Alternatives to End-to-End Congestion Control

Dan Duchamp

Computer Science Department
Stevens Institute of Technology
Hoboken, NJ 07030 USA
djd@cs.stevens-tech.edu

***Abstract*—**

Work on congestion control has always been focused on end-to-end transport layer techniques. The reason is that there are only two layers of protocol in the TCP/IP architecture, and IP is limited to being a dumb routing layer. This paper explains a new architecture for the Internet that introduces a third layer of protocol, at the "session layer" above transport. The session layer handles end-to-end delivery verification, freeing the transport layer to focus on congestion control techniques that need not necessarily be end-to-end. Accordingly, the paper proposes an inter-router congestion control protocol that would operate at the transport layer.

I. Introduction

The background for this paper is our work on a generalized "session layer" protocol [1]. The idea is to expand the TCP/IP architecture by adding a protocol at layer 5 (L5). In this new architecture, application software invokes the L5 API to open a connection to the corresponding application on another host. The L5 protocol is responsible for verifying end-to-end data delivery according to the application's requested semantics. Consequently, the transport layer (L4) is relieved of this responsibility.

The initial motivation for this work was to provide a clean model for developing "intermediate services" such as web caches, NAT boxes, etc. For example, an L5 connection between a browser and a web server that runs through two intermediaries (say, a firewall and a web cache) would run over three L4 connections in series: browser-firewall, firewall-cache, and cache-server. Each intermediary would benefit from a terminated L4 connection, and thereby would be freed from concerns such as IP fragmentation, misordered packets, packets traveling divergent routes, etc.

An L5 protocol brings a number of additional advantages that cannot be surveyed here because of lack of space; see the cited paper. One disadvantage of L5 is setup overhead; in large measure, this can be alleviated by caching L4 connections and multiplexing L5 connections over a single long-lived L4 connection.

Making the (big) assumption that L5 becomes widely adopted, what should be the role of transport protocols? The naive answer is to leave transport protocols as they are now – L4 would provide congestion control between each piece of an L5 connection: browser-firewall, firewall-cache, etc. Indeed, since each such L4 connection would be shorter than end-to-end, and since it would stay in place for a longer time, it is reasonable to suppose that overall congestion control would be improved:

- Long-lived L4 connections benefit from the well-documented effects of maintaining congestion control information over a long period.
- Shorter RTTs should lead to less RTT variation and hence faster timeout.
- More points of control should lead to faster and more precise congestion control, where "precise" means that remedy is applied more precisely to the subpart of the network that is experiencing congestion.

Moreover, the presence of L5 offers the chance for increased clarity in protocol design. For example, TCP's cumulative ack plays a role in reliability, flow control, congestion control, and in-order delivery semantics. On one hand, it is impressive to get so much from so little mechanism. On the other hand, heaping so much responsibility on the ack message makes it difficult to change TCP (or any similarly designed protocol) independently in any of the four dimensions.

L5 allows us to ask the fanciful question of what layers 3 and 4 should do (assuming they exist at all) in a from-scratch re-design of the Internet. This paper asks one aspect of that question, namely: how best to do congestion control?

II. Requirements of the Modern Internet

In the original Internet universally-addressable hosts were connected by a "dumb" routing layer. All decisions about whether and when to inject packets into the Internet were made by host software – TCP and UDP modules – and equipment at the routing layer could only route a packet or drop it. It was reasonable to suppose that host software would behave properly because the user community was mostly limited to like-minded individuals who shared the goal of keeping the Internet running well.

That model has been replaced by a "Discrete Internet" in which competing network operators own and manage their own subnetworks, and connect to other subnetworks only with contractual and operational caution. Network operators are keenly interested in maximizing the utility of their assets through traffic engineering (TE). TE includes distributing traffic between A and B among many "acceptable" paths rather than a single "best" path, and distributing different flows between A and B among different paths according to the quality-of-service (QoS) needs of the flows.

Another aspect of the modern Internet is that router software and host software have developed into different markets served by different vendors – designers of router software cannot necessarily depend on designers of host software to share their goals.

In this modern environment, it doesn't make sense to depend on proper end-host behavior to ensure the optimal operation of each network provider's subnetwork, and yet that is the situation today and in the proposed future [4], [2] regarding congestion control. The next section will survey the ways in which dependence on end-to-end congestion control limits the design and operation of the core of the Internet. The final section will then explain how a new architecture that includes L5 might result in superior congestion control.

III. Consequences of End-to-End Congestion Control

Since the task of congestion control is presently assigned to the transport layer, work on congestion control has naturally focused on transport layer improvements. One such improvement is DCCP [2], a congestion-controlled datagram oriented protocol meant as a replacement for UDP. Another improvement is Explicit Congestion Notification (ECN) [4], which is intended to work with ack-ing transport protocols such as TCP and DCCP. ECN allows a router to signal that congestion may arise in the future. Specifically, when a router senses rising queue lengths (but before packet-drop becomes necessary), it "marks" IP datagrams (using a bit in IPv4's TOS field). A receiver host that sees a marked packet passes on that fact to the sending host via a bit in the TCP/DCCP ack header. (There is a final step when the sending host tells the receiver it has received the congestion signal; this too is accomplished using a bit in the transport header.)

All these mechanisms – TCP, DCCP, and ECN – have in common the notion of feeding back a binary signal (congestion is/isn't present) to the sending host's transport software, which is then responsible for curing congestion by reducing its sending rate. TCP currently receives feedback implicitly through packet drops – no ack is received for dropped packets. The ECN proposal makes the feedback explicit. DCCP is designed to work with both ECN and packet-drop.

There are several problems with host-implemented congestion control:

1) The notion of feedback depends on all packets in each direction of a flow traveling the same route. That is, it is assumed that feedback delivered earlier accurately describes the conditions on the same path that future packets will travel.
   Delivering feedback to the sender constrains network operators' approaches to routing and traffic engineering. For example, the naive approach to balancing traffic among equivalent paths – round robin – is out of the question because of the danger it poses to TCP performance. As another example, multipath routing must be performed at the granularity of *port-flow*, not *host-flow*, per-packet, or other granularities that may be convenient for TE.
2) There is a greater danger of incorrect implementation. Host software is out of the hands of those who manage networks and build routers. Implementers of host software are probably less motivated to get algorithms exactly right and/or incorporate the latest improvements. RFC 2525 [3] documents mis-implementation of basic TCP congestion control algorithms, more than a decade after their widespread promulgation.
   ECN, although a sensible and beneficial idea, adds complexity and therefore will inevitably lead to another round of mis-implementation. For example, research on ECN has already given rise to further research on how to prevent hosts from subverting ECN and how to prevent denial-of-service attacks that throttled L4 connections by improperly signaling congestion [6].

3) Congestion may be caused by improper aggregation: the fault of routing and TE, not any individual host. That is, congestion may occur because of the momentary confluence of many small flows ("mice") rather than because of a few "elephants." However, host-based congestion control always treats congestion as a "global" problem even when it is a "local" problem that could be alleviated by TE.
4) There is a greater time lag in responding to congestion. A host must wait until a timer pops, until it receives multiple duplicate acks, or, if ECN is in use, for a round-trip time.
5) The overloading of L4 acks to signal absence or (with ECN) presence of congestion leads to intellectual muddle in protocol design, as a single mechanism is used for multiple purposes.

   A clearer mapping between protocol messages and functions might lead to protocols that are more easily composable and whose performance can be analyzed and predicted more easily.
6) Use of acks also causes bandwidth to be consumed by small, inefficient packets in part because of the need for a steady stream of acks to "clock" the sender's congestion control mechanism.
7) The goals of user-operated hosts are different from the goals of provider-operated routers. This fact, plus the desire not to leak too much information between layers effectively limits the information that the network can report to hosts when congestion control is needed. Both packet-drop and ECN's "congestion experienced" bit deliver a simple binary signal.

   It is not hard to imagine that – if in-network congestion control mechanisms existed – they could exchange more, more exact, and more timely information among themselves and to hosts.

Our argument is not against host-provided congestion control, but rather against depending *exclusively* on the host to interpret a *binary* signal generated by the network. The next section proposes a mechanism that is richer in both dimensions: routers signal congestion among themselves as well as (if needed) to hosts; and the signals carry much more than a single bit of information.

IV. Proposed Solution

The basic idea is for routers to keep each other continually informed of the conditions of their queues. As a certain router becomes more congested, its upstream neighbors discover the fact and throttle themselves. If throttling fails to cure the downstream congestion, or if congestion results at any of the now-throttled upstream routers, then throttling continues upstream. Upstream throttling may continue until, at the limit, hosts are reached. However, congestion is initially assumed to be a "local" problem, with the problem becoming ever more global only as local remedies fail.

Unthrottling happens naturally because the information passed among routers is not a discrete simple binary signal ("I am congested now"), but rather a continuous and rich measure of conditions downstream. When downstream conditions tighten, upstream throttles. When downstream conditions ease, upstream un-throttles.

It is typical for core routers to be linked to very few others; Spring et al. report that approximately 90% are attached to 5 or fewer neighbors [5], and this figure includes PoP routers. Therefore, the amount of information exchanged should not be burdensome, while an indication of congestion can have wide effect after only a few levels of upstream notification.

Furthermore, adjacent core routers are set up by management policy, not by automatic neighbor discovery. Therefore, it is reasonable to suppose that adjacent core routers could perform a somewhat heavyweight initialization procedure that would set the parameters for a congestion control protocol that they would run among themselves. This would allow different router implementations, with different queueing measures and policies, to interact. The central requirement of the inter-router congestion control protocol would be a syntax whereby any router could express to any other its *relative* conditions at the moment.

The information exchanged by routers could be rich enough to help cure the congestion. For example, if a router reported the condition of all its outgoing queues, upstream routers might be able to select different routes – routers in a common AS typically already share a common route map and/or set of MPLS LSPs for the AS.

References

[1] D. Duchamp. *The Discrete Internet and What to Do About It.* New York Metro Area Networking Workshop, September 2002.

[2] E. Kohler, M. Handley, S. Floyd. *Designing DCCP: Congestion Control Without Reliability.* In submission, May 2003.

[3] V. Paxson et al. *Known TCP Implementation Problems.* RFC 2525. March 1999.

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

[5] N. Spring, R. Mahajan, D. Weatherall. *Measuring ISP Topologies with Rocketfuel.* in Proc. ACM SIGCOMM '02, pp. 133-145, August 2002.

[6] N. Spring, D. Weatherall, D. Ely. *Robust Explicit Congestion Notification (ECN) Signaling with Nonces.* RFC 3540. June 2003.