# Breaking Up the Transport Logjam

Bryan Ford
Massachusetts Institute of Technology[*]
baford@mit.edu

Janardhan Iyengar
Franklin & Marshall College
jiyengar@fandm.edu

## ABSTRACT

Current Internet transports conflate transport semantics with endpoint addressing and flow regulation, creating roadblocks to Internet evolution that we propose to address with a new layering model. Factoring endpoint addressing (port numbers) into a separate *Endpoint Layer* permits incremental rollout of new or improved transports at OS or application level, enables transport-oblivious firewall/NAT traversal, improves transport negotiation efficiency, and simplifies endpoint address space administration. Factoring congestion control into a separate *Flow Layer* cleanly enables in-path performance optimizations such as on satellite or wireless links, permits incremental rollout of new congestion control schemes within administrative domains, frees congestion control evolution from the yoke of "TCP-friendliness," and facilitates multihoming and multipath communication. Though this architecture is ambitious, existing protocols can act as starting points for the new layers—UDP or UDP-Lite for the Endpoint Layer, and Congestion Manager or DCCP for the Flow Layer—providing both immediate deployability and a sound basis for long-term evolution.

## 1. INTRODUCTION

Typical transport protocols combine several functions, such as identifying application endpoints via port numbers [38,49], providing end-to-end congestion control [27], utilizing alternate communication paths [33,46], and implementing reliable/ordered communication [37,46,49]. Lumping these functions into one layer has made the transport layer brittle and difficult to evolve, however, by preventing evolution of individual transport functions without affecting *the entire transport layer*. Since firewalls and NATs [45] must understand transport headers to extract port numbers, for example, new transports [28, 46] are almost undeployable because they cannot pass through existing middleboxes. Similarly, new congestion control schemes [20] and performance enhancing proxies [11] cannot be deployed on specific segments of a communication path without breaking end-to-end semantics [41] and fate-sharing properties [16].

To remove these evolutionary roadblocks, we propose splitting the Transport Layer into (at least) three separate layers, shown in Figure 1. We factor out the function of identifying logical communication endpoints—traditionally represented as 16-bit port numbers—into an *Endpoint Layer* protocol to be shared among transports. We factor out congestion control and other performance-related mechanisms into a separate *Flow Regulation Layer*, or simply *Flow Layer*. The services remaining in the Transport Layer are limited to providing the end-to-end communication semantics needed by higher-level layers, such as reliability, ordering, and error recovery.

[*]Now at the Max Planck Institute for Software Systems (MPI-SWS)

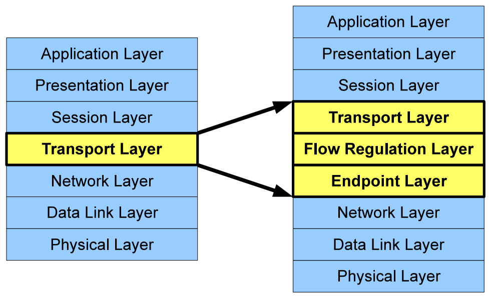

**Figure 1: Breaking up the Transport Layer**

In contrast with prior work that factored out these functions for specific technical reasons [6, 18, 28, 50], our focus is on identifying and addressing *evolutionary* impediments to Transport Layer development. Our primary contribution is a new architectural model that better facilitates evolution, and that places a variety of existing, often mutually exclusive "transport hacks" into a clean and interoperable framework.

Section 2 details the purpose, architecture, and practical implications of our Endpoint Layer, and Section 3 similarly details our Flow Regulation Layer. Section 4 outlines issues in implementing and further evolving the Endpoint, Flow, and Transport layers, and Section 5 concludes.

## 2. THE ENDPOINT LAYER

Our first modification to the classic Internet architecture is separating the function of identifying *logical endpoints* or *ports* out of transport protocols and into a common underlying *Endpoint Layer*. We view the Endpoint Layer as an extension to the Network Layer: where the Network Layer provides *inter-host addressing and routing* via IP addresses, the Endpoint Layer provides *intra-host addressing and routing* via port numbers. The Endpoint Layer does not otherwise affect the underlying best-effort delivery service: higher layers are responsible for congestion control, ordering, and reliability. All higher layers ideally reside atop a single Endpoint protocol, sharing one endpoint address space per host.

Our insight is that building new transports atop a common Endpoint Layer, instead of atop IP as in the current model, may facilitate flexibility and protocol evolution in several ways. We first address architectural foundations, followed by practical benefits of the proposed model. We leave Endpoint Layer implementation issues to Section 4, which proposes reusing UDP [38] as an already widely supported "Endpoint Layer protocol," and then suggests paths toward richer functionality.

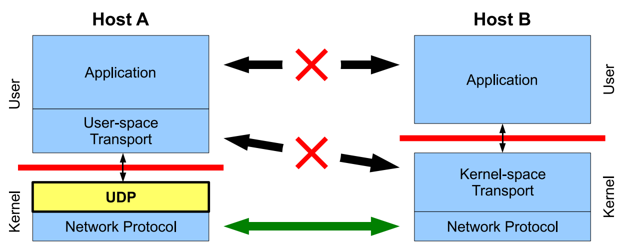

**Figure 2: A UDP-based user-space transport *cannot* interoperate with a "native" IP-based kernel-space transport.**

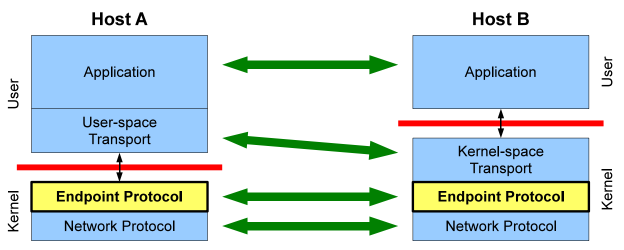

**Figure 3: A UDP-based user-space transport interoperates with a UDP-based kernel-space transport.**

## 2.1 Architectural Perspective

All standard Internet transports [28, 37, 38, 46, 49] multiplex transport sessions onto a host's few IP address(es) via 16-bit port numbers. Each transport implements this multiplexing separately and embeds port numbers in its own transport header, making port numbers a common design pattern but not a shared facility. Nevertheless, each port space tends to be functionally equivalent; the IANA historically assigns well-known ports consistently across transports although the port spaces are technically independent.

Embedding port numbers into transports is consistent with the OSI reference model [56], where each layer provides its own space of *service access points* (SAPs) for higher layers to bind to: IP addresses correspond to *Network-SAPs* (NSAPs), port numbers to *Transport-SAPs* (TSAPs), and OSI additionally has *Session-SAPs* and *Presentation-SAPs*. The full "identity" of an endpoint consists of the SAPs of all layers bundled together: IP address and port number on the Internet, all four SAPs in OSI. This *layered multiplexing* design has appeal but causes known problems: Tennenhouse argued against it due to the difficulty of real-time scheduling across layers [50], and Feldmeier elaborated on several related issues [18].

An alternative approach is to treat the intra-host addressing provided by port numbers or SAPs as an extension to the inter-host addressing already provided by the Network Layer, and implement this intra-host addressing *once* in a facility shared by all higher layers. In Sirpent [14], intra-host addresses (port numbers) are part of the source routes the network layer uses for routing. An analogous design with CIDR addressing would be to assign each physical host or network interface a whole "virtual subnet" of addresses representing the logical endpoints on that physical host. It may be too late to merge port numbers into IP addresses, but our Endpoint Layer revisits the idea of *sharing* one endpoint space among upper-level protocols instead of each transport implementing its own.

## 2.2 Practical Benefits

Independent of the concerns of Tennenhouse and Feldmeier, factoring out endpoint multiplexing brings several practical benefits that are relevant today: transport implementation flexibility, firewall/NAT traversal, and transport protocol negotiation.

### 2.2.1 *Transport Implementation Flexibility*

The IP header's 8-bit Protocol field was intended to distinguish between only a few standard transport protocols, not between many application-level endpoints, so most operating systems prohibit unprivileged applications from "hooking" IP Protocol numbers in the way they can allocate and use ports. The OS thus reserves the right to implement new "first-class" transports. If an application wishes to deploy its own transport protocol that is not yet supported by the host OS, it must layer the new transport atop UDP. The resulting application-level transport not only has second-class status but is *unable to interoperate* with a first-class OS-level implementation of the same transport on another host, as shown in Figure 2. This restriction creates a barrier to the deployment of new transports, since the easiest way to deploy new protocols incrementally is often to bundle them with the applications that need them.

If new transports are built atop an Endpoint Layer, however, applications can easily ship with new transports implemented in user-space libraries requiring no special privilege. Once a transport begins migrating into OS kernels, kernel-level and user-level implementations of the same transport can remain interoperable, as shown in Figure 3.

### 2.2.2 *Transport-Independent Middlebox Traversal*

For better or worse, *middleboxes* such as firewalls and network address translators (NATs) are now ubiquitous, and most of them are sensitive to the full endpoints of a given flow: not only IP addresses but port numbers as well. Since each transport traditionally implements its own port space, middleboxes must parse transport headers, and so only the few already-ubiquitous transports—TCP and UDP—can traverse most middleboxes. New transports like SCTP [46] and DCCP [28] that are designed to run directly atop IP thus cannot traverse most middleboxes. NAT proliferation has in effect shifted the Internet's "narrow waist"—the ubiquitous interface atop which new protocols may be built and reliably deployed—upward to encompass not just IP but also TCP and UDP [40].

By building new transports atop a shared Endpoint Layer, middleboxes need to understand only Endpoint Layer and not Transport Layer headers. Middleboxes can still recognize and optimize the handling of specific transport protocols if desired, but doing so is no longer a *prerequisite* for traversal. The Endpoint Layer also provides a clean space for mechanisms allowing hosts to "advertise" endpoints intended to be publicly reachable, enabling middleboxes to create persistent bindings for them as policy permits—a demand currently met via ad hoc, transport-specific mechanisms such as those in UPnP [53].

### 2.2.3 *Negotiation of Alternative Transports*

Many application protocols such as RPC and SIP can use several alternative transports. Every application packet is traditionally associated with *exactly one* transport protocol, how-

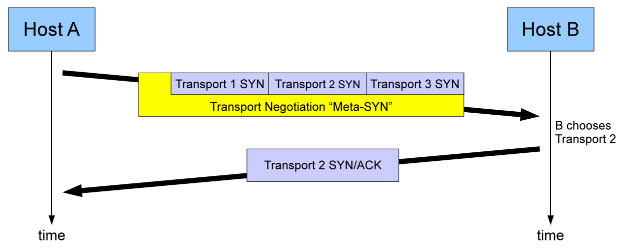

**Figure 4: Applications can negotiate among several UDP-based transports with no extra round trips.**

ever, via the IP header's Protocol field. Negotiating *which* transport to use for a communication session therefore requires the initiating application either to use a special transport exchange just for this negotiation, or to open new sessions "speculatively" for each supported transport, only to continue using the most preferred one that succeeds and shut down the rest.

Building transports atop a shared Endpoint Layer with one port space, in contrast, leaves transport identification and negotiation under the application's control. The Internet already follows this design philosophy for Session Layer and Presentation Layer functions, leaving their negotiation up to applications (e.g., HTTP's persistent streams and content encodings); our architecture extends this flexibility to the Transport Layer.

Without prescribing specific mechanisms, we suggest one way an application in our model might combine transport negotiation with the initial exchange of the selected transport, avoiding unnecessary round-trips or state setup. The application first locally requests from each supported transport a copy of the "SYN" packet the transport *would* send to initiate a new session. The application collects the SYN packets for all such transports, bundles them together into one "Meta-SYN" packet, and sends the Meta-SYN to the responding endpoint, as shown in Figure 4. The responding application breaks apart the Meta-SYN, passes the SYN for some transport it supports to its implementation of that transport, and subsequent communication proceeds normally via that transport. This design assumes that packets for different transports are distinguishable from each other and from Meta-SYN packets; the application might interpose a minimal header for this purpose if required.

A side effect of making endpoints transport-independent is to close the debate over whether to allocate well-known ports across several transports at once. IANA would need to manage only one port space, and existing applications could adopt new transports without having to register new ports for each.

## 3. THE FLOW REGULATION LAYER

Our *Flow Regulation Layer*, or simply *Flow Layer*, manages the performance of a flow between a pair of endpoints. The Flow Layer takes the underlying best-effort delivery service, which typically provides limited information about available bandwidth and other network characteristics, and builds a *flow-regulated best-effort delivery service*, which "knows" how to regulate the flow of packets for best use of the available path(s). The Flow Layer implements congestion control [27] and may encapsulate performance-related mechanisms such as performance enhancing proxies [11], end-to-end multihoming [46], multipath transmission [33], and forward error correction.

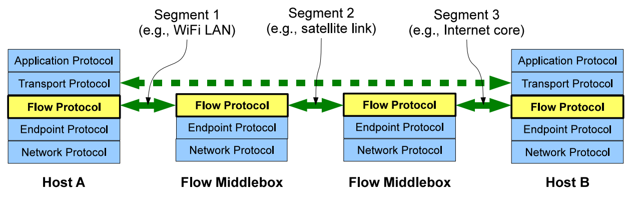

**Figure 5: An end-to-end path composed of multiple Flow Layer segments. Flow middleboxes can optimize network performance based on the properties of a specific segment, such as a satellite link.**

The idea of factoring congestion control into a separate protocol is embodied in the Congestion Manager (CM) [6] and Datagram Congestion Control Protocol (DCCP) [28]; these protocols offer starting points for our Flow Layer, as discussed in Section 4. Beyond merely factoring out congestion control, our insight is that the Flow Layer is a clean place to implement many performance-related mechanisms, enabling them to benefit many transports, and avoiding interference with transport reliability or end-to-end fate-sharing [16]. The following sections explore several such performance enhancement techniques: dividing communication paths into segments for performance tuning, utilizing multiple redundant communication paths, and aggregating flows to improve fairness or efficiency.

### 3.1 Path Segmentation

Our architecture permits devices in the network, called *flow middleboxes*, to interpose on Flow Layer communication by dividing a path into *segments*, as shown in Figure 5. Flow middleboxes "split" the path by terminating one segment's Flow Layer connection and initiating a new one for the next segment. Each segment may consist of several Network Layer hops; path segmentation does not imply hop-by-hop congestion control [34], although the latter may be viewed as a limit case of path segmentation.

Flow middleboxes do not touch Transport Layer headers or payloads, so they are compatible with any transport protocol. Since flow middleboxes affect only communication performance and not transport semantics, they serve in precisely the role for which the end-to-end principle [41] justifies such in-network mechanisms. In contrast with the analogous technique of TCP splitting [4], where transport state may be lost if a middlebox fails after acknowledging data received on one segment but before transmitting it on the next, Flow Layer splitting preserves end-to-end fate-sharing [16] because flow middleboxes hold only performance-related soft state.

Motivations for splitting a communication path into individually congestion-controlled segments include performance benefits from reduced RTT, specialization to network characteristics, and administrative isolation. We expore each in turn.

#### 3.1.1 *Performance Benefits from Reduced RTT*

A TCP flow's throughput is adversely affected by large round-trip time (RTT), especially in competition with flows of smaller RTT [19]. In addition, since information requires one RTT to propagate around the control loop, *any* end-to-end congestion control scheme's responsiveness to changing conditions is limited by RTT. Subdividing a communication path into independently congestion-controlled segments reduces each segment's RTT to a fraction of the path's total RTT, which can improve both throughput and responsiveness. This benefit has been noted in the context of hop-by-hop congestion control schemes for packet-switched [34], cell-switched [29], and wireless networks [54]. The Logistical Session Layer [48] similarly leverages this effect to improve wide-area grid performance. Our Flow Layer thus provides a semantically clean way to obtain the benefits of shorter RTTs within segmented paths.

### 3.1.2 *Specialization to Network Characteristics*

The best congestion control scheme for a communication path often depends on the characteristics of the underlying network [8]. Classic TCP congestion control [27] performs well on wired LANs and the Internet core, but poorly on networks that are loss-prone due to transmission errors or mobility, and on long-delay connections such as satellite links or wireless wide-area networks. Since integrating diverse networks is a fundamental goal of the Internet [16], we must assume that any communication path may traverse several network types, each of which might place conflicting requirements on any single end-to-end congestion control scheme. New end-to-end schemes are available for high-bandwidth, long-delay links [20], and others for mobile ad hoc networks [31], but will any one scheme perform well on a path that includes links of both types (and others)? Path segmentation in the Flow Layer provides a clean method of specializing congestion control to the characteristics of individual path segments while avoiding the pitfalls of traditional performance enhancing proxies [11].

Other fixes are available for specific performance issues [5], but we feel that none of them solves the general network path heterogeneity problem. A "sledgehammer approach" is to open parallel TCP streams over one path, either at transport [44] or application level [2], boosting throughput at the cost of fairness by amplifying TCP's aggressiveness [21]. TCP snooping [7] enables intermediate nodes to retransmit lost packets and suppress duplicate acknowledgments without violating TCP's semantics, but this technique is transport-specific and does not allow adjacent segments to run independent congestion control schemes. Many approaches assume that only the "last hop" of a path requires specialization—an assumption violated by important scenarios such as wireless mesh networks [1]. In contrast, our architecture supports any number of segments and permits independent performance tuning of each.

### 3.1.3 *Administrative Isolation*

Even where one end-to-end congestion control scheme may be technically adequate, the Internet's inertia makes it politically difficult to agree on, evolve, and deploy new end-to-end schemes. Any new scheme encounters resistance unless it is "TCP-friendly"—no more aggressive than TCP Reno—since the new scheme's flows will compete with Reno streams "in the wild." But since the Internet does not *enforce* TCP-friendliness [21], selfish or unaware users can and do deploy unfairly aggressive mechanisms anyway—e.g., in the form of TCP-unfair UDP flows [15] or concurrent TCP flows [30].

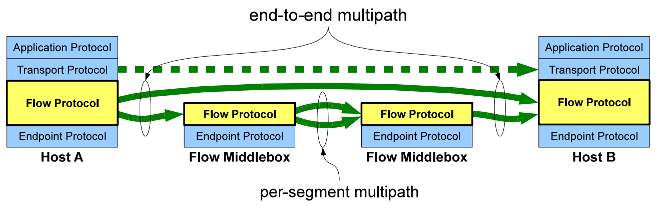

**Figure 6: Flow Layer multipath communication example. The multihomed hosts use two end-to-end paths, one passing through a pair of middleboxes implementing an in-network multipath segment.**

Path segmentation offers an incremental solution to congestion control evolution: split the Flow Layer path at administrative boundaries, and deploy the new scheme only on segments traversing domains in which the scheme has been adequately tested and approved, preserving TCP-friendliness on other segments. Path segmentation allows network administrators to roll out a new scheme *one administrative domain at a time*, and homogenize the congestion control algorithms used within their domain if desired, ensuring that the new scheme's flows compete only with each other within the domain and not with Reno flows or other arbitrary schemes deployed by end hosts.

Even for end-to-end streams not conforming to our architecture—e.g., flows with congestion control in the Transport Layer or no congestion control at all—homogeneous congestion control can still be enforced within a domain if needed, by encapsulating such streams in a Flow Layer "tunnel" while crossing that domain. Our architecture thus provides a clean framework for proposed mechanisms that use per-flow state at border routers to implement new congestion control schemes within a domain [47], or to enforce TCP-friendliness [39] or differential service agreements [24].

## 3.2 Multipath Communication

There are many ways to exploit alternative network paths to improve reliability [23], balance load [36], or enhance security [32]. To be deployable, however, a multipath scheme must be compatible with upper layer protocols designed assuming single-path routing, and must remain interoperable with single-path routing domains. Our architecture addresses these deployment issues by permitting end hosts and flow middleboxes to implement multipath communication end-to-end or in the network, as shown in Figure 6.

### 3.2.1 *Flow Layer Multihoming*

The Flow Layer provides a clean place to implement end-to-end *multihoming*: binding several endpoints together to provide multiple paths over the existing routing infrastructure. In contrast with transport multihoming [33, 46], multihoming in the Flow Layer can benefit any transport without interfering with transport semantics. An address rewriting mechanism similar to shim6 [43] in the Flow Layer can make all of a host's endpoints appear as one to these transports.

Path segmentation in our architecture can also facilitate the incremental deployment of multipath routing. A multipath routing protocol may be deployed within an administrative domain, surrounded by flow middleboxes that can exploit avail-

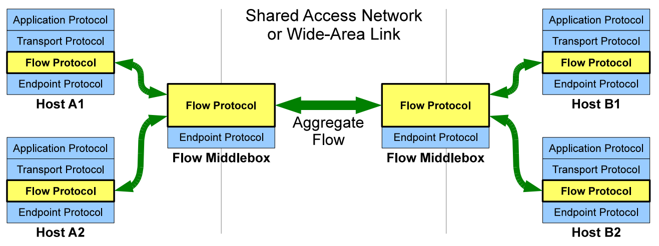

**Figure 7: Flow Layer aggregation example containing two end-to-end flows, which appear as one flow to the intermediate network.**

able paths in flow segments crossing that domain, without affecting external segments (see Figure 6). Alternatively, or simultaneously, a multi-site organization might deploy flow middleboxes at site boundaries to distribute inter-site traffic across redundant wide-area links.

### 3.2.2 *Coping with Path Diversity in Upper Layers*

Naïvely distributing packets among multiple paths with varying delay, whether end-to-end or in-network, can confuse the congestion control and reliability mechanisms of existing transports [10]. In our architecture, a multihomed Flow Layer can avoid this confusion by implementing per-path congestion control, but the Transport Layer remains responsible for retransmission and thus vulnerable to similar confusion. To support arbitrary transports, therefore, a multihomed Flow Layer needs to preserve the illusion of single-path delivery, either by using only one path at once as SCTP does [46], or through order-preserving traffic dispersion [23].

Multipath-aware transports [26] and applications [3] can benefit from the ability to maintain per-path state and explicitly associate packets with paths. Through a simple path indexing mechanism inspired by *path splicing* [35], which we do not elaborate here for space reasons, a multipath Flow Layer in our architecture can expose alternative paths to upper layer protocols capable of using them, while retaining compatibility with multipath-oblivious protocols.

## 3.3 Flow Aggregation

Finally, the Flow Layer provides a clean point at which to *aggregate* related flows when desired, so that the intervening network treats the aggregate as one flow (see Figure 7). Flow aggregation can provide several benefits including reuse of congestion control state and improved fairness.

### 3.3.1 *Reuse of Congestion Control State*

Since an aggregate of many transport instances is typically longer-lived and represents more traffic than any of its constituents, measurements of the aggregate's characteristics can benefit from a longer history and more samples. Transport extensions have been proposed to aggregate congestion control state across reincarnations of one transport session [12], across concurrent sessions [51], across transport protocols [6], and across hosts in an edge network [55].

Placing optimizations such as these in the Flow Layer allows arbitrary transports to benefit from them, and permits aggregation to be performed cleanly within the network as well as end-to-end. In our architecture, for example, a flow middlebox can aggregate congestion control state across the hosts in an edge network and use that information to optimize flows crossing that middlebox transparently, without requiring end host modifications as in TCP/SPAND [55].

### 3.3.2 *Fairness Control*

TCP's per-stream "fairness" notion often fails to match the expectations of users and network operators [13]; Flow Layer aggregation may be useful to implement higher-level fairness policies. For example, an ISP may want each *customer* to get equal bandwidth at bottlenecks in its network, regardless of whether a customer uses few transport instances (web browsing, SSH) or many (BitTorrent). To implement such a policy, the ISP could deploy flow middleboxes at its borders that aggregate all segments crossing its network into one "macro-flow": since each macro-flow has one congestion control context, each macro-flow gets an equal share of congestion bottleneck bandwidth. Most such macro-flows will connect one customer's access router to one of a few upstream network attachment points, so this meta-flow fairness should approximate a per-customer fairness policy. Flow aggregation can thus implement policies similar to those motivating hierarchical fair queuing schemes [9], without changing interior routers.

## 4. IMPLEMENTATION AND EVOLUTION

One of the benefits of the proposed architecture is that existing protocols already provide starting points for implementing its new layers. Since these existing protocols were designed in the traditional architectural framework, however, the fit is not perfect, so further development will be needed.

- **Endpoint Layer:** UDP [38] provides a pervasive first approximation to our Endpoint Layer. Viewing UDP not as transport but as implementing a common endpoint space to be shared by all (new) transports, it becomes worthwhile to consider evolving this shared endpoint space, to support larger port numbers or service names for instance [52]. Also needed are extensions enabling NATs and firewalls to detect which endpoints within a private network are intended to be publicly reachable, and create persistent bindings for them as policy permits. Our intent is for the Endpoint Layer to provide these services, with incremental deployment facilitated by dual-stack Endpoint Layer gateways that map between UDP and the new Endpoint Protocol.
- **Flow Regulation Layer:** Both DCCP [28] and CM [6] approximately implement our Flow Layer, and each has unique features we would like to see combined in one protocol. CM offers aggregation of congestion control state across flows and a packet transmission API that facilitates application-layer framing (ALF) [17], whereas DCCP provides explicit negotiation of congestion control schemes. We also need to examine how to reposition them atop the Endpoint Layer, and to develop extensions supporting Flow Layer optimizations such as path segmentation, multipath communication, and flow aggregation.
- **Transport Layer:** Finally, new Transport Layer protocols will build upon the Flow Layer to offer communication abstractions such as reliable byte streams [49], reliable data-

grams [37], media frames [42], multi-streams [46], or structured streams [22]. We need to consider how to reposition transports atop the Flow Layer in an incremental and backward-compatible way, and how the absence of congestion control in the Transport Layer may impact transport mechanisms such as (fast) retransmit and receive window control.

## 5. CONCLUSION

Although the OSI protocol stack has been dead for years, its layering model remains the standard frame of reference for the Internet, and aspects of its layering model have created serious roadblocks to Internet evolution. By factoring endpoint addressing into a common Endpoint Layer instead of distributing it among transports as in OSI, we obtain more flexibility in transport implementation and deployment, transport-oblivious firewall/NAT traversal, and more efficient transport negotiation. Similarly, by factoring congestion control into an intermediate Flow Layer, we decouple performance-oriented flow regulation from transport semantics, enabling the clean, modular, and incremental deployment of a host of performance optimizations both end-to-end and in the network, without interfering with transport reliability or fate-sharing. The new architectural model therefore appears promising, although many protocol details remain to be worked out.

Our model may appear to make the Internet architecture more complex, but we believe this complexity has already been forced upon us via the patchwork of interposers that have proliferated across the Internet [11, 25]. Our proposal provides a framework in which to fit these interposers together cleanly, recognizing and satisfying the needs that have led to the prevalence of these middleboxes. This project is ambitious, and many unresolved issues remain, such as NAT traversal details, buffering issues in flow middleboxes, APIs and interfaces between the new layers, and cross-layer dependencies. We hope to resolve these issues as we work out mechanisms and protocols to implement the new architecture.

## 6. REFERENCES

[1] I. F. Akyildiz, X. Wang, and W. Wang. Wireless mesh networks: a survey. *Computer Networks*, 47(4), Mar. 2005.

[2] M. Allman, H. Kruse, and S. Ostermann. An application-level solution to TCP's satellite inefficiencies. In *1st WOSBIS*, Nov. 1996.

[3] J. Apostolopoulos et al. On multiple description streaming with content delivery networks. In *INFOCOM*, June 2002.

[4] A. V. Bakre and B. Badrinath. Implementation and performance evaluation of indirect TCP. *IEEE Transactions on Computers*, 46(3):260–278, Mar. 1997.

[5] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz. A comparison of mechanisms for improving TCP performance over wireless links. *IEEE Transactions on Networking*, 5(6), Dec. 1997.

[6] H. Balakrishnan, H. S. Rahul, and S. Seshan. An integrated congestion management architecture for Internet hosts. In *SIGCOMM*, Sept. 1999.

[7] H. Balakrishnan, S. Seshan, E. Amir, and R. H. Katz. Improving TCP/IP performance over wireless networks. In *1st MOBICOM*, Nov. 1995.

[8] C. Barakat, E. Altman, and W. Dabbous. On TCP performance in a heterogeneous network: A survey. *IEEE Communications Magazine*, 38(1):40–46, Jan. 2000.

[9] J. C. R. Bennett and H. Zhang. Hierarchical packet fair queueing algorithms. In *SIGCOMM*, pages 143–156, Aug. 1996.

[10] E. Blanton and M. Allman. On making TCP more robust to packet reordering. *Computer Communications Review*, 32(1), Jan. 2002.

[11] J. Border et al. Performance enhancing proxies intended to mitigate link-related degradations, June 2001. RFC 3135.

[12] R. Braden. T/TCP – TCP extensions for transactions, July 1994. RFC 1644.

[13] B. Briscoe. Flow rate fairness: Dismantling a religion. *Computer Communications Review*, 37(2):63–74, Apr. 2007.

[14] D. R. Cheriton. Sirpent: A high-performance internetworking approach. In *SIGCOMM*, Sept. 1989.

[15] J. Chung, Y. Zhu, and M. Claypool. FairPlayer or FoulPlayer? — head to head performance of RealPlayer streaming video over UDP versus TCP. Technical Report WPI-CS-TR-02-17, Worcester Polytechnic Institute, May 2002.

[16] D. D. Clark. The design philosophy of the DARPA Internet protocols. In *SIGCOMM*, Aug. 1988.

[17] D. D. Clark and D. L. Tennenhouse. Architectural considerations for a new generation of protocols. In *SIGCOMM*, pages 200–208, 1990.

[18] D. C. Feldmeier. Multiplexing issues in communication system design. In *SIGCOMM*, Sept. 1990.

[19] S. Floyd. Connections with multiple congested gateways in packet-switched networks, part 1: One-way traffic. *ACM CCR*, 21(5):30–47, Oct. 1991.

[20] S. Floyd. HighSpeed TCP for large congestion windows, Dec. 2003. RFC 3649.

[21] S. Floyd and K. Fall. Promoting the use of end-to-end congestion control in the internet. *Transactions on Networking*, 7(4):458–472, Aug. 1999.

[22] B. Ford. Structured streams: a new transport abstraction. In *SIGCOMM*, Aug. 2007.

[23] E. Gustafsson and G. Karlsson. A literature survey on traffic dispersion. *IEEE Network*, 11(2):28–36, Mar. 1997.

[24] A. Habib and B. Bhargava. Unresponsive flow detection and control using the differentiated services framework. In *PDCS*, Aug. 2001.

[25] M. Holdrege and P. Srisuresh. Protocol complications with the IP network address translator, Jan. 2001. RFC 3027.

[26] J. R. Iyengar, P. D. Amer, and R. Stewart. Concurrent multipath transfer using SCTP multihoming over independent end-to-end paths. *Transactions on Networking*, 14(5):951–964, Oct. 2006.

[27] V. Jacobson. Congestion avoidance and control. pages 314–329, Aug. 1988.

[28] E. Kohler, M. Handley, and S. Floyd. Datagram congestion control protocol (DCCP), Mar. 2006. RFC 4340.

[29] H. T. Kung and A. Chapman. The FCVC (flow-controlled virtual channels) proposal for ATM networks: A summary. In *1st ICNP*, Oct. 1993.

[30] Y. Liu, W. Gong, and P. Shenoy. On the impact of concurrent downloads. In *WSC*, 2001.

[31] C. Lochert, B. Scheuermann, and M. Mauve. A survey on congestion control for mobile ad-hoc networks. *WCMC*, 7(5):655–676, June 2007.

[32] W. Lou and Y. Fang. A multipath routing approach for secure data delivery. In *MILCOM*, Oct. 2001.

[33] L. Magalhaes and R. Kravets. Transport level mechanisms for bandwidth aggregation on mobile hosts. In *9th ICNP*, Nov. 2001.

[34] P. P. Mishra and H. Kanakia. A hop by hop rate-based congestion control scheme. In *SIGCOMM*, Aug. 1992.

[35] M. Motiwala et al. Path Splicing. In *SIGCOMM*, Aug. 2008.

[36] S. Murthy and J. Garcia-Luna-Aceves. Congestion-oriented shortest multipath routing. In *INFOCOM*, Mar. 1996.

[37] C. Partridge and R. Hinden. Version 2 of the reliable data protocol (RDP), Apr. 1990. RFC 1151.

[38] J. Postel. User datagram protocol, Aug. 1980. RFC 768.

[39] A. Rangarajan and A. Acharya. ERUF: Early regulation of unresponsive best-effort traffic. In *7th ICNP*, Oct. 1999.

[40] J. Rosenberg. UDP and TCP as the new waist of the Internet hourglass, Feb. 2008. Internet-Draft (Work in Progress).

[41] J. H. Saltzer, D. P. Reed, and D. D. Clark. End-to-end arguments in system design. *TOCS*, 2(4):277–288, Nov. 1984.

[42] H. Schulzrinne et al. RTP: A transport protocol for real-time applications, July 2003. RFC 3550.

[43] Site multihoming by IPv6 intermediation (shim6). `http://www.ietf.org/html.charters/shim6-charter.html`.

[44] H. Sivakumar, S. Bailey, and R. Grossman. Psockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *SC2000*, Nov. 2000.

[45] P. Srisuresh and K. Egevang. Traditional IP network address translator (Traditional NAT), Jan. 2001. RFC 3022.

[46] R. Stewart, ed. Stream control transmission protocol, Sept. 2007. RFC 4960.

[47] I. Stoica, S. Shenker, and H. Zhang. Core-stateless fair queueing: A scalable architecture to approximate fair bandwidth allocations in high speed networks. In *SIGCOMM*, Aug. 1998.

[48] M. Swany. Improving throughput for grid applications with network logistics. In *SC2004*, Nov. 2004.

[49] Transmission control protocol, Sept. 1981. RFC 793.

[50] D. L. Tennenhouse. Layered multiplexing considered harmful. In *1st International Workshop on Protocols for High-Speed Networks*, May 1989.

[51] J. Touch. TCP control block interdependence, Apr. 1997. RFC 2140.

[52] J. Touch. A TCP option for port names, Apr. 2006. Internet-Draft (Work in Progress).

[53] UPnP Forum. Internet gateway device (IGD) standardized device control protocol, Nov. 2001. http://www.upnp.org/.

[54] Y. Yi and S. Shakkottai. Hop-by-hop congestion control over a wireless multi-hop network. *Transactions on Networking*, 15(1):133–144, Feb. 2007.

[55] Y. Zhang, L. Qiu, and S. Keshav. Speeding up short data transfers: Theory, architectural support and simulation results. In *10th NOSSDAV*, June 2000.

[56] H. Zimmermann. OSI reference model—the ISO model of architecture for open systems interconnection. *Transactions on Communications*, 28(4):425–432, Apr. 1980.