Towards More Expressive Transport-Layer Interfaces

Lars Eggert
NEC Europe Ltd.
Network Laboratories
Kurfürstenanlage 36
69115 Heidelberg
Germany
lars.eggert@netlab.nec.de

Wesley M. Eddy
Verizon Federal Network Systems
NASA Glenn Research Center
MS 54-5
21000 Brookpark Rd.
Cleveland, OH, 44135, USA
weddy@grc.nasa.gov

ABSTRACT

The individual layers in the Internet protocol stack provide communication abstractions that expose a limited set of operations and information and otherwise hide layer-internal and lower-layer complexities. This paper argues that the communication abstractions provided by the layers through these interfaces – especially the network/transport and transport/application layer interfaces – do not support efficient and performant communication in an increasingly dynamic Internet. This paper proposes to extend the established interfaces by exposing optional, generic and technology-independent information and operations that allow the development of layer-internal mechanisms to improve operation and performance of the Internet protocols while maintaining the layering abstraction.

1. INTRODUCTION

In the layered Internet model, transport protocols – at the "transport layer" – provide communication primitives that let applications exchange data. They also provide some support functions that let applications manage individual instances of communication primitives. Transport protocols provide their service on top of IP – the "network layer." Different transport protocols provide communication primitives with different characteristics. TCP [11], for example, provides a communication primitive that is a congestion- and flow-controlled, reliable, in-order byte stream. UDP [12] provides a different communication primitive that lets applications exchange individual messages in an unreliable, unordered fashion without flow or congestion control. DCCP [13] and SCTP [14] provide communication primitives with yet different characteristics.

An important feature of the layered structure of the Internet protocols is that each layer provides an abstract interface to its users, which are usually other ("higher") layers, including user applications. This interface typically offers a number of well-defined operations and exposes a set of well-defined information. It also hides other functionality present within a layer and below it. Consequently, a layer provides a generalized and idealized communication abstraction to its users, which often simplifies implementation of functionality on top of a layer.

MobiArch '06 San Francisco, California, USA

The communication primitives provided by the transport layer to applications are one example of such a communication abstraction. A second example is the interface that the network layer provides to the transport protocols. It offers functions to deliver packets in some order, without guaranteeing delivery or protecting against duplication or loss. It also offers functions that expose some well-defined pieces of network-layer information. At the same time, the communication abstraction provided by the network layer hides other complexities from its users, i.e., the transport protocols. These include, for example, the establishment and management of end-to-end routing paths, configuration of network-layer addresses, and fragmentation and reassembly of packets, among others.

The concept of layers that provide communication abstractions in the form of simplified and generalized "virtual machines" that abstract away layer-internal or underlying complexities is certainly useful. However, it can also introduce limitations, when the interface does not offer specific functions or expose certain pieces of information that would be of benefit to its users.

The interfaces defined between the layers in the Internet protocol stack currently have such limitations. Section 2 discusses how the assumptions that were made at the time these interfaces were designed no longer accurately reflect the realities of the current, much more dynamic Internet. It also analyzes how these limitations can cause operational problems and limit performance and efficiency. Section 3 makes an argument for richer, more expressive communication abstractions, i.e., interfaces between layers.

It is important to note that this paper argues that the interfaces between layers should provide additional pieces of generic, technology-independent information that let protocols be more efficient and effective, and less prone to behave incorrectly due to faulty assumptions. This paper does not argue for specific niche optimizations of particular transports over particular link layers, such as "TCP over 802.11b" or "SCTP over Bluetooth." Instead, it proposes to identify a set of network- and lower-layer information that is generic and can be provided in different ways by different specific underlying technologies.

2. CURRENT TRANSPORT-LAYER INTERFACES

The previous section has described how layers in the Internet protocol stack can be seen as "virtual machines" that expose generic communication abstractions to their users. This section discusses two of these interfaces – between the network and transport layers, and between the transport and application layers – in more detail. Specifically, it argues that some of the assumptions underlying the design of these interfaces no longer accurately reflect the network-

ing environment in the current Internet. This mismatch limits the performance and efficiency of the Internet protocols.

The communication abstraction provided by IP at the network layer delivers packets in an unordered, unreliable manner and does not protect against duplication. The users of this abstraction, i.e., the transport protocols, have made additional assumptions in the past about the abstraction provided by the network layer. Many of these assumptions are critical to the effective operation of important transport mechanisms, such as congestion control, flow control or reliability. These assumptions include, for example, that hosts remain at network locations identified by an IP address ("their" IP address) on timescales that are orders of magnitude larger than the duration of a communication instance. Another such assumption is that packets that flow from a source host to a destination host mostly follow the same path and that changes to that path occur on timescales that are orders of magnitude larger than the round-trip time (RTT) between the two hosts. Similarly, transport mechanisms have assumed that the characteristics of such paths, such as bandwidth, delay, reordering and loss probabilities, also change on timescales much larger than the RTT.

Note that these assumptions were valid for the Internet at the time that the interface between network and transport layer and the corresponding layer-internal functions were initially designed. However, in the current Internet, many of these assumptions are no longer universally true, and the rate at which such assumptions do not hold is increasing. This is, because the Internet has become much more dynamic in recent years. Mobile hosts and whole subnetworks have started to move between network locations on relatively short timescales. A growing number of hosts are multihomed, i.e., are present at multiple network locations at the same time, connected through links with possibly very different properties. The Internet has incorporated new link technologies with characteristics that are much more dynamic than in the past, due to functionality such as link-layer retransmissions, adaptive coding or support for link-local mobility. This trend is likely to continue and even accelerate in the future.

Several extensions to the internal functionality of the network layer, such as Mobile IP [19], NEMO [20], HIP [21] or SHIM6 [22], support efficient IP communication in such dynamic environments. They generally expose static tags to the transport layer, instead of directly exposing IP addresses, and internally manage the changes to IP addresses due to mobility or multihoming. This approach maintains the traditional interface between network and transport layers, isolating the transports from the dynamic effects present at and below the network layer, similar to how transports remain unaware of routing changes or packet fragmentation. This approach allows existing transport protocols to continue to operate without modifications.

This isolation, however, comes at a cost, because the traditional communication abstraction maintained by these new network-layer extensions hides information that transport-layer protocols should act on. As described above, many transport mechanisms, such as congestion window estimation [8], RTT measurements [9] or path MTU discovery [10], are not agile enough to properly handle the significant instantaneous path changes that these network-layer extensions introduce. This, in turn, can decrease the effectiveness of important transport mechanisms, such as congestion control. Consequently, although existing transports can operate on top of these network-layer extensions to some degree, their performance and efficiency decreases, as documented in several studies [15, 26, 27].

Before discussing ways to address these issues in Section 3, note that the interface between the transport and applications layers has similar limitations that cause similar problems. The most common operating systems today implement the "socket" API (derived from BSD UNIX) to provide applications with access to transport protocol services in much the same way as file I/O. This familiarity and convenience is one reason for the popularity of the socket API.

The similarity to file I/O, however, is also one weakness of the socket API, because it hides nearly all network-specific information, such as packet loss or flow control events, from the applications. Although some systems offer mechanisms to obtain such information, such as the `proc` file system on Linux, these interfaces are extremely platform-specific. One of the few standardized, cross-platform interfaces to transport- and network-layer information is through the Simple Network Management Protocol (SNMP) [16], which allows applications to access standardized sets of information about network- and transport-layer information, such as TCP statistics [7]. However, SNMP is not typically available on most hosts, and requiring its use for applications to acquire information about their own transport connections is an extremely cumbersome solution.

In summary, both the interface between the network and transport layers and the interface between the transport and application layers fail to expose sufficient information that would allow transport mechanisms and applications to operate efficiently in a dynamic Internet. This paper argues that refining the interactivity between the protocol layers – in a way that is independent of specific link-, network, and transport-layer extensions and technologies – can enable better performance and operation while maintaining the benefits that the layering abstraction provides.

3. MORE EXPRESSIVE TRANSPORT-LAYER INTERFACES

The simple, general purpose interface between the network and transport layers is one of the key features that has guaranteed the evolvability of the Internet architecture, because it maintains the independence of transport layers from functionality located below it, and vice versa. Approaches for extending this core component must therefore be broadly applicable and be of general usefulness. Point solutions that optimize for specific deployment scenarios or technologies are thus not relevant to this discussion.

The current interface between the network and transport layer encompasses a relatively small set of pieces of information that belong to one of two types. The transport layer derives the first type of "implicit" information from the observed behavior of the network layer. This implicit information includes, for example, the RTT, available bandwidth and path MTU to a destination. The network layer signals the second type of "explicit" information directly to the transport protocols. Explicit information includes, for example, host, network and protocol unreachable indications derived from ICMP [1]. The deprecated "source quench" ICMP message was another such piece of explicit information.

It is important to note that the idea of extending the interfaces between layers in the protocol stack is not new. For example, a large number of research papers have presented and analyzed TCP extensions that attempt to improve performance over specific link technologies, such as IEEE 802.11, satellites or cellular links, to name a few. Although the performance benefits can be substantial, many of these "cross-layer optimizations" have the significant drawback that they encode knowledge about the specifics of a single underlying link technology inside the transport protocol. In essence, they eliminate the common base that IP provides as a network-layer abstraction, and result in different flavors of TCP that are each optimized for particular link technologies. This can be seen as a "layering violation," because TCP now effectively relies

on knowledge of specific link layers, instead of a unifying network layer. In addition, similar modifications must be made to optimize other transport protocols over each link technology. Given the increasing number of transport protocols and the vastly increasing number of deployed link technologies, this approach quickly leads to a very complex system. Hence, such optimizations for limited scenarios are not appropriate for a general-purpose Internet.

This paper thus argues that the network-layer abstraction is important to maintain, because it establishes independence of transport layers from functionality located below the network layer. The current practice of keeping transport mechanisms mostly independent of the specific characteristics of different lower-layers is correct and should be retained. The goal is to identify a generic, technology-independent set of network- and lower-layer information that improves transport performance and operation and can be provided in different ways by different specific underlying technologies.

In addition, it is important that these additional pieces of information or operations should be optional, i.e., not required for the correct operation of transport protocols. This enables incremental, uncoordinated deployment of layers that provide the extended interface and layers that utilize it. For example, a traditional transport protocol implementation can operate on top of a network layer that provides some additional pieces of information without change in functionality or performance. Likewise, an improved transport protocol must be able to operate on top of a network layer that does not provide an enhanced set of lower-layer information.

In addition to these two requirements, the Internet Architecture Board (IAB) has described architectural issues and has provided examples of appropriate and inappropriate uses of inter-layer indications [6]. Although their discussion blurs the distinction between indications originating at the link and network layers to some degree, it still contains useful observations that solutions in this space should consider.

One example of an improvement that follows the approach described above is the proposed TCP response mechanism after receiving a "path characteristics have changed" signal from lower layers [5]. This signal has a well-defined meaning that represents the implication of several types of events at the network and lower layers. For example, it can be generated after Mobile IP or HIP mobility events, when link status changes [28], or even when route computation changes the path, in addition to other possibilities. The proposed TCP response mechanism does not differentiate which underlying mechanism generated the signal – the response is always the same and only depends on the state of the connection that receives such a signal. While transmitting, the signal causes a connection to slow-start restart with a freshly initialized path state; when disconnected, the signal triggers a speculative retransmission; see [5] for details on the mechanism and the interaction with trigger sources. Experiments with this TCP extension [15, 29] have shown that it significantly improves TCP performance and efficiency when operating over paths with intermittent connectivity.

Other proposals in the past have introduced extensions to the interface between the network and transport layer that are similarly independent of specific technologies. For example, Explicit Congestion Notification (ECN) [2] at the network layer and the corresponding response mechanisms for different transport protocols are similar in spirit. The ECN signal is well-defined and can be provided in different ways by lower layers; transport protocols act on the signal independently of who and how it was generated. ECN is an interesting example because it involves both inter-layer signaling on a local host (passing the "congestion experienced" information received in IP headers to the transport layer congestion control mechanisms) and transport-layer signaling between remote peers (relaying the "congestion experienced" information and conveying the response indication).

Another example of such an extension is Quick-Start [3], which is a less ambitious and more immediately practical version of the ideas first described by the Explicit Congestion Control Protocol (XCP) [4]. Here, routers in the network explicitly signal source hosts the available capacity along the path to their destinations. Transport protocols can utilize this generic, technology-independent, network-layer information in different ways to improve operation and performance. The Congestion Manager [17, 18] is a proposal to share explicit information about bandwidth and congestion between related transport layer entities and the applications they are bound to.

Similar to how giving transport protocols more information about current network layer conditions can improve their operation and performance, giving applications more information about the conditions their transport connections experience can also improve their operation and performance. For example, applications can implement application-layer framing [24] more effectively, if the transport path MTU estimates are available to them. Applications that implement internal timers can similarly benefit from information about transport path round-trip time estimates.

Extensions to the interface between transport and application layers that provide additional operations can be similarly useful. For example, UDP-Lite [25] enables applications to specify the number of payload bytes that the UDP-Lite checksum covers, in addition to an otherwise normal UDP send call. Applications that receive UDP-Lite payloads must explicitly state their willingness to receive potentially damaged payload data. It can be beneficial for some applications to be able to poll UDP-Lite for an estimate of the number of damaged datagrams that are caught and discarded based on the partial checksum, so that they can adapt their codecs even more intelligently.

These existing examples all demonstrate that more expressive interfaces between protocol layers are not a new idea, but rather an overarching common theme that can unify several independent efforts. Uniform extensions to these interfaces can greatly aid the development of protocol enhancements that require inter-layer information and can similarly enhance the deployability and usefulness of such approaches, compared to the scenario-dependent cross-layer techniques that are currently popular.

It is worth noting that although the idea of extending the interfaces between protocol layers in this way is simple, the difficulty lies in the details. The unsuccessful "triggers for transport" [23] and ALIAS efforts in the IETF bear witness to this. One key issue is security, especially when local protocol instances act on pieces of information provided by untrusted remote entities. Even with the restriction that such information be advisory in nature, this practice may create an attack vector that can be exploited to interfere with correct and efficient protocol operation, depending on the response mechanisms that are employed. Cryptographic authentication of remote information would be an ideal but unfortunately also infeasible solution, due to the well-known overheads of establishing and maintaining a security framework, which has so far been unsuccessful even for arguably more important uses. It may, however, be possible to mitigate potential attacks through careful design of the response mechanisms. Response mechanisms should be robust in the presence of misleading or malicious hints. Response mechanisms may correlate information from different sources before committing to an action, or actively attempt to verify the accuracy of the reported information. It may also be possible to establish trust in a source of information – and the accuracy of the

information of that source – based on an accumulated history of past behavior.

4. CONCLUSION

The individual layers in the Internet protocol stack provide communication abstractions that expose a limited set of operations and information, and otherwise hide layer-internal and lower-layer complexities. This paper has argued that the communication abstractions provided by the layers through these interfaces – especially the network/transport and transport/application layer interfaces – do not support efficient and performant communication in an increasingly dynamic Internet.

This paper has proposed to extend the established interfaces by exposing optional, generic and technology-independent information and operations that allow the development of layer-internal mechanisms to improve operation and performance of the Internet protocols while maintaining the layering abstractions.

Defining the exact technical means for achieving these improved interfaces is future work that the authors believe the community should pursue. These relatively minor extensions to the established stack architecture promises long-term benefits greater than the multitude of short-term point solutions that are frequently pursued today.

5. ACKNOWLEDGMENTS

The authors would like to thank Mark Allman, Jari Arkko, Aaron Falk, Simon Schütz, Yogesh Swami, Magnus Westerlund and the participants of the IETF-66 "Transport-Enhancing Refinements to the Network-Layer Interface (TERNLI) Bar BOF" for their thoughtful suggestions on the issues and approach described in this paper.

Lars Eggert is partly supported by Ambient Networks, a research project funded by the European Commission under its Sixth Framework Program.

6. REFERENCES

[1] Robert Braden (ed.) Requirements for Internet Hosts – Communication Layers. RFC 1122, October 1989.

[2] K. K. Ramakrishnan, Sally Floyd and David L. Black. The Addition of Explicit Congestion Notification (ECN) to IP. RFC 3168, September 2001.

[3] Amit Jain, Sally Floyd, Mark Allman and Pasi Sarolahti. Quick-Start for TCP and IP. Work in Progress (Internet-Draft draft-ietf-tsvwg-quickstart-06), August 2006.

[4] Dina Katabi, Mark Handley and Charlie Rohrs. Congestion Control for High Bandwidth-Delay Product Networks. Proc. ACM SIGCOMM, Pittsburgh, PA, USA, August 19-22, 2002.

[5] Simon Schütz, Lars Eggert, Wesley M. Eddy, Yogesh P. Swami and Khiem Le. TCP Response to Lower-Layer Connectivity-Change Indications. Work in Progress (Internet-Draft draft-schuetz-tcpm-tcp-rlci-00), May 2006.

[6] Bernard Aboba (ed.) Architectural Implications of Link Indications. Work in Progress (Internet-Draft draft-iab-link-indications-05), July 2006.

[7] Matt Mathis, John Heffner, and Rajiv Raghunarayan. TCP Extended Statistics MIB. Work in Progress (Internet-Draft draft-ietf-tsvwg-tcp-mib-extension-11), August 2006.

[8] Mark Allman, Vern Paxson and W. Richard Stevens. TCP Congestion Control. RFC 2581, April 1999.

[9] Vern Paxson and Mark Allman. Computing TCP's Retransmission Timer. RFC 2988, November 2000.

[10] Jeffrey Mogul and Steve Deering. Path MTU Discovery. RFC 1191, November 1990.

[11] Jon Postel (ed.) Transmission Control Protocol. RFC 793, September 1981.

[12] Jon Postel (ed.) User Datagram Protocol. RFC 768, August 1980.

[13] Eddie Kohler, Mark Handley and Sally Floyd. Datagram Congestion Control Protocol (DCCP). RFC 4340, March 2006.

[14] Randall R. Stewart, Qiaobing Xie, Ken Morneault, Chip Sharp, Hanns Jrgen Schwarzbauer, Tom Taylor, Ian Rytina, Malleswar Kalla, Lixia Zhang and Vern Paxson. Stream Control Transmission Protocol, RFC 2960, October 2000.

[15] Simon Schütz, Lars Eggert, Stefan Schmid and Marcus Brunner. Protocol Enhancements for Intermittently Connected Hosts. ACM Computer Communication Review (CCR), Vol. 35, No. 3, July 2005.

[16] Randy Presuhn. Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP). RFC 3416, December 2002.

[17] Hari Balakrishnan, Hariharan Rahul, and Srinivasan Seshan. An Integrated Congestion Management Architecture for Internet Hosts. Proc. ACM SIGCOMM, Cambridge, MA, September 1999.

[18] Hari Balakrishnan and Srinivasan Seshan. The Congestion Manager. RFC 3124, June 2001.

[19] Charlie Perkins (ed.) IP Mobility Support for IPv4. RFC 3344, August 2002.

[20] Vijay Devarapalli, Ryuji Wakikawa, Alexandru Petrescu and Pascal Thubert. Network Mobility (NEMO) Basic Support Protocol. RFC 3963, January 2005.

[21] Robert Moskowitz and Pekka Nikander. Host Identity Protocol (HIP) Architecture. RFC 4423, May 2006.

[22] Erik Nordmark and Marcelo Bagnulo. Level 3 Multihoming Shim Protocol. Work in Progress (Internet-Draft draft-ietf-shim6-proto-05), May 2006.

[23] Spencer Dawkins, Carl E. Williams and Alper E. Yegin. Problem Statement for Triggers for Transport (TRIGTRAN). Expired Work in Progress (Internet-Draft draft-dawkins-trigtran-probstmt-01), March 2003.

[24] David Clark and David Tennenhouse. Architectural Considerations for a New Generation of Protocols. Proc. ACM SIGCOMM, Philadelphia, PA, USA, September 1990.

[25] Lars-Ake Larzon, Mikael Degermark, Stephen Pink, Lars-Erik Jonsson and Godred Fairhurst. The Lightweight User Datagram Protocol (UDP-Lite). RFC 3828, July 2004.

[26] Ramón Cáceres and Liviu Iftode. Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments. IEEE Journal on Selected Areas in Communications (JSAC), Vol. 13, No. 5, 1995.

[27] Hala Elaarag. Improving TCP Performance over Mobile Networks. ACM Computing Surveys, Vol. 34, No. 3, September 2002.

[28] Suresh Krishnan, Nicolas Montavont, Eric Njedjou, Siva Veerepalli and Alper E. Yegin. Link-layer Event Notifications for Detecting Network Attachments. Work in Progress (Internet-Draft draft-ietf-dna-link-information-04), October 2006.

[29] Wesley M. Eddy and Yogesh P. Swami. Adapting End-Host Congestion Control for Mobility. Technical Report CR-2005-213838, NASA Glenn Research Center, July 2005.