Network Working Group J. Iyengar
Internet-Draft Franklin and Marshall College
Intended status: Informational B. Ford
Expires: January 7, 2010 Max Planck Institute for Software Systems
July 6, 2009

A Next Generation Transport Services Architecture
draft-iyengar-ford-tng-00.txt

Status of this Memo

Copyright Notice

Abstract

While there is substantial community interest in next-generation multipath-capable Internet transports, evolutionary pressures have gradually eroded the simplicity of the Internet's original transport architecture to a point where it is no longer realistically applicable to new tranports. This document proposes a new architectural framework for next-generation multipath-capable transport protocols, focusing immediately on multipath TCP but taking care to allow for generalization to other multipath-capable transports. The architecture places emphasis on enabling new multipath features in a safe, TCP-friendly, and backward-compatible fashion, retaining full interoperability with both existing applications and existing network infrastructure, and enabling reuse of existing protocols as much as possible while providing incremental deployment paths to new, more powerful and/or more efficient protocols. The architecture re-establishes the long-lost principles of end-to-end reliability and fate sharing, in the presence of existing and future network middleboxes, and enables the deployment of transport-neutral end-to-end protection without interfering with these policy-enforcing or performance-enhancing middleboxes. This document describes architecture goals, a layering model supporting these goals, abstract properties of the interfaces between the architecture's new layers, general approaches to multipath congestion control and how they fit into the architecture, realistic protocol design and incremental deployment paths, and ways in which this document complements and relates to ongoing protocol design activities in the IETF.

Table of Contents

1. Introduction

While there is substantial community interest in developing next-generation Internet transports supporting concurrent multipath communication and other new features, the Internet's evolution over the past decades have left us in a situation in which the Internet's traditional transport architecture is not well-suited to support new transports in general, or multipath transports in particular. This document attempts to address this problem by defining a new conceptual framework for Internet transports, which is both compatible with the pragmatic reality of the Internet today and the backward compatibility constraints new transports must face, while effectively enabling the deployment of compelling new features and the further long-term evolution of Internet transports.

The Internet's transport layer traditionally combines many functions, whose combination in one layer has led to numerous pragmatic challenges to transport evolution. Concurrent multipath transport requires decoupling congestion control state from application-visible transport instances and associating one transport instance with multiple per-path congestion control contexts [iyengar06concurrent]. Avoiding an explosion in congestion control contexts in the presence of applications like HTTP [RFC2616], which frequently use multiple transport instances in parallel, similarly requires decoupling and sharing congestion control contexts among transport instances [RFC2140] [RFC3124].

Some transport functions have pragmatically proven to be of concern to network operators and the policy-enforcing and performance-enhancing devices they now frequently deploy within the network, despite the the transport's original role as a purely "end-to-end" layer. For example, already-ubiquitous firewalls [RFC2979], NATs [RFC3022], and flow-aware routers [roberts03next] need access to transport layer port numbers to enforce application-sensitive access and traffic management policies, and performance enhancing proxies (PEPs) [RFC3135] need to interact with the transport's congestion control mechanisms [RFC2581] to optimize communication performance across diverse network technologies such as high-speed [RFC3649] and wireless [balakrishnan97comparison] links. Middlebox traversal challenges have rendered the traditional Network/Transport layer interface effectively useless for deploying new transports [I-D.rosenberg-internet-waist-hourglass], and the TCP header's limited 40-byte option space is almost consumed by options now considered "essential", such as SACK [RFC2018], leaving little leeway for further extending TCP in-place. Deploying end-to-end security is problematic because IP-level mechanisms such as IPsec [RFC4301] and HIP [RFC4423] interfere with middleboxes [RFC3947] [RFC5207], and transport-level mechanisms such as TLS [RFC5246] must be modified for

each transport [RFC4347] [RFC5238].

All of these challenges point to a pressing need to decompose the Internet's traditional monolithic transport layer; this document proposes such a decomposition as a guide for current and future transport evolution efforts.

The proposed architecture, which we refer to as Tng ("Transport next-generation"), decomposes traditional transport functions into four layers, described in detail in Section 3. Tng factors out endpoint-related functions into a separate Endpoint Layer, and performance-related transport layer functions such as congestion-control into a Flow Layer, leaving end-to-end semantic functions such as reliability and ordering in a Semantic Layer, and end-to-end security functions to an optional Security/Identity Layer.

The architecture described here is based on ideas proposed earlier [ford08breaking] [iyengar09flow], and builds on previous research in concurrent multipath transfer [iyengar06concurrent], congestion control state sharing [RFC2140] [RFC3124], and transport efficiency [RFC1644] [ford07structured]. We argue that addition of new non-trivial functionality to existing transport protocols must recognize, at least implicitly, the existence of these distinct components within the transport layer; we illustrate this point in Section 6 by juxtaposing the multipath TCP work currently underway with the Tng model. Our intent is for this architecture to define a framework into which specific protocols can fit in a modular fashion; the architecture further permits existing transport protocols to be reused and combined in new ways with only minor modifications to support a wide variety of important deployment scenarios.

The rest of this draft is organized as follows: Section Section 2 outlines the most important architectural goals for a next-generation transport architecture, with a particular emphasis on multipath transport; Section Section 3 describes our Tng architecture with specific details on how this architecture cleanly satisfies the goals identified above; Section Section 4 discusses the applicability of multipath congestion control work and its place in Tng; Section Section 5 identifies specific protocol stack designs that are compelling instantiations of Tng, with discussions about incremental deployability; and finally, Section Section 6 discusses Tng's relevance and potential contributions to current projects in and outside the IETF.

2. Architecture Goals

This section outlines what we perceive to be the most important goals

for a next-generation transport architecture, and for multipath-capable transport protocols conforming to such an architecture. We divide these goals into two categories: functional goals and performance/efficiency goals. We acknowledge that these goals and their categorization is subjective and invite debate on what the true goals and priorities should be.

2.1. Functional Goals

- Multihoming: The transport architecture must allow for a logical transport endpoint as seen by the application to correspond to multiple physical network attachment points, such as multiple IP addresses on the same or different network interfaces.

- Application Compatibility: The architecture must enable next-generation equivalents of existing transports, such as concurrent multipath versions of TCP, SCTP, or DCCP, to serve as "drop-in" replacements for the corresponding existing transports transparently to existing applications using those transports, merely by upgrading the operating systems of the end hosts.

- Network Compatibility: The architecture must enable next-generation transport instances to remain backward compatible with the Internet as it exists today, including being able to traverse predominant existing middleboxes such as firewalls, NATs, and performance enhancing proxies.

- Automatic Negotiation: A host supporting a next-generation equivalent of an existing transport must be able to detect reliably whether a new communication partner supports the next-generation protocol, using it if so, and otherwise automatically falling back to the existing protocol (e.g., standard TCP, SCTP, or DCCP).

- Naming/Addressing Neutrality: The transport architecture should cleanly abstract over specific naming/addressing schemes for hosts, endpoints, and services, with a goal of permitting transports conforming to this architecture to support multiple endpoint address formats (e.g., IPv4 and IPv6), cryptographic endpoint identities [RFC4423] [ford06persistent], and/or name-based endpoint identification [cheriton00triad]

- Coexistence with End-to-End Security: The architecture must allow for the optional deployment of end-to-end authentication and/or privacy protection in a transport-neutral but network-compatible fashion: i.e., supporting any transport conforming to this architecture without requiring modifications specific to each transport, while maintaining compatibility with legacy

middleboxes.

2.2. Performance/Efficiency Goals

- Multihoming and Multipath Capable: The architecture must enable a next-generation transport to detect and utilize multiple available paths between two logical endpoints, either one path at a time (fail-over multipath) or several at once (concurrent multipath).

- Resource Pooling: Transports should be able to balance traffic among available paths, optimizing network utility in a global sense by shifting load away from congested bottlenecks and taking advantage of spare capacity wherever it may be located [wischik08resource].

- Congestion State Sharing: Since popular applications such as HTTP often use multiple transport instances between the same pair of hosts, the architecture must avoid multiplicative explosions in multipath congestion control contexts - i.e., N transport instances times M multipath flows each - by enabling a multipath "bundle" of congestion control contexts to be shared cleanly among application-visible transport instances.

- TCP Friendliness: The architecture must enable new multipath transport flows to coexist gracefully with predominant existing transport flows, competing for bandwidth neither unduly aggressively or unduly timidly (unless low-precedence operation is specifically requested by the application, such as with LEDBAT [I-D.shalunov-ledbat-congestion]).

- Support Small Transactions: Recognizing that many applications today make heavy use of frequent small communications, such as HTTP conditional GET transactions or streaming media frames, next-generation transports should minimize the performance costs of supporting these common application behaviors, including by minimizing unnecessary protocol overhead on small packets and by unnecessary round-trip delays or state maintenance costs when applications use short transactions.

- Simplicity: The architecture should enable implementations of next-generation transports to be as simple as possible while remaining consistent with the above goals. In particular, end host implementations that never need certain transport features (e.g., embedded hosts) should ideally not have to incur the costs of implementing and validating those features.

3. Architecture Overview

This section presents an overview of one possible transport architecture that we believe can effectively support the above goals and provides a clean framework for next-generation multipath-capable transport protocols. We refer to this architecture for now simply as Tng, for "Transport next-generation". Tng as described here is based on ideas we proposed earlier [ford08breaking] [iyengar09flow]. While by no means the only possible architecture supporting multipath transport, our proposal incorporates many lessons learned from previous transport research and development practice, should fit well with the protocol design and congestion control work already in progress in the SCTP and multipath TCP communities, and we believe offers a strong starting point from which to develop a long-term architecture for next-generation Internet transports.

Tng is based on a decomposition of the Internet's traditional Transport Layer into up to four new layers, as illustrated below in Figure 1:

```
                                   +---------------+
                                   |  Application  |  ^
                               --> +---------------+  |
        +---------------+     /    |   Semantic    |  |
        |   Application |    /     +---------------+  | Application-
        +---------------+ <--      |   Isolation   |  | Oriented
        |   Transport   |          +---------------+  --------------
        +---------------+ <--      |     Flow      |  | Network-
        |    Network    |    \     +---------------+  | Oriented
        +---------------+     \    |   Endpoint    |  |
                               --> +---------------+  |
                                   |    Network    |  v
                                   +---------------+

         Existing Layers              Tng Layers
```

Figure 1

We first summarize Tng's new layers briefly here, then further discuss their functions and interactions below.

- Semantic Layer: implements whatever communication abstractions are to be made available to applications, including providing the end-to-end reliability and ordering properties of abstractions like TCP's byte streams or SCTP's message-based multi-streams.

- Isolation Layer (optional): provides end-to-end protection of the application's communication and of the end-to-end reliability

mechanisms it builds on.

- Flow Regulation Layer or Flow Layer: implements congestion control and other performance management functions.

- Endpoint Layer: Implements endpoint/service identification functions (e.g., port numbers) that network operators and their middleboxes require to enforce network access and security policies.

We loosely classify Tng's layers into "application-oriented" and "network-oriented" layers, as shown in Figure 1. We describe the top two layers as "application-oriented" because their functions are driven primarily by concerns of supporting and protecting the application's end-to-end communication. We consider the bottom two layers "network-oriented" because they represent functions that, while traditionally located in the ostensibly "end-to-end" Transport Layer, have proven in practice to be of great concern to network operators and the middleboxes they deploy in the network to enforce network usage policies [RFC3022] [RFC2979] or optimize communication performance [RFC3135].

Not all of Tng's functional layers need be present in the form described above in a particular instantiation of the architecture: e.g., the Isolation Layer may be omitted whenever strong end-to-end authentication or encryption are not needed. Similarly, layers may be recombined when pragmatic considerations require: e.g., we discuss later in Section Section 5 how legacy TCP connections may serve as one combined implementation of the Endpoint and Flow Layers for maximum compatibility with legacy networks and middleboxes. The layering diagram above is intended to serve as a conceptual model that helps appropriately place new functionality in the transport suite, and not a protocol design straitjacket.

3.1. Addressing Operational Concerns

Tng's decomposition of traditional transport functions serves several pragmatically important purposes at once concerning the operation of next-generation transports on today's and tomorrow's Internet:

- Multihoming and Multipath Transfer: By separating congestion control state from the semantic state of application-visible transport instances such as reliable byte streams, one transport instance may be associated with multiple congestion control (CC) contexts representing individual physical network paths to implement concurrent multipath transfer cleanly, as illustrated in Figure 2 below. The congestion control contexts for different paths may still interact, for example to achieve global resource

pooling as discussed later in Section Section 4, but the ability to maintain per-path congestion control state is a basic prerequisite for concurrent multipath operation on the Internet [iyengar06concurrent].

```
                          +-----------------------+
   Application            |    HTTP, SMTP, etc.   |
                          +-----------+-----------+
                                      |
                          +-----------------------+
   Semantic               |      Byte Stream      |
                          +-----------+-----------+
                                      |
                  Path 1      Path 2  |   Path 3      Path 4
                    /-----------/-----+-----\-----------\
                    |           |           |           |
                +------+    +------+    +------+    +------+
                | Per- |    | Per- |    | Per- |    | Per- |
   Flow         | Path |    | Path |    | Path |    | Path |
                |  CC  |    |  CC  |    |  CC  |    |  CC  |
                +------+    +------+    +------+    +------+
                    |           |           |           |
                +------+    +------+    +------+    +------+
   Endpoint     | Port |    | Port |    | Port |    | Port |
                +------+    +------+    +------+    +------+
                    |           |           |           |
                +------------------------------------------+
   Network      |               IP Forwarding              |
                +------------------------------------------+
```

Figure 2

- o Congestion State Sharing: Separating congestion control state from semantic state further enables multiple application-visible transport instances to share a single underlying congestion control context, or a single multipath bundle of congestion control contexts, as illustrated in Figure 3 below. Compelling even in the absence of multipath transport [RFC2140] [RFC3124] [ford07structured], congestion state sharing becomes essential in the presence of multipath transfer, to avoid an otherwise N x M explosion in congestion control contexts resulting from applications like HTTP that use N concurrent transport instances, each using M communication paths.

```
                      +------------------------------------------+
         Application  |                    HTTP                  |
                      +----------+---------------------+---------+
                                 |                     |
                          +------+------+       +------+------+
         Semantic         |   Stream 1  |       |   Stream 2  |
                          +------+------+       +------+------+
                                 |                     |
                                 \----------+----------/
                                            |
                        Path 1      Path 2  |   Path 3      Path 4
                          /-----------/-----+-----\-----------\
                          |           |           |           |
                      +------+    +------+    +------+    +------+
                      | Per- |    | Per- |    | Per- |    | Per- |
         Flow         | Path |    | Path |    | Path |    | Path |
                      |  CC  |    |  CC  |    |  CC  |    |  CC  |
                      +------+    +------+    +------+    +------+
                          |           |           |           |
                      +------+    +------+    +------+    +------+
         Endpoint     | Port |    | Port |    | Port |    | Port |
                      +------+    +------+    +------+    +------+
                          |           |           |           |
                      +------------------------------------------+
         Network      |               IP Forwarding              |
                      +------------------------------------------+
```

Figure 3

- Middlebox Compatibility: Decomposing the "network-oriented" functions of endpoint identification and congestion control enables middleboxes to enforce network policies and optimize performance without interfering with end-to-end reliability [saltzer84endtoend], fate sharing [clark88design], or end-to-end security [RFC3135]. In particular, the architecture allows middleboxes to interpose on the lower network-oriented layers, as illustrated below in Figure 4, without having to understand or interfere with the end-to-end application-oriented layers, yielding a variety of practical benefits [ford08breaking]. Through suitable reuse of existing protocols, a Tng protocol stack can even remain fully compatible with existing middleboxes, as described below in Section 5.

```
+-------------+                                  +-------------+
| Application |<------------ end-to-end ------------->| Application |
+-------------+                                  +-------------+
|  Semantic   |<------------ end-to-end ------------->|  Semantic   |
+-------------+                                  +-------------+
|  Isolation  |<------------ end-to-end ------------->|  Isolation  |
+-------------+                  +-------------+   +-------------+
|    Flow     |<----------------->|    Flow     |<->|    Flow     |
+-------------+   +-------------+   +-------------+   +-------------+
|  Endpoint   |<->|  Endpoint   |<->|  Endpoint   |<->|  Endpoint   |
+-------------+   +-------------+   +-------------+   +-------------+
|   Network   |<->|   Network   |<->|   Network   |<->|   Network   |
+-------------+   +-------------+   +-------------+   +-------------+
                     Firewall        Performance
   End Host           or NAT       Enhancing Proxy      End Host
```

Figure 4

- End-to-End Security: By locating the Isolation Layer precisely on the boundary between "network-oriented" and "application-oriented" functions, in a position effectively corresponding to somewhere in the middle of the classical Transport Layer, end-to-end security mechanisms at this position avoid interfering with network middleboxes as described above, while retaining IPsec's benefit of operating in a "transport neutral" fashion and protecting all application communication regardless of transport, without having to be modified for each new transport as application-level solutions such as TLS/DTLS does [RFC5246] [RFC4347].

Tng's contribution as an architecture is not to find a perfect or complete decomposition of the transport layer, but to identify specific transport functions that have proven in practice to be "network-oriented" contrary to their traditional placement in the transport layer, and to construct a new but incrementally deployable layering that reflects this reality and restores the "end-to-endness" of the remaining application-oriented functions.

The following subsections outline each Tng layer in more detail, including a brief summary of the interface each layer provides to higher layers.

3.2. Endpoint Layer

As in the OSI model, TCP/IP traditionally breaks application endpoint identifiers into Network Layer (IP address) and Transport Layer (port number) components, including only the former in the IP header on the assumption that the network need know only how to route to a given host, and leaving port numbers to be parsed and demultiplexed by the

transport. As the Internet's size and diversity exploded, however, network operators needed to enforce access policies that depend on exactly who is communicating--not just which hosts, but which applications and users. Now-ubiquitous middleboxes such as Firewalls, traffic shapers, and NATs must therefore understand transport headers in order to enforce these network policies. Since middleboxes cannot forward traffic for transports whose headers they do not understand, new transports have become effectively undeployable other than atop TCP or UDP [I-D.rosenberg-internet-waist-hourglass].

Recognizing that communicating rich endpoint information is a network-oriented function relevant to in-network policy enforcement, Tng factors this function into its Endpoint Layer so that middleboxes can extract this information without having to understand application-oriented headers. Tng reinterprets UDP as an initial Endpoint Layer protocol already supported by most middleboxes, but TCP can be used as well. In the longer term, we envision the Endpoint Layer evolving in a backward-compatible fashion to incorporate ideas from NAT traversal mechanisms [ford05p2p] [RFC5389] [I-D.ietf-mmusic-ice] and delegation-friendly architectures [walfish04middleboxes] [guha07end], thereby incorporating these mechanisms as required into the protocol stack and eventually eliminating the need for applications to manage connectivity challenges like these.

The Endpoint Layer's interface to higher layers is almost identical to the Network Layer's interface - it provides a simple best-effort packet delivery service - but with richer endpoint names than IP addresses alone provide. These endpoint names will initially consist simply of (IP address, port number) pairs for compatibility with TCP and UDP, but may later evolve in a backward-compatible fashion to include extensions such as service/port names [I-D.touch-tcp-portnames].

3.3. The Flow Regulation Layer

As Tng's Endpoint Layer factors out endpoint identification, the Flow Regulation Layer similarly factors out performance related functions such as congestion control, with the recognition that these functions have likewise become "network-oriented" in practice. The Flow Layer assumes that the underlying Endpoint Layer provides only best-effort packet delivery between application endpoints, and builds a flow-regulated best-effort delivery service for higher layers to build on. In particular, the Flow Layer's interface to higher layers includes an explicit signal indicating when the higher layer may transmit new packets.

To perform this flow regulation, the Flow Layer may either implement standard TCP-like congestion control, or, as we discuss in [iyengar09flow], may use more specific knowledge of an underlying network technology or administrative domain. In order to support multipath transports effectively, the Flow Layer also needs to enable higher layers to indicate sets of flows or "bundles" representing a single logical communication activity, and adjust the Flow Layer's congestion control algorithms to perform resource pooling as discussed later in Section 4. Tng's flow layer might eventually incorporate other performance-enhancing mechanisms as well, such as forward error correction.

3.4. The Isolation Layer

Having factored out network-oriented transport functions into the Endpoint and Flow Layers, the optional Isolation Layer "isolates" the application from the network, and protects the "end-to-endness" of higher layers. This isolation includes two elements. First, the Isolation Layer protects the application's end-to-end communication from interference or eavesdropping within the path, via transport-neutral cryptographic security as in IPsec. Second, the Isolation Layer protects the application and end-to-end transport from unnecessary exposure to details of network topology and attachment points, by implementing location-independent endpoint identities as in HIP [RFC4423] or UIA [ford06persistent], which remain stable even as devices move or the network reconfigures.

The Isolation Layer's interface to higher layers is functionally equivalent to the interface exported by the Flow Layer, but with transformed packet payloads and/or endpoint identities. We believe the Isolation Layer represents a suitable location for end-to-end security precisely because it defines the boundary between network-oriented and application-oriented functions, thus ensuring integrity and security of the latter, while allowing middleboxes to interact with the former. In contrast with SSL/TLS, the Isolation layer is neutral to transport semantics and does not need to be adapted to each transport. In contrast with IPsec's standard location immediately above IP, the Isolation Layer does give up the ability to protect Endpoint and Flow Layer mechanisms from off-path DoS attacks as IPsec protects TCP's signaling mechanisms, but if standard non-cryptographic defenses against such attacks are deemed insufficient, then IPsec authentication can still be deployed in Tng underneath the flow layer, ideally via a delegation-friendly scheme permitting controlled interposition by middleboxes.

3.5. The Semantic Layer

Tng's Semantic Layer implements the remaining application-oriented end-to-end transport functions, particularly end-to-end reliability. In the case of TCP, these functions are all those in the original TCP protocol [RFC0793] except port numbers, including acknowledgment and retransmission, order preservation, and receive window management. Other application-visible semantics, such as RDP's reliable datagrams and SCTP's message-based multistreaming [RFC4960], could fit equally well into Tng's Semantic Layer as distinct protocols.

The Semantic Layer's interface to lower layers differs from that of traditional Internet transports in two ways. First, a Tng semantic protocol uses the Endpoint Layer's endpoint identities (possibly transformed by the Isolation Layer) instead of implementing its own port number demultiplexing. Second, a Tng semantic protocol implements no congestion control but relies on the underlying Flow Layer to signal when packets may be transmitted. The Semantic Layer's interface to higher layers (e.g., the application) depends on the transport semantics it implements, but need not differ in any application-visible way from existing transport APIs--a fact that could aid deployment as discussed later in Section 5.

4. Multipath Congestion Control

Tng's decomposition of the Flow Layer from the Semantic Layer provides a natural decoupling for per-path congestion state in the Flow Layer from the per-transport-instance state in the Semantic Layer, allowing for a one-to-one, one-to-many, many-to-one, or many-to-many relationship between flows and transport instances in support of efficient multipath transport. An important outstanding issue, however, is how and whether multipath operation should affect the congestion control algorithm implemented in the Flow Layer. In particular, when a host is transmitting packets from one logical transport instance over multiple flows concurrently, how should the host adjust its congestion control behavior for each path? We briefly summarize here a few alternative high-level approaches, while making no attempt to explore the details or identify the "right" approach:

- No Adjustment: Each flow independently runs a standard, unmodified TCP congestion control algorithm [RFC2581]. This approach is simple and inherits all the well-understood properties of TCP congestion control, but creates fairness concerns: if M flows happen to share a congestion bottleneck, such as a home broadband link, then the multipath aggregate competes at that bottleneck M times as aggressively as a single-path TCP connection. The

effects of this approach are essentially the same - no better but also no worse - as the effects that BitTorrent clients cause as they utilize many parallel streams at application level.

- Static Weight Adjustments: Each flow independently runs standard TCP congestion control, but with each flow's aggressiveness lowered equally using known techniques [crowcroft98differentiated] so that the multipath aggregate as a whole exhibits the same aggressiveness as a single traditional TCP stream (or perhaps equivalent to two traditional TCP streams, if we take HTTP's allowance for two parallel streams [RFC2616] as a "de facto" fairness benchmark). This approach addresses the above fairness issue, at the cost of potentially underutilizing network capacity when some paths are more congested than others.

- Dynamic Weight Adjustments: As above, but dynamically adjust the relative weights of each flow to take greater advantage of lightly loaded paths while maintaining aggregate fairness [honda09multipath]. Such an approach can provide both fairness and resource pooling [wischik08resource], but it is not clear that these adjustments will converge to stability as many multipath aggregates dynamically shifting their flows' weights in response to each others' effects on the network.

- New Response Functions: The basic congestion control response function can be modified in a multipath-sensitive fashion so that the flows comprising all multipath aggregates provably converge toward stable and fair use of the network [kelly05stability]. As with most new congestion control schemes that modify the basic congestion response function, however, it is unclear how such a new scheme will behave in competition with existing TCP flows in the existing Internet, especially in the presence of connections with varying round-trip times.

Addressing the question of the right approach to adjusting congestion control for multipath operation will require further research and experimentation, which we consider to be independent of and orthogonal to Tng as an architecture. We assume that we will develop multipath transport protocols in parallel with this continuing research in multipath congestion control, and that multipath transports will for the time being need to support multiple alternative multipath congestion control schemes, in the same way that mature single-path transport implementations such as Linux's already support a variety of selectable single-path schemes.

To support such a "plug-and-play" selection of multipath congestion control schemes, Tng's Flow Layer will need one new architectural feature: a way for higher layers to indicate which flows should be

logically "tied together" and treated as a single logical aggregate for congestion control purposes. For this purpose, the interface that Tng's Flow Layer provides to higher layers provides an operation allowing higher layers to tie multiple flows into a "flow bundle". If we consider this operation to be a function, e.g., flow_bundle(flow1,flow2), and a Semantic Layer protocol opens three flows A, B, and C, to support a particular logical communication session, then the Semantic Layer protocol might call flow_bundle(A,B) followed by flow_bundle(A,C) to tie the three flows together into a bundle. Exactly what effect this action would have depends of course on the specific congestion control algorithm in effect: with the "No Adjustment" approach above, for example, tying together flows will have no effect on congestion control behavior; with the "Static Weight Adjustments" approach, the flow_bundle() calls will cause the Flow Layer to recompute the weights of all the newly-bundled flows so that the bundle as a whole exhibits the same effective aggressiveness as that of one (or two) single-path TCP connections. The Semantic Layer does not generally need to know exactly how these flow_bundle() calls are affecting congestion control, since it merely depends on the Flow Layer to indicate when each individual flow is ready to accept new packets for transmission.

While the Tng Flow Layer's flow_bundle operation is primarily intended for use by a multipath-capable Semantic Layer, there is no fundamental reason this operation could not or should not be exported further upwards, all the way to the application. Given a sockets API extension providing access to this operation, for example, an HTTP client could use it to indicate that the multiple concurrent HTTP streams it opens are logically part of the same application-level communication session, mitigating the fairness concerns that HTTP clients currently cause if they open more than one or two concurrent HTTP connections at once. A flow_bundle operation could probably be considered "safe" to export to untrusted applications since bundling can only cause flows to become less aggressive, not more. Nevertheless, exporting a flow bundling operation to applications is outside of Tng's immediate scope and purposes, so we leave it for future exploration.

5. Protocol Implementation and Incremental Deployment Paths

Any new Internet transport, or even any refactoring of an existing Internet transport, faces major deployment hurdles due to the Internet's inertia, and Tng is no exception. However, we find several reasons for optimism that an architecture incorporating the principles described here could overcome these deployment hurdles. This section therefore outlines potential approaches to overcoming these hurdles and facilitating deployment of next-generation

multipath-capable transports.

There are many possible protocol stack designs that would be architecturally consistent with Tng and its goals; we make no attempt to specify such a protocol stack design here, but leave that to be decided and standardized by the Internet transport community. Here we merely point out certain compelling possibilities that may be particularly worthy of further exploration, a few of which are illustrated in Figure 1 and discussed below:

```
                   +------+     +-------+     +-------+     +-------+
      Semantic     | TCP+ |     | SCTP+ |     | DCCP+ |     |  ???  |
                   +------+     +-------+     +-------+     +-------+
                        \------------\-----+------/-------------/
                                           |
                       /------------/------+------\-------------\
                   +-------+    +---------+    +-------+    +-------+
      Isolation    | DTLS  |    |  IPsec  |    |  TLS  |    |  ???  |
                   +-------+    +---------+    +-------+    +-------+
                       |             |             |            |
                   +---------------------+     +-------+    +-------+
      Flow         |        DCCP         |     |       |    |  ???  |
                   +---------------------+     |       |    +-------+
                              |                |  TCP  |        |
                   +---------------------+     |       |    +-------+
      Endpoint     |         UDP         |     |       |    |  ???  |
                   +---------------------+     +-------+    +-------+
                              |                    |            |
                   +------------------------------------------------+
      Network      |                       IP                       |
                   +------------------------------------------------+
```

Figure 5

- o Transport Reuse in the Semantic Layer: Any existing Internet transport may be adapted into a Tng Semantic Layer protocol merely by relocating it atop suitable Isolation (optional), Flow, and Endpoint Layers, and disabling the legacy transport's built-in congestion control, if any, in favor of using the Flow Layer's congestion control facilities. Figure 1 above illustrates TCP and SCTP protocols modified in this way, which we refer to as "TCP+" and "SCTP+" respectively to emphasize that they are identical in application-visible semantics but not identical in implementation to the corresponding original protocols. The "DCCP+" protocol illustrated in the figure provides congestion controlled datagram delivery like DCCP does, hence its name, but in fact needs to do little other than provide an application interface to the lower

layers, since the Flow Layer implements congestion control. This "DCCP+" protocol is thus likely to be closest to DCCP in application-visible function, but closest to UDP in implementation.

- Application Transparency: A Semantic Layer protocol in Tng can offer applications an API similar or identical to that of any legacy Internet transport. Our current Tng prototype already includes a Semantic Layer protocol providing a reliable stream abstraction compatible with TCP's, although it currently operates in user space and does not implement some TCP features such as urgent data. With careful design, a system-level implementation of a Tng protocol stack could replace TCP or another legacy transport completely transparently to applications: a host initiating a connection would dynamically probe the remote host for Tng support, use the new protocol stack if possible transparently to the application, and fall back on standard TCP (again transparently to the application) otherwise.

- Transport Reuse in Lower Layers: A protocol stack fully conforming to the Tng architectural model could be composed entirely of existing protocols "top-to-bottom": e.g., TCP with congestion control disabled as the Semantic Layer as described above, either DTLS or IPsec as the Isolation Layer, DCCP as the Flow Layer, and UDP as the Endpoint Layer (see the left two columns of Figure 1.) This approach may not necessarily yield the most far-reaching benefits without allowing some modifications to the original protocols, and may incur overheads due to redundancies between layers. Nevertheless, reuse could mitigate the difficulty of new protocol development and standardization.

- Compatibility with Existing Firewalls, NATs, and PEPs: While a DCCP-like protocol is most suited to Tng's Flow Layer, a Tng stack use standard TCP as a fallback "Flow Layer," atop which the Tng stack's true Isolation and Semantic Layer protocols would run as if a TCP "application (see the TLS/TCP column in Figure 1). While TCP's overhead and ordering constraints may incur a performance cost, encapsulation in legacy TCP flows would make the new stack even more compatible with existing networks and capable of benefiting from existing TCP-based PEPs, and could still restore end-to-end fate-sharing by ensuring that the new Semantic Layer retains all end-to-end "hard state" and can restart failed Flow Layer TCP flows.

- Single-Ended Operation: If only one end host supports the new Tng protocol stack, maintaining compatibility with legacy TCP hosts constrains the other host to use legacy TCP on the wire as well, making it impossible for the on-the-wire format to be decomposed

according to the Tng layering model. On the other hand, even in this case, the Tng layering model can still help guide and conceptually organize protocol stack implementations in the end hosts. For example, the single-ended multipath TCP protocol currently under development [I-D.van-beijnum-1e-mp-tcp], while avoiding any changes to TCP's wire protocol for compatibility reasons, still requires a clean internal separation between per-path congestion control functions and TCP semantic functions such as reliability and ordering, as illustrated in Figure 2. Organizing next-generation transports according to this model, regardless of whether the wire protocol matches the model or not, may simplify the design and implementation of end hosts that wish to support both single-ended and double-ended multipath operation, for example.

6. How Tng Complements Current Projects

Two drafts are being considered in the IETF as multipath transport solutions: Single-Ended MPTCP [I-D.van-beijnum-1e-mp-tcp], which is a sender-only based multipath TCP, and Two-Ended MPTCP [I-D.ford-mptcp-multiaddressed], which requires receiver participation as well. Having discussed how Tng applies to the One-Ended proposal in Section Section 5, we now focus on how Tng applies to the Two-Ended MPTCP proposal.

Two-Ended MPTCP, functionally, divides the Transport Layer into two components: the MPTCP part, which is responsible for global ordering of application data and for reliability; and the "legacy TCP" part, which is essentially responsible for congestion control on each path. MPTCP suggests adding new paths by creating separate connections and adding them into the MPTCP pool and also acknowledges the possibility that new connections may be established from a sender to a multihomed receiver on different ports, since the MPTCP connection is ultimately identified by the MPTCP Sender Token. For all practical purposes, the upper MPTCP component maps to our Semantic Layer with the lower TCP implementing our Flow and Endpoint Layers. We juxtapose the Two-Ended MPTCP architecture from [I-D.ford-mptcp-multiaddressed] with Tng's architecture below:

```
        +---------------------------+     +--------------------------+
        |       Application         |     |      Application         |
        +---------------------------+     +--------------------------+
        |         MPTCP             |     |        Semantic          |
        + - - - - - -  +  - - - - - - +   +--------------------------+
        |    TCP      |      TCP      |     | Flow+Endp  |  Flow+Endp  |
        +---------------------------+     +--------------------------+
        |     IP      |      IP       |     |     IP     |      IP     |
        +---------------------------+     +--------------------------+
```

Figure 6

There are significant functional and structural similarities between the Tng architecture and MPTCP, and it is not by accident. We argue that adding any new non-trivial functionality into an existing transport must come to terms with this separation of concerns necessary within the transport layer.

While [I-D.ford-mptcp-multiaddressed] identifies the same components as we do for implementing a multipath transport protocol, Tng separates these components and places them in a clean framework in which other Semantic protocols (e.g., SCTP) or other Flow protocols (e.g., DCCP over UDP) can also fit and interoperate together. Our work is complementary to the current MPTCP effort, since it describes the architecturally clean space within which MPTCP can exist, and describes clear directions in which this architecture can evolve. Tng defines an architectural blueprint in which MPTCP is one instantiation, providing a TCP-compatible byte streams at the Semantic Layer and legacy TCP as a backward-compatible Flow Layer.

We now turn our attention to other multihoming- and multipath-capable transport proposals, showing in turn how their design, explicitly or implicitly, embodies Tng's architectural separation of concerns:

o SCTP [RFC4960] has sophisticated mechanisms for multihoming and for managing addition and deletion of endpoints within an end-to-end "association" [RFC5061]. The SCTP protocol design makes a clear distinction between the application-oriented ordering through streams and separate Stream Sequence Numbers (SSNs); this "upper" component maps to our Semantic Layer. SCTP uses a separate Transmission Sequence Number (TSN) space for the flow-level function of congestion control, which maps to our Flow Layer. SCTP does use flow-level TSNs for reliability; and some form of global per-flow sequencing information will be important to enable collusion among streams for loss recovery --- this information does not have to be explicit in packets, but can be signaling between the Semantic and Flow Layers as discussed in Section 4.

- LS-SCTP [al04ls-sctp] and pTCP [hsieh02ptcp] are examples of multipath transports that require both sender- and receiver-side modifications to SCTP and TCP, respectively. Both use a separate sequence space for global ordering while using a per-path sequence space for flow functions. Similar to Two-Ended MPTCP, these protocols can be divided into upper and lower parts that map to our Semantic and Flow Layers.

- CMT [iyengar06concurrent] and mTCP [zhang04transport] are examples of multipath transports that require modifications only to the sender-side in SCTP and TCP, respectively. Similar to One-Ended MPTCP, as discussed in Section Section 5, such protocols may still benefit from a cleaner conceptualization of functions within the transport suite.

- SST [ford07structured] is a new transport protocol that implements Tng's separation of concerns internally in addition to other new application-visible features. SST uses a separate Stream Protocol with stream sequence numbers for application-oriented ordering and reliability functions, a separate Channel Protocol with transmit sequence numbers for security and for network-oriented congestion control, and a separate Endpoint Layer implemented by UDP. Our first prototype of Tng builds on the SST implementation.

It should also be possible to implement Tng by adapting other established protocols as well. DCCP [RFC4340] or CM [RFC3124], could implement our Flow Layer; IPsec and/or HIP, modified to run atop the Flow Layer, could provide our Isolation Layer; and TCP, SCTP, or another existing transport, modified to disable its internal congestion control functions and instead use those of the underlying Flow Layer, could provide our Semantic Layer. There would be costs to this approach: the existing protocols were not designed for Tng, and they each independently re-implement considerable functionality; MPTCP is an integrated design that reuses or shares synergistically between layers. Thus, while incremental deployment is possible with Tng, one can build integrated designs for specific protocol combinations that are tighter and better optimized for bit-space.

Finally, we note that Tng's Endpoint Layer, being responsible for interaction with NATs, provides an architecturally clean space for NAT traversal work, such as STUN [RFC5389] and TURN [I-D.ietf-behave-turn], that are in progress in the IETF BEHAVE working group.

7. Experiences with Running Code

We have a working prototype implementation of a protocol stack

conforming to Tng's layering model and providing the basic structural benefits described above. The prototype and experiments with it are described in more detail elsewhere [iyengar09flow]. More to be written (perhaps).

8. Security Considerations

To be written.

9. IANA Considerations

To be written: endpoint ids, transport protocol numbers, ...

10. Acknowledgments

To be written. Discussions and participants on the multipathtcp mailing list.

11. References

11.1. Normative References

[RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, September 1981.

[RFC1644] Braden, B., "T/TCP -- TCP Extensions for Transactions Functional Specification", RFC 1644, July 1994.

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

[RFC2140] Touch, J., "TCP Control Block Interdependence", RFC 2140, April 1997.

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

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2979] Freed, N., "Behavior of and Requirements for Internet Firewalls", RFC 2979, October 2000.

[RFC3022] Srisuresh, P. and K. Egevang, "Traditional IP Network Address Translator (Traditional NAT)", RFC 3022, January 2001.

[RFC3124] Balakrishnan, H. and S. Seshan, "The Congestion Manager", RFC 3124, June 2001.

[RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, June 2001.

[RFC3649] Floyd, S., "HighSpeed TCP for Large Congestion Windows", RFC 3649, December 2003.

[RFC3947] Kivinen, T., Swander, B., Huttunen, A., and V. Volpe, "Negotiation of NAT-Traversal in the IKE", RFC 3947, January 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.

[RFC4347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", RFC 4347, April 2006.

[RFC4423] Moskowitz, R. and P. Nikander, "Host Identity Protocol (HIP) Architecture", RFC 4423, May 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5061] Stewart, R., Xie, Q., Tuexen, M., Maruyama, S., and M. Kozuka, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC 5061, September 2007.

[RFC5207] Stiemerling, M., Quittek, J., and L. Eggert, "NAT and Firewall Traversal Issues of Host Identity Protocol (HIP) Communication", RFC 5207, April 2008.

[RFC5238] Phelan, T., "Datagram Transport Layer Security (DTLS) over the Datagram Congestion Control Protocol (DCCP)", RFC 5238, May 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

11.2. Informative References

[I-D.ford-mptcp-multiaddressed]
Ford, A., Raiciu, C., Handley, M., and S. Barre, "TCP Extensions for Multipath Operation with Multiple Addresses", draft-ford-mptcp-multiaddressed-00 (work in progress), May 2009.

[I-D.ietf-behave-turn]
Rosenberg, J., Mahy, R., and P. Matthews, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", draft-ietf-behave-turn-16 (work in progress), July 2009.

[I-D.ietf-mmusic-ice]
Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", draft-ietf-mmusic-ice-19 (work in progress), October 2007.

[I-D.rosenberg-internet-waist-hourglass]
Rosenberg, J., "UDP and TCP as the New Waist of the Internet Hourglass", draft-rosenberg-internet-waist-hourglass-00 (work in progress), February 2008.

[I-D.shalunov-ledbat-congestion]
Shalunov, S., "Low Extra Delay Background Transport (LEDBAT)", draft-shalunov-ledbat-congestion-00 (work in progress), March 2009.

[I-D.touch-tcp-portnames]
Touch, J., "A TCP Option for Port Names", draft-touch-tcp-portnames-00 (work in progress), April 2006.

[I-D.van-beijnum-1e-mp-tcp]
Beijnum, I., "One-ended multipath TCP", draft-van-beijnum-1e-mp-tcp-00 (work in progress), May 2009.

[al04ls-sctp]
Al, A., Saadawi, T., and M. Lee, "LS-SCTP: A Bandwidth Aggregation Technique For Stream Control Transmission

Protocol", Computer Communications Vol. 27, No. 10, June 2004.

[balakrishnan97comparison]
Balakrishnan, H., Padmanabhan, V., Seshan, S., and R. Katz, "A Comparison of Mechanisms for Improving TCP Performance over Wireless Links", IEEE/ACM Transactions on Networking Vol. 5, No. 6, December 1997.

[cheriton00triad]
Cheriton, D. and M. Gritter, "TRIAD: A New Next-Generation {Internet} Architecture", Online at: http://www.dsg.stanford.edu/triad, July 2000.

[clark88design]
Clark, D., "The Design Philosophy of the DARPA Internet protocols", ACM SIGCOMM, August 1988.

[crowcroft98differentiated]
Crowcroft, J. and P. Oechslin, "Differentiated End-to-End Internet Services using a Weighted Proportional Fair Sharing TCP", ACM SIGCOMM Computer Communication Review Vol. 28, No. 3, July 1998.

[ford05p2p]
Ford, B., "Peer-to-Peer Communication Across Network Address Translators", USENIX Annual Technical Conference, April 2005.

[ford06persistent]
Ford, B., Strauss, J., Lesniewski-Laas, C., Rhea, S., Kaashoek, F., and R. Morris, "Persistent Personal Names for Globally Connected Mobile Devices", USENIX OSDI, November 2006.

[ford07structured]
Ford, B., "Structured Streams: a New Transport Abstraction", ACM SIGCOMM, August 2007.

[ford08breaking]
Ford, B. and J. Iyengar, "Breaking Up the Transport Logjam", ACM HotNets, October 2008.

[guha07end]
Guha, S. and P. Francis, "An End-Middle-End Approach to Connection Establishment", ACM SIGCOMM, August 2007.

[honda09multipath]

Honda, M., Nishida, Y., Eggert, L., Sarolahti, P., and H. Tokuda, "Multipath Congestion Control for Shared Bottleneck", International Workshop on Protocols for Future Large-Scale and Diverse Network Transports (PFLDnet), May 2009.

[hsieh02ptcp]
Hsieh, H-Y. and R. Sivakumar, "pTCP: An End-to-End Transport Layer Protocol for Striped Connections", IEEE ICNP, November 2002.

[iyengar06concurrent]
Iyengar, J., Amer, P., and R. Stewart, "Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Paths", IEEE/ACM Transactions on Networking Vol. 15, No. 5, October 2006.

[iyengar09flow]
Iyengar, J. and B. Ford, "Flow Splitting with Fate Sharing in a Next Generation Transport Services Architecture", Under submission, Online at http://www.fandm.edu/jiyengar/papers/flowsplitting.pdf, June 2009.

[kelly05stability]
Kelly, F. and T. Voice, "Stability of end-to-end algorithms for joint routing and rate control", ACM SIGCOMM Computer Communication Review Vol. 35, No. 2, April 2005.

[roberts03next]
Roberts, L., "The Next Generation of IP --- Flow Routing", International Conference on Advances in Infrastructure for Electronic Business, Science, Education, Medicine, and Mobile Technologies on the Internet, July 2003.

[saltzer84endtoend]
Saltzer, J., Reed, D., and D. Clark, "End-To-End Arguments in System Design", Transactions on Computer Systems Vol. 2, No. 4, November 1984.

[walfish04middleboxes]
Walfish, M. and et. al, "Middleboxes No Longer Considered Harmful", USENIX OSDI, December 2004.

[wischik08resource]
Wischik, D., Handley, M., and M. Bagnulo-Braun, "The Resource Pooling Principle", ACM SIGCOMM CCR Vol. 38, No.

5, October 2008.

[zhang04transport]
Zhang, M. and et. al, "", USENIX Annual Technical Conference, June 2004.

Authors' Addresses

Janardhan Iyengar
Franklin and Marshall College
Mathematics and Computer Science
PO Box 3003
Lancaster, PA 17604-3003
USA

Phone: 717-358-4774
Email: jiyengar@fandm.edu

Bryan Ford
Max Planck Institute for Software Systems
Saarbrucken,
Germany

Email: baford@mpi-sws.org