedures on the one end to 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, or hard state). The additional router processing is a challenge for Internet scalability and could also increase end-to-end latencies.

Although there are many research proposals that do not require per-flow state and thus do not cause a large processing overhead, there are no known full solutions (i.e., including anti-cheating) that do not require per-flow processing. Also, scalability issues could 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 to a minimum?

3.1.3. Information Acquisition

In order to support congestion control, network components 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 operating at the IP layer do not necessarily know the link layer network topology and link capacities, and these are not always constant (e.g., on shared wireless links or bandwidth-on-demand links). Depending on the network technology, there can be queues or bottlenecks that are not directly visible at the IP networking layer.

   Difficulties also arise when using IP-in-IP tunnels [RFC2003], IPsec tunnels [RFC4301], IP encapsulated in the Layer Two Tunneling Protocol (L2TP) [RFC2661], Generic Routing Encapsulation (GRE) [RFC1701] [RFC2784], the Point-to-Point Tunneling Protocol (PPTP) [RFC2637], or Multiprotocol Label Switching (MPLS) [RFC3031] [RFC3032]. In these cases, link information could be determined by cross-layer information exchange, but this requires interfaces capable of processing link layer technology specific information. An alternative could be online measurements, but this can cause significant additional network overhead. It is an open research question as to how much, if any, online traffic measurement would be acceptable (at run-time). Encapsulation and decapsulation of explicit congestion information have been specified for IP-in-IP tunnelling [RFC6040] and for MPLS-in-MPLS or MPLS-in-IP [RFC5129].

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, network components 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?

- Which information is so important to higher-layer controllers that machine architecture research should focus on designing to provide it?

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 [Sarola07]. The latter case requires additional protocols and a secure binding between the signals and the packets they refer to. Out-of-band signaling 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.)? Should the feedback signaling be path-coupled or path-decoupled?

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

- What direction should feedback take (from network resource via receiver to sender, or directly back to sender)?

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 loss, rendering the typical reaction of a congestion control mechanism inappropriate. As a result, non-congestive loss may be more prevalent in these networks due to corruption loss (when the wireless link cannot be conditioned to properly control its error rate or due to transient wireless link interruption in areas of poor coverage).

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 (see, for instance, the discussion that occurred in April 2003 on the Datagram Congestion Control Protocol (DCCP) working group list http://www.ietf.org/mail-archive/web/dccp/current/mail6.html) 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 because it has lost the integrity of the feedback loop.

Generally, two questions must be addressed when designing a congestion control mechanism that takes corruption loss 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 General Packet Radio Service (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 detecting and accordingly reacting (or not) 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 that 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, e.g., [Tickoo05]. For instance, TCP Westwood [Mascolo01] sets the congestion window equal to the measured bandwidth at the 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.

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?

- Can an ECN-capable flow infer that loss must be due to corruption just from lack of explicit congestion notifications around a loss episode [Tickoo05]? Or could this inference be dangerous, given the transport does not know whether all queues on the path are ECN-capable or not?

3.3. Challenge 3: Packet Size

TCP does not take packet size into account when responding to losses or ECN. Over past years, the performance of TCP congestion avoidance algorithms has been extensively studied. The well-known "square root formula" provides an estimation of 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 B is proportional to the segment size and inversely proportional to the RTT and the square root of the drop probability:

$$B \sim C \frac{S}{RTT} \frac{1}{\sqrt{p}}$$

where

C	is a constant
S	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 tradeoff between high throughput (the larger a packet, the smaller the relative header overhead) and low packet latency (the smaller a packet, the shorter the time that is needed until it is filled with data). Observing that TCP is not optimal for applications with streaming media (since reliable in-order delivery and congestion control can cause arbitrarily long delays), this tradeoff has not usually been considered for TCP applications. Therefore, the influence of the packet size on the sending rate has not typically been seen as a significant issue, given there are still few paths through the Internet that support packets larger than the 1500 bytes common with Ethernet.

The situation is already 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 approximation of TCP's congestion control as well as incorporating a variant of Selective Acknowledgment (SACK) [RFC2018] [RFC3517]. CCID 2 is suitable for senders that can adapt to the abrupt changes in the congestion window typical of TCP's AIMD congestion control, and particularly useful for senders that 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) [RFC5348] is a congestion control mechanism designed for unicast flows operating in a best-effort Internet environment. When competing for bandwidth, its window is similar to TCP flows but has a much lower variation of throughput over time than 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 (CCID 4) [RFC5622]: TFRC Small Packets (TFRC-SP) [RFC4828], a variant of the TFRC mechanism, has been designed for applications that exchange small packets. The objective of TFRC-SP is to achieve the same bandwidth in 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. 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 the packet header size, as specified in [RFC4828].

The resulting open questions are:

- How does TFRC-SP operate under various network conditions?

- How can congestion control be designed so as to scale with packet size (dependency of congestion algorithm on packet size)?

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., frame, packet, and transport 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-case scenario of all minimum-size packets. In this case, x = r*(h + e) for some small value of e. Therefore, packet congestion is not impossible for runs of small packets (e.g., TCP

ACKs or denial-of-service (DoS) attacks with TCP SYNs or small UDP datagrams). But absent such anomalous workloads, equipment vendors at the 2008 ICCRG meeting believed that equipment could still be designed so that any congestion should be due to bit overload and not packet overload.

This observation raises additional open issues:

- Can bit congestion remain prevalent?

  Being able to assume that congestion is generally due to excess bits and not excess packets is a useful simplifying assumption in the design of congestion control protocols. Can we rely on this assumption for the future? An alternative view is that in-network processing will become commonplace, so that per-packet processing will as likely be the bottleneck as per-bit transmission [Shin08].

  Over the last three decades, performance gains have mainly been achieved through increased packet rates and not bigger packets. But if bigger maximum segment sizes do become more prevalent, tiny segments (e.g., ACKs) will not stop being widely used -- leading to 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 loss

  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 loss is longer and includes transmission corruption 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 packet 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 does not necessarily reduce the bit rate.

Given distinguishing between corruption 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 loss, and not just two of them.

Nonetheless, even if we find a way for network equipment to explicitly distinguish which sort of loss 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, and servers; and at every lower-layer device like low-end hubs, DSLAMs, Wireless LAN (WLAN) cards, cellular base-stations, and so on. Thus, transport protocols will always have to cope with packet drops due to unpredictable causes, so we should always treat AQM as an optimization, given it will never be ubiquitous throughout the public Internet.

- 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 [Bri10], along with other aspects of packet size and congestion control.

  [Bri10] 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.

- 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, and not when the network writes, the notification -- it opens up a significant problem 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 1500 byte data segments is very prevalent for TCP, and the incidence of alternative maximum 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 to be sensitive to packet size, 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 account for packet size. It is still an open research issue to establish whether TCP was correct in not taking packet size into account. If it is determined that TCP was wrong in this respect, we should discourage future protocol designs from following TCP's example. For example, as explained above, the small-packet variant of TCP-friendly rate control (TFRC-SP [RFC4828]) is an experimental protocol that aims to take packet size into account. Whatever packet size it uses, it ensures that its rate approximately equals that of a TCP using 1500 byte segments. This raises the further question of whether TCP with 1500 byte segments will be a suitable long-term gold standard, or whether we need a more thorough review of what it means for a congestion control mechanism to scale with packet size.

3.4. Challenge 4: Flow Startup

The beginning of data transmissions imposes some further, unique challenges: when a connection to a new destination is established, the end-systems have hardly any information about the characteristics of the path in between and the available bandwidth. In this flow startup situation, there is no obvious choice as to how to start to send. A similar problem also occurs after relatively long idle times, since the congestion control state then no longer reflects current information about the state of the network (flow restart problem).

Van Jacobson [Jacobson88] suggested using the slow-start mechanism both for the flow startup and the flow restart, and this is today's standard solution [RFC2581] [RFC5681]. Per [RFC5681], the slow-start algorithm is used when the congestion window (cwnd) < slow-start threshold (ssthresh), whose initial value is set arbitrarily high (e.g., to the size of the largest possible advertised window) and reduced in response to congestion. During slow-start, TCP increments the cwnd by at most Sender Maximum Segment Size (MSS) bytes for each ACK received that cumulatively acknowledges new data. Slow-start ends when cwnd exceeds ssthresh or when congestion is observed. However, the slow-start is not optimal in many situations. First, it can take quite a long time until a sender can fully utilize the available bandwidth on a path. Second, the exponential increase may be too aggressive and cause multiple packet loss if large congestion windows are reached (slow-start overshooting). Finally, the slow-start does not ensure that new flows converge quickly to a reasonable share of resources, particularly when the new flows compete with long-lived flows and come out of slow-start early (slow-start vs overshoot tradeoff). This convergence problem may even worsen if more aggressive congestion control variants are widely used.

The slow-start and its interaction with the congestion avoidance phase was largely designed by intuition [Jacobson88]. So far, little theory has been developed to understand the flow startup problem and its implication on congestion control stability and fairness. There is also no established methodology to evaluate whether new flow startup mechanisms are appropriate or not.

As a consequence, it is a non-trivial task to address the shortcomings of the slow-start algorithm. Several experimental enhancements have been proposed, such as congestion window validation [RFC2861] and limited slow-start [RFC3742]. There are also ongoing research activities, focusing, e.g., on bandwidth estimation techniques, delay-based congestion control, or rate-pacing mechanisms. However, any alternative end-to-end flow startup approach has to cope with the inherent problem that there is no or only little information about the path at the beginning of a data transfer. This uncertainty could be reduced by more expressive feedback signaling (cf. Section 3.1). For instance, a source could learn the path characteristics faster with the Quick-Start mechanism [RFC4782]. But even if the source knew exactly what rate it should aim for, it would still not necessarily be safe to jump straight to that rate. The end-system still does not know how a change in its own rate will affect the path, which also might become congested in less than one RTT. Further research would be useful to understand the effect of decreasing the uncertainty by explicit feedback separately from control theoretic stability questions. Furthermore, flow startup also raises fairness questions. For instance, it is unclear whether it could be reasonable to use a faster startup when an end-system detects that a path is currently not congested.

In summary, there are several topics for further research concerning flow startup:

- Better theoretical understanding of the design and evaluation of flow startup mechanisms, concerning their impact on congestion risk, stability, and fairness.

- Evaluating whether it may be appropriate to allow alternative starting schemes, e.g., to allow higher initial rates under certain constraints [Chu10]; this also requires refining the definition of fairness for startup situations.

- Better theoretical models for the effects of decreasing uncertainty by additional network feedback, particularly if the path characteristics are very dynamic.

3.5. Challenge 5: Multi-Domain Congestion Control

Transport protocols such as TCP operate over the Internet, which is divided into autonomous systems. These systems are characterized by their heterogeneity as IP networks are realized by a multitude of technologies.

3.5.1. Multi-Domain Transport of Explicit Congestion Notification

Different conditions and their variations lead 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 encourages them to efficiently use the network. In IP, ECN uses the two least significant bits of the (former) IPv4 Type of Service (TOS) octet or the (former) IPv6 Traffic Class octet [RFC2474] [RFC3260]. 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. The congestion-based feedback scheme, however, has limitations when applied on an inter-domain basis. Indeed, Sections 8 and 19 of [RFC3168] detail the implications of two possible attacks:

1. non-compliance: a network erasing a Congestion Experienced (CE) codepoint introduced earlier on the path, and

2. subversion: a network changing Not ECN-Capable Transport (Not-ECT) to ECT.

Both of these problems could allow an attacking network to cause excess congestion in an upstream network, even if the transports were behaving correctly. There are to date two possible solutions to the non-compliance problem (number 1 above): the ECN-nonce [RFC3540] and the [CONEX] work item inspired by the re-ECN incentive system

[Bri09]. Nevertheless, accidental rather than malicious erasure of ECN is an issue for IPv6 where the absence of an IPv6 header checksum implies that corruption of ECN could be more impacting than in the IPv4 case.

Fragmentation is another issue: the ECN-nonce cannot protect against misbehaving receivers that conceal marked fragments; thus, some protection is lost in situations where path MTU discovery is disabled. Note also that ECN-nonce wouldn't protect against the subversion issue (number 2 above) because, by definition, a Not-ECT packet comes from a source without ECN enabled, and therefore without the ECN-nonce enabled. So, there is still room for improvement on the ECN mechanism when operating in multi-domain networks.

Operational/deployment experience is nevertheless required to determine the extent of these problems. The second problem is mainly related to deployment and usage practices and does not seem to result in any specific research challenge.

Another controversial solution in a multi-domain environment may be the TCP rate controller (TRC), a traffic conditioner that regulates the TCP flow at the ingress node in each domain by controlling packet drops and delays 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 interferes with the end-to-end TCP model, and thus it would interfere with past and future diversity of TCP implementations (violating the end-to-end principle). In particular, the TRC embeds the flow rate equality view of fairness in the network, and would prevent evolution to forms of fairness based on congestion-volume (Section 2.3).

3.5.2. Multi-Domain Exchange of Topology or Explicit Rate Information

Security is a challenge for multi-domain exchange of explicit rate signals, whether in-band or out-of-band. At domain boundaries, authentication and authorization issues can arise whenever congestion control information is exchanged. From this perspective, the Internet does not so far have any security architecture for this problem.

The future evolution of Internet inter-domain operation has to show whether more multi-domain information exchange can be effectively realized. This is of particular importance for congestion control schemes that make use of explicit per-datagram rate feedback (e.g., RCP or XCP) or explicit rate feedback that uses in-band congestion signaling (e.g., Quick-Start) or out-of-band signaling (e.g., CADPC/PTP). Explicit signaling exchanges at the inter-domain level that result in local domain triggers are currently absent from the

Internet. From this perspective, security issues resulting from limited trust between different administrative units result in policy enforcement that exacerbates the difficulty encountered when explicit feedback congestion control information is exchanged between domains. Note that even though authentication mechanisms could be extended for this purpose (by recognizing that explicit rate schemes such as RCP or XCP have the same inter-domain security requirements and structure as IntServ), they suffer from the same scalability problems as identified in [RFC2208]. Indeed, in-band rate signaling or out-of-band per-flow traffic specification signaling (like in the Resource Reservation Protocol (RSVP)) results in similar scalability issues (see Section 3.1).

Also, many autonomous systems 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. Indeed, revealing the internal network structure is highly sensitive in multi-domain network operations and thus also a concern when it comes to the deployability of congestion control schemes. For instance, a network-assisted congestion control scheme with explicit signaling could reveal more information about the internal network dimensioning than TCP does today.

3.5.3. Multi-Domain Pseudowires

Extending pseudowires across multiple domains poses specific issues. Pseudowires (PWs) [RFC3985] may carry non-TCP data flows (e.g., Time-Division Multiplexing (TDM) traffic or Constant Bit Rate (CBR) ATM traffic) over a multi-domain IP network. Structure-Agnostic TDM over Packet (SAToP) [RFC4553], Circuit Emulation Service over Packet Switched Network (CESoPSN) [RFC5086], and TDM over IP (TDMoIP) [RFC5087] are not responsive to congestion control as discussed in [RFC2914] (see also [RFC5033]). The same observation applies to ATM circuit emulating services (CESs) interconnecting CBR equipment (e.g., Private Branch Exchanges (PBX)) across a Packet Switched Network (PSN).

Moreover, it is not possible to simply reduce the flow rate of a TDM PW or an ATM PW when facing packet loss. Providers can rate-control corresponding incoming traffic, but they may not be able to detect that PWs carry TDM or CBR ATM traffic (mechanisms for characterizing the traffic's temporal properties may not necessarily be supported).

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 Service Level Agreement (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 because it is not able to detect that IP packets are carrying constant-bit-rate service traffic for which the only useful congestion control mechanism would rely on implicit or explicit admission control, meaning self-blocking or enforced blocking, respectively.

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 (e.g., each flow within another PW) would follow TCP's AIMD algorithm of reducing the sending rate by half, in response to each packet drop. Nevertheless, the PWs carrying TDM traffic could take all the available capacity while other more TCP-friendly or generally congestion-responsive traffic reduced itself to nothing. Note here that the situation may simply occur because S4 suddenly turns on additional TDM channels.

It is neither possible nor desirable to assume that edge routers will soon have the ability to detect the responsiveness of the carried traffic, but it is still important for transit providers to be able to police a fair, robust, responsive, and efficient congestion control technique in order to avoid impacting congestion-responsive Internet traffic. However, we must not require only certain specific responses to congestion to be embedded within the network, which would harm evolvability. So designing the corresponding mechanisms in the data and control planes still requires further investigation.

3.6. Challenge 6: Precedence for Elastic Traffic

Traffic initiated by so-called elastic applications adapts to the available bandwidth using feedback about the state of the network.

For elastic applications, the transport dynamically adjusts the data traffic sending rate to different network conditions. Examples encompass short-lived elastic traffic including HTTP and instant-messaging traffic, as well as long file transfers with FTP and applications targeted by [LEDBAT]. 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 combined 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 the interactions between congestion control and DiffServ [RFC2475] remained unaddressed until recently.

Recently, a study and a potential solution have been proposed that introduce Guaranteed TFRC (gTFRC) [Lochin06]. gTFRC is an adaptation of TCP-Friendly Rate Control providing throughput guarantees for unicast flows over the DiffServ/Assured Forwarding (AF) class. The purpose of gTFRC is to distinguish the guaranteed part from the best-effort part of the traffic resulting from AF conditioning. The proposed congestion control has been specified and tested inside DCCP/CCID 3 for DiffServ/AF networks [Lochin07] [Jourjon08].

Nevertheless, there is still work to be performed regarding lower-precedence traffic -- data transfers that are useful, yet not important enough to warrant significantly impairing 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 Low Priority (TCP-LP) [Kuzmanovic03]), and there is a specification for achieving it via router mechanisms [RFC3662]. It seems, however, that network-based lower-precedence mechanisms are not yet a common service on the Internet. Since early 2010, end-to-end mechanisms for lower precedence, e.g., [Shal10], have become common -- at least when competing with other traffic as part of its own queues (e.g., in a home router). But it is less clear whether users will be willing to make their background traffic yield to other people's foreground traffic, unless the appropriate incentives are created.

There is an issue over how to reconcile two divergent views of the relation between traffic class precedence and congestion control. One view considers that congestion signals (losses or explicit notifications) in one traffic class are independent of those in another. The other relates marking of the classes together within the active queue management (AQM) mechanism [Gibbens02]. In the independent case, using a higher-precedence class of traffic gives a higher scheduling precedence and generally lower congestion level. In the linked case, using a higher-precedence class of traffic still gives higher scheduling precedence, but results in a higher level of congestion. This higher congestion level reflects the extra congestion higher-precedence traffic causes to both classes combined. The linked case separates scheduling precedence from rate control. The end-to-end congestion control algorithm can separately choose to take a higher rate by responding less to the higher level of congestion. This second approach could become prevalent if weighted congestion controls were common. However, it is an open issue how the two approaches might co-exist or how one might evolve into the other.

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 improve 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., [Savage99], [RFC3540]). But, although such strategies are worryingly powerful, they do not yet seem common (however, evidence of attack prevalence is itself a research requirement).

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 as they experience more losses. Some believe the Internet was designed to allow such freedom, so it can hardly be called misbehavior. But others consider it 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]. Also, Internet video on demand services that transfer much greater data rates without congestion control are becoming popular. In general, it is recommended that such UDP applications use some form of congestion control [RFC5405].

Note that the problem is not just misbehavior driven by a self-interested desire for more bandwidth. Indeed, congestion control may be attacked by someone who makes no gain for themselves, other than the satisfaction of harming others (see Security Considerations in 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. How would such mechanisms be designed?

- Which congestion control primitives could safely 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 could disappear from the Internet. So, it may no longer be sufficient to rely on developers/users voluntarily submitting themselves to congestion control. As a consequence, mechanisms to enforce fairness (see Sections 2.3, 3.4, and 3.5) 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 so significant, or they are of a different nature compared to the main challenges depicted so far.

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, if echoed back immediately by the 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 into 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 transport protocols such as TCP, the Stream Control Transmission Protocol (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) or the rate-control equation. 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 acknowledgments. For TCP, this is the default RTT measurement procedure; it is used in combination with Karn's algorithm, which prohibits RTT measurements from retransmitted segments [RFC2988]. Traditionally, TCP implementations take one RTT measurement at a time (i.e., about once per RTT). As an 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, e.g., to find the one-way variation in delay due to one-way congestion. 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. The more fundamental question is to determine whether it is necessary or not for network elements to measure or know the RTT.

As outlined earlier in this document, the round-trip time is typically not a constant value. For a given path, there is a 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-media access control schemes, recovery procedures, or other sub-IP layer mechanisms. Furthermore, a change of the path (e.g., route flapping, hand-over 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 more slowly to sudden changes in 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., [Sarola02]). There are also proposals for more sophisticated estimators, such as Kalman filters or estimators that utilize mainly peak values.

However, open questions related to RTT estimation in the Internet remain:

- Optimal measurement frequency: Currently, there is no theory or common understanding of the right time scale of RTT measurement. In particular, the necessity for 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 toward higher-precision timers, limited granularity (particularly on low-cost devices) 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 or even crash completely 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 and is still hindered by malfunctioning home-hubs, but there are many other examples (e.g., the Window Scaling option of TCP) [Thaler07].

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. Recommendations 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 to the maintainers of such devices.

A possible way to reduce such problems in the future would be guidelines for standards authors to ensure that "forward compatibility" is considered in all IETF work. That is, the default behavior of a device should be precisely defined for all possible values and combinations of protocol fields, and not just the minimum necessary for the protocol being defined. Then, when previously unused or reserved fields start to be used by newer devices to comply with a new standard, older devices encountering unusual fields should at least behave predictably.

3.8.3. Dependence on RTT

AIMD window algorithms that have the goal of packet conservation end up converging on a rate that is inversely proportional to RTT. However, control theoretic approaches to stability have shown that only the increase in rate (acceleration), and not the target rate, needs to be inversely proportional to RTT [Jin04].

It is possible to have more aggressive behaviors for some demanding applications as long as they are part of a mix with less aggressive transports [Key04]. This beneficial effect of transport type mixing is probably how the Internet currently manages to remain stable even in the presence of TCP slow-start, which is more aggressive than the theory allows for stability. Research giving deeper insight into these aspects would be very useful.

3.8.4. Congestion Control in Multi-Layered Networks

A network of IP nodes is just as vulnerable to congestion in the lower layers between IP-capable nodes as it is to congestion on the IP-capable nodes themselves. If network elements take a greater part in congestion control (ECN, XCP, RCP, etc. -- see Section 3.1), these techniques will either need to be deployed at lower layers as well, or they will need to interwork with lower-layer mechanisms.

[RFC5129] shows how to propagate ECN from lower layers upwards for the specific case of MPLS, but to the authors' knowledge the layering problem has not been addressed for explicit rate protocol proposals such as XCP and RCP. Some issues are straightforward matters of interoperability (e.g., how exactly to copy fields up the layers) while others are less obvious (e.g., re-framing issues: if RCP were deployed in a lower layer, how might multiple small RCP frames, all with different rates in their headers, be assembled into a larger IP layer datagram?).

Multi-layer considerations also confound many mechanisms that aim to discover whether every node on the path supports a new congestion control protocol. For instance, some proposals maintain a secondary Time to Live (TTL) field parallel to that in the IP header. Any nodes that support the new behavior update both TTL fields, whereas legacy IP nodes will only update the IP TTL field. This allows the endpoints to check whether all IP nodes on the path support the new behavior, in which case both TTLs will be equal at the receiver. But mechanisms like these overlook nodes at lower layers that might not support the new behavior.

A further related issue is congestion control across overlay networks of relays [Hilt08] [Noel07] [Shen08].

Section 3.5.3 deals with inelastic multi-domain pseudowires (PWs), where the identity of the pseudowire itself implies the characteristics of the traffic crossing the multi-domain PSN (independently of the actual characteristics of the traffic carried in the PW). A more complex situation arises when inelastic traffic is carried as part of a pseudowire (e.g., inelastic traffic over Ethernet PW over PSN) whose edges do not have the means to characterize the properties of the traffic encapsulated in the Ethernet frames. In this case, the problem explained in Section 3.5.3 is not limited to multi-domain pseudowires but more generally arises from a "pseudowire carrying inelastic traffic" (whether over a single- or multi-domain PSN).

The problem becomes even more intricate when the Ethernet PW carries both inelastic and elastic traffic. Addressing this issue further supports our observation that a general framework to efficiently deal with congestion control problems in multi-layer networks without harming evolvability is absolutely necessary.

3.8.5. Multipath End-to-End Congestion Control and Traffic Engineering

Recent work has shown that multipath endpoint congestion control [Kelly05] offers considerable benefits in terms of resilience and resource usage efficiency. The IETF has since initiated a work item on multipath TCP [MPTCP]. By pooling the resources on all paths, even nodes not using multiple paths benefit from those that are.

There is considerable further research to do in this area, particularly to understand interactions with network-operator-controlled route provisioning and traffic engineering, and indeed whether multipath congestion control can perform better traffic engineering than the network itself, given the right incentives [Arkko09].

3.8.6. ALGs and Middleboxes

An increasing number of application layer gateways (ALGs), middleboxes, and proxies (see Section 3.6 of [RFC2775]) are deployed at domain boundaries to verify conformance but also filter traffic and control flows. One motivation is to prevent information beyond routing data leaking between autonomous systems. These systems split up end-to-end TCP connections and disrupt end-to-end congestion control. Furthermore, transport over encrypted tunnels may not allow other network entities to participate in congestion control.

Basically, such systems disrupt the primal and dual congestion control components. In particular, end-to-end congestion control may be replaced by flow-control backpressure mechanisms on the split connections. A large variety of ALGs and middleboxes use such mechanisms to improve the performance of applications (Performance Enhancing Proxies, Application Accelerators, etc.). However, the implications of such mechanisms, which are often proprietary and not documented, have not been studied systematically so far.

There are two levels of interference:

- The "transparent" case, i.e., the endpoint address from the sender perspective is still visible to the receiver (the destination IP address). Relay systems that intercept payloads but do not relay congestion control information provide an example. Such middleboxes can prevent the operation of end-to-end congestion control.

- The "non-transparent" case, which causes fewer problems for congestion control. Although these devices interfere with end-to-end network transparency, they correctly terminate network, transport, and application layer protocols on both sides, which individually can be congestion controlled.

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 require endpoints 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 that attack machines and the 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 against the network, which can be considered (from the perspective of a well-behaved Internet user) as a congestion control enforcement problem. Even some denial-of-service attacks on hosts (rather than the network) could be considered as a congestion control enforcement issue at the higher layer. But clearly there are also denial-of-service attacks that would not be solved by enforcing congestion control.

Sections 3.5 and 3.7 on multi-domain issues and misbehaving senders and receivers also discuss some information security issues suffered by various congestion control approaches.

5. References

5.1. Informative References

[Allman99] Allman, M. and V. Paxson, "On Estimating End-to-End Network Path Properties", Proceedings of ACM SIGCOMM'99, September 1999.

[Andrew05] Andrew, L., Wydrowski, B., and S. Low, "An Example of Instability in XCP", Manuscript available at <http://netlab.caltech.edu/maxnet/XCP_instability.pdf>.

[Arkko09] Arkko, J., Briscoe, B., Eggert, L., Feldmann, A., and M. Handley, "Dagstuhl Perspectives Workshop on End-to-End Protocols for the Future Internet," ACM SIGCOMM Computer Communication Review, Vol. 39, No. 2, pp. 42-47, April 2009.

[Ath01] Athuraliya, S., Low, S., Li, V., 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., and A.L. Ananda, "TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links", Proceedings of IEEE INFOCOM'01, Anchorage (Alaska), USA, April 2001.

[Bonald00] Bonald, T., May, M., and J.-C. Bolot, "Analytic Evaluation of RED Performance", Proceedings of IEEE INFOCOM'00, Tel Aviv, Israel, March 2000.

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

[Bri07] Briscoe, B., "Flow Rate Fairness: Dismantling a Religion", ACM SIGCOMM Computer Communication Review, Vol. 37, No. 2, pp. 63-74, April 2007.

[Bri08] Briscoe, B., Moncaster, T. and L. Burness, "Problem Statement: Transport Protocols Don't Have To Do Fairness", Work in Progress, July 2008.

[Bri09] Briscoe, B., "Re-feedback: Freedom with Accountability for Causing Congestion in a Connectionless Internetwork", UCL PhD Thesis (2009).

[Bri10] Briscoe, B. and J. Manner, "Byte and Packet Congestion Notification," Work in Progress, October 2010.

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

[Chhabra02] Chhabra, P., Chuig, S., Goel, A., John, A., Kumar, A., Saran, H., and R. Shorey, "XCHOKe: Malicious Source Control for Congestion Avoidance at Internet Gateways," Proceedings of IEEE International Conference on Network Protocols (ICNP'02), Paris, France, November 2002.

[Chiu89] Chiu, D.M. 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.

[Clark88] Clark, D., "The design philosophy of the DARPA internet protocols", ACM SIGCOMM Computer Communication Review, Vol. 18, No. 4, pp. 106-114, August 1988.

[Clark98] Clark, D. 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.

[Chu10] Chu, J., Dukkipati, N., Cheng, Y., and M. Mathis, "Increasing TCP's Initial Window", Work in Progress, October 2010.

[CONEX] IETF WG Action: Congestion Exposure (conex).

[Dukki05] Dukkipati, N., Kobayashi, M., Zhang-Shen, R., and N. McKeown, "Processor Sharing Flows in the Internet", Proceedings of International Workshop on Quality of Service (IWQoS'05), Passau, Germany, June 2005.

[Dukki06] Dukkipati, N. and N. McKeown, "Why Flow-Completion Time is the Right Metric for Congestion Control", ACM SIGCOMM Computer Communication Review, Vol. 36, No. 1, January 2006.

[ECODE] "ECODE Project", European Commission Seventh Framework Program, Grant No. 223936, <http://www.ecode-project.eu>.

[Falk07] Falk, A., Pryadkin, Y., and D. Katabi, "Specification for the Explicit Control Protocol (XCP)", Work in Progress, January 2007.

[Feldman04] Feldman, M., Papadimitriou, C., Chuang, J., and I. Stoica, "Free-Riding and Whitewashing in Peer-to-Peer Systems" Proceedings of ACM SIGCOMM Workshop on Practice and Theory of Incentives in Networked Systems (PINS'04) 2004.

[Firoiu00] Firoiu, V. and M. Borden, "A Study of Active Queue Management for Congestion Control", Proceedings of IEEE INFOCOM'00, Tel Aviv, Israel, March 2000.

[Floyd93] Floyd, S. and V. Jacobson, "Random early detection gateways for congestion avoidance", IEEE/ACM Transactions on Networking, Vol. 1, No. 4, pp. 397-413, August 1993.

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

[Gibbens02] Gibbens, R. and Kelly, F., "On Packet Marking at Priority Queues", IEEE Transactions on Automatic Control, Vol. 47, No. 6, pp. 1016-1020, 2002.

[Ha08] Ha, S., Rhee, I., and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant", ACM SIGOPS Operating System Review, Vol. 42, No. 5, pp. 64-74, 2008.

[Hilt08] Hilt, V. and I. Widjaja, "Controlling Overload in Networks of SIP Servers", Proceedings of IEEE International Conference on Network Protocols (ICNP'08), Orlando (Florida), USA, October 2008.

[Hollot01] Hollot, C., Misra, V., Towsley, D., and W.-B. Gong, "A Control Theoretic Analysis of RED", Proceedings of IEEE INFOCOM'01, Anchorage (Alaska), USA, April 2001.

[Jacobson88] Jacobson, V., "Congestion Avoidance and Control", Proceedings of ACM SIGCOMM'88 Symposium, August 1988.

[Jain88] Jain, R. and K. Ramakrishnan, "Congestion Avoidance in Computer Networks with a Connectionless Network Layer: Concepts, Goals, and Methodology", Proceedings of IEEE Computer Networking Symposium, Washington DC, USA, April 1988.

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

[Jin04] Jin, Ch., Wei, D.X., and S. Low, "FAST TCP: Motivation, Architecture, Algorithms, Performance", Proceedings of IEEE INFOCOM'04, Hong-Kong, China, March 2004.

[Jourjon08] Jourjon, G., Lochin, E., and P. Senac, "Design, Implementation and Evaluation of a QoS-aware Transport Protocol", Elsevier Computer Communications, Vol. 31, No. 9, pp. 1713-1722, June 2008.

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

[Katabi04] Katabi, D., "XCP Performance in the Presence of Malicious Flows", Proceedings of PFLDnet'04 Workshop, Argonne (Illinois), USA, February 2004.

[Kelly05] Kelly, F. and Th. Voice, "Stability of end-to-end algorithms for joint routing and rate control", ACM SIGCOMM Computer Communication Review, Vol. 35, No. 2, pp. 5-12, April 2005.

[Kelly98]   Kelly, F., Maulloo, A., 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] Keshav, S., "What is congestion and what is congestion control", Presentation at IRTF ICCRG Workshop, PFLDnet 2007, Los Angeles (California), USA, February 2007.

[Key04]   Key, P., Massoulie, L., Bain, A., and F. Kelly, "Fair Internet Traffic Integration: Network Flow Models and Analysis", Annales des Telecommunications, Vol. 59, No. 11-12, pp. 1338-1352, November-December 2004.

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

[Kuzmanovic03] Kuzmanovic, A. and E.W. Knightly, "TCP-LP: A Distributed Algorithm for Low Priority Data Transfer", Proceedings of IEEE INFOCOM'03, San Francisco (California), USA, April 2003.

[LEDBAT]   IETF WG Action: Low Extra Delay Background Transport (ledbat).

[Lochin06] Lochin, E., Dairaine, L., and G. Jourjon, "Guaranteed TCP Friendly Rate Control (gTFRC) for DiffServ/AF Network", Work in Progress, August 2006.

[Lochin07] Lochin, E., Jourjon, G., and L. Dairaine, "Study and enhancement of DCCP over DiffServ Assured Forwarding class", 4th Conference on Universal Multiservice Networks (ECUMN 2007), Toulouse, France, February 2007.

[Low02]   Low, S., Paganini, F., Wang, J., Adlakha, S., and J.C. Doyle, "Dynamics of TCP/RED and a Scalable Control", Proceedings of IEEE INFOCOM'02, New York (New Jersey), 2002.

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

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

[Low05] Low, S., Andrew, L., and B. Wydrowski, "Understanding XCP: equilibrium and fairness", Proceedings of IEEE INFOCOM'05, Miami (Florida), USA, March 2005.

[MacK95] MacKie-Mason, J. and H. Varian, "Pricing Congestible Network Resources", IEEE Journal on Selected Areas in Communications, Advances in the Fundamentals of Networking, Vol. 13, No. 7, pp. 1141-1149, 1995.

[Mascolo01] Mascolo, S., Casetti, Cl., Gerla M., Sanadidi, M.Y., and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links", Proceedings of MOBICOM 2001, Rome, Italy, July 2001.

[Moors02] Moors, T., "A critical review of "End-to-end arguments in system design"", Proceedings of IEEE International Conference on Communications (ICC) 2002, New York City (New Jersey), USA, April/May 2002.

[MPTCP] IETF WG Action: Multipath TCP (mptcp).

[Noel07] Noel, E. and C. Johnson, "Initial Simulation Results That Analyze SIP Based VoIP Networks Under Overload", International Teletraffic Congress (ITC'07), Ottawa, Canada, June 2007.

[Padhye98] Padhye, J., Firoiu, V., Towsley, D., and J. Kurose, "Modeling TCP Throughput: A Simple Model and Its Empirical Validation", University of Massachusetts (UMass), CMPSCI Tech. Report TR98-008, February 1998.

[Pan00] Pan, R., Prabhakar, B., and K. Psounis, "CHOKe: a stateless AQM scheme for approximating fair bandwidth allocation", Proceedings of IEEE INFOCOM'00, Tel Aviv, Israel, March 2000.

[Pap02] Papadimitriou, I. and G. Mavromatis, "Stability of Congestion Control Algorithms using Control Theory with an application to XCP", Technical Report, 2002. <http://www.stanford.edu/class/ee384y/projects/reports/ionnis.pdf>.

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

[RFC793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

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

[RFC1701] Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 1701, October 1994.

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

[RFC2003] Perkins, C., "IP Encapsulation within IP", RFC 2003, October 1996.

[RFC2018] Mathis, M., Mahdavi, J., Floyd, S., and A. Romanow, "TCP Selective Acknowledgment Options", RFC 2018, October 1996.

[RFC2208] Mankin, A., Ed., Baker, F., Braden, B., Bradner, S., O'Dell, M., Romanow, A., Weinrib, A., and L. Zhang, "Resource ReSerVation Protocol (RSVP) -- Version 1 Applicability Statement Some Guidelines on Deployment", RFC 2208, September 1997.

[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 Service", RFC 2475, December 1998.

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

[RFC2637] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and G. Zorn, "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, July 1999.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, August 1999.

[RFC2775] Carpenter, B., "Internet Transparency", RFC 2775, February 2000.

[RFC2784] Farinacci, D., Li, T., Hanks, S., Meyer, D., and P. Traina, "Generic Routing Encapsulation (GRE)", RFC 2784, March 2000.

[RFC2861] Handley, M., Padhye, J., and S. Floyd, "TCP Congestion Window Validation", RFC 2861, June 2000.

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

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

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

[RFC3031] Rosen, E., Viswanathan, A., and R. Callon, "Multiprotocol Label Switching Architecture", RFC 3031, January 2001.

[RFC3032] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

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

[RFC3260] Grossman, D., "New Terminology and Clarifications for Diffserv", RFC 3260, April 2002.

[RFC3517] Blanton, E., Allman, M., Fall, K., and L. Wang, "A Conservative Selective Acknowledgment (SACK)-based Loss Recovery Algorithm for TCP", RFC 3517, April 2003.

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

[RFC3662] Bless, R., Nichols, K., and K. Wehrle, "A Lower Effort Per-Domain Behavior (PDB) for Differentiated Services", RFC 3662, December 2003.

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

[RFC3742] Floyd, S., "Limited Slow-Start for TCP with Large Congestion Windows", RFC 3742, March 2004.

[RFC3985] Bryant, S., Ed., and P. Pate, Ed., "Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture", RFC 3985, March 2005.

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 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., Ed., and YJ. Stein, Ed., "Structure-Agnostic Time Division Multiplexing (TDM) over Packet (SAToP)", RFC 4553, June 2006.

[RFC4614] Duke, M., Braden, R., Eddy, W., and E. Blanton, "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, January 2007.

[RFC4828] Floyd, S. and E. Kohler, "TCP Friendly Rate Control (TFRC): The Small-Packet (SP) Variant", RFC 4828, April 2007.

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

[RFC5033] Floyd, S. and M. Allman, "Specifying New Congestion Control Algorithms", BCP 133, RFC 5033, August 2007.

[RFC5086] Vainshtein, A., Ed., Sasson, I., Metz, E., Frost, T., and P. Pate, "Structure-Aware Time Division Multiplexed (TDM) Circuit Emulation Service over Packet Switched Network (CESoPSN)", RFC 5086, December 2007.

[RFC5087]   Stein, Y(J)., Shashoua, R., Insler, R., and M. Anavi, "Time Division Multiplexing over IP (TDMoIP)", RFC 5087, December 2007.

[RFC5129]   Davie, B., Briscoe, B., and J. Tay, "Explicit Congestion Marking in MPLS", RFC 5129, January 2008.

[RFC5290]   Floyd, S. and M. Allman, "Comments on the Usefulness of Simple Best-Effort Traffic", RFC 5290, July 2008.

[RFC5348]   Floyd, S., Handley, M., Padhye, J., and J. Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification", RFC 5348, September 2008.

[RFC5405]   Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines for Application Designers", BCP 145, RFC 5405, November 2008.

[RFC5622]   Floyd, S. and E. Kohler, "Profile for Datagram Congestion Control Protocol (DCCP) Congestion ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP)", RFC 5622, August 2009.

[RFC5681]   Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681 (Obsoletes RFC 2581), September 2009.

[RFC5783]   Welzl, M. and W. Eddy, "Congestion Control in the RFC Series", RFC 5783, February 2010.

[RFC6040]   Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, November 2010.

[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://heim.ifi.uio.no/michawe/research/projects/corruption/.

[Saltzer84] Saltzer, J., Reed, D., and D. Clark, "End-to-end arguments in system design", ACM Transactions on Computer Systems, Vol. 2, No. 4, November 1984.

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

[Sarola07] Sarolahti, P., Floyd, S., and M. Kojo, "Transport-layer Considerations for Explicit Cross-layer Indications", Work in Progress, March 2007.

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

[Shal10] Shalunov, S., Hazel, G., and J. Iyengar, "Low Extra Delay Background Transport (LEDBAT)", Work in Progress, October 2010.

[Shen08] Shen, C., Schulzrinne, H., and E. Nahum, "Session Initiation Protocol (SIP) Server Overload Control: Design and Evaluation, Principles", Systems and Applications of IP Telecommunications (IPTComm'08), Heidelberg, Germany, July 2008.

[Shin08] Shin, M., Chong, S., and I. Rhee, "Dual-Resource TCP/AQM for Processing-Constrained Networks", IEEE/ACM Transactions on Networking, Vol. 16, No. 2, pp. 435-449, April 2008.

[Thaler07] Thaler, D., Sridharan, M., and D. Bansal, "Implementation Report on Experiences with Various TCP RFCs", Presentation to the IETF Transport Area, March 2007. <http://www.ietf.org/proceedings/07mar/ slides/tsvarea-3/>.

[Tickoo05] Tickoo, O., Subramanian, V., Kalyanaraman, S., and K.K. Ramakrishnan, "LT-TCP: End-to-End Framework to Improve TCP Performance over Networks with Lossy Channels", Proceedings of International Workshop on QoS (IWQoS), Passau, Germany, June 2005.

[TRILOGY] "Trilogy Project", European Commission Seventh Framework Program (FP7), Grant No: 216372, <http://www.trilogy-project.org>.

[Vinnic02] Vinnicombe, G., "On the stability of networks operating TCP-like congestion control," Proceedings of IFAC World Congress, Barcelona, Spain, 2002.

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

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

[Xia05] Xia, Y., Subramanian, L., Stoica, I., and S. Kalyanaraman, "One more bit is enough", ACM SIGCOMM Computer Communication Review, Vol. 35, No. 4, pp. 37-48, 2005.

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

6. Acknowledgments

The authors would like to thank the following people whose feedback and comments contributed to this document: Keith Moore, Jan Vandenabeele, and Larry Dunn (his comments at the Manchester ICCRG and discussions with him helped with the section on packet-congestibility).

Dimitri Papadimitriou's contribution was partly funded by [ECODE], a Seventh Framework Program (FP7) research project sponsored by the European Commission.

Bob Briscoe's contribution was partly funded by [TRILOGY], a research project supported by the European Commission.

Michael Scharf is now with Alcatel-Lucent.

7. Contributors

The following additional people have contributed to this document:

- Wesley Eddy <weddy@grc.nasa.gov>

- Bela Berde <bela.berde@gmx.de>

- Paulo Loureiro <loureiro.pjg@gmail.com>

- Chris Christou <christou_chris@bah.com>

Authors' Addresses

Dimitri Papadimitriou (editor)
Alcatel-Lucent
Copernicuslaan, 50
2018 Antwerpen, Belgium

Phone: +32 3 240 8491
EMail: dimitri.papadimitriou@alcatel-lucent.com

Michael Welzl
University of Oslo, Department of Informatics
PO Box 1080 Blindern
N-0316 Oslo, Norway

EMail: michawe@ifi.uio.no

Michael Scharf
University of Stuttgart
Pfaffenwaldring 47
70569 Stuttgart, Germany

EMail: michael.scharf@googlemail.com

Bob Briscoe
BT & UCL
B54/77, Adastral Park
Martlesham Heath
Ipswich IP5 3RE, UK

EMail: bob.briscoe@bt.com