Middlebox Models Compatible with the Internet

Joe Touch
USC/ISI
4676 Admiralty Way
Marina del Rey, CA 90292-6695 USA
touch@isi.edu

***Abstract*— A hybrid model for middleboxes is presented that describes constraints on their compatibility with the Internet. The Internet is composed of hosts, routers, and links that exchange messages, and these components have been combined into hybrid models to describe tunnels and virtual routers. This document extends these models to describe the behavior of a variety of types of middleboxes, including network address translators, proxies, and transparent proxies.**

Keywords—Internet architecture, middlebox, NAT, proxy, PEP, tunnel.

I. INTRODUCTION

Middleboxes are devices that bend – and sometimes break – the network architecture in which they are deployed [RFC3234]. They represent an intermediate between the role of *host* and *router*, transiting traffic while also modifying it. As a result, they can be modeled as hybrid components in a forwarding-based architecture to better understand their behavior and limitations.

The following is a discussion of the Internet and the role of middleboxes therein. A simple Internet architecture is presented, as are existing models for tunnels and encapsulation subnets. Hybrid models of middleboxes are introduced that explain the conditions under which these devices can safely participate in the Internet architecture.

II. THE INTERNET ARCHIECTURE

A network architecture is an abstract description of a distributed communications system, its components and their relationships, the requisite properties of those components, and the resulting properties of that system. Such descriptions can help explain behavior, as when the OSI seven-layer model is used as a teaching example [Zi80]. Architectures describe capabilities – and, just as importantly, constraints.

A network can be defined as a system of endpoints and relays interconnected by communication paths, abstracting away issues of naming in order to focus on message delivery. To the extent that the Internet has a single, coherent interpretation, its architecture is defined by its core protocols (IP [RFC791], TCP [RFC793], UDP [RFC768]) and the concepts [ToFPN][To03] depicted in Figure 1:

- **Message**: variable length data labeled with globally-unique endpoint IDs [RFC791]
- **Host**: a device that sources and/or sinks messages labeled from/to its IDs [RFC1122]
- **Router**: a device that relays messages using longest-prefix match of destination IDs and local context, when possible [RFC1812]
- **Link**: a communication device that transfers messages between network devices [RFC1122][RFC1812]

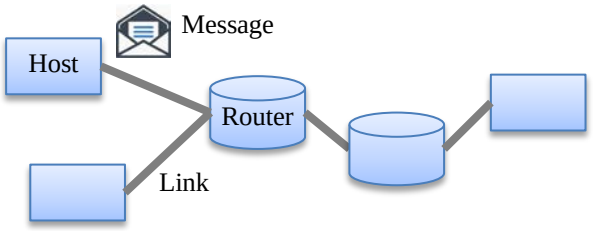

Figure 1 The Internet architecture

As a network architecture, the Internet is a system of *hosts* and *routers* interconnected by *links* that exchange *messages* when possible. "When possible" describes the Internet's "best effort" principle. Hosts that are not directly connected can communicate indirectly through a sequence of links connected by routers, called a *path*. The limited role of routers and links represents the "End-to-End Principle" [Sa84], and longest-prefix match supports hierarchical forwarding using efficient representations of relaying tables.

Although the definitions of host, router, and link seem absolute, they are often viewed relative to the context of only one OSI layer. Each OSI layer can thus be viewed as a distinct network architecture. An Internet gateway is a Layer 3 *router* when it transits IP datagrams but it acts as a Layer 2 *host* when it sources or sinks Layer 2 messages on attached links to accomplish this transit capability. In this way, a single device (Internet gateway) behaves as different components (router, host) at different layers.

Even though a single device may have multiple roles – even concurrently – at a given layer, each role is typically static and pre-determined. An Internet gateway always acts as a Layer 2 host, and that behavior does not depend on where the gateway is viewed from within Layer 2. In the context of a single layer, a device's behavior is modeled as a single component from all viewpoints in that layer.

III. ARCHITECTURE EXTENSIONS

The Internet architecture can be extended to explain the role of new devices and to understand how those devices affect the roles of existing components. Two illustrative

architectural extensions that are already understood are tunnels and encapsulation networks.

A. *Tunnels as links*

A tunnel can be modeled as a network that emulates a link in another network [To03][To16]. It consists of two devices (*ingress*, *egress*), connected by a network ***N,*** that lie along a path within a (possibly different) network ***M*** (shown in its entirety in Figure 2). Messages arriving at the ingress are encapsulated to traverse network ***N***, addressed to the egress. The egress decapsulates those messages, which continue on network ***M*** as if emerging from a link.

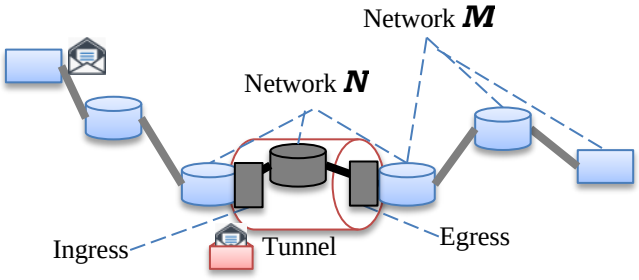

Figure 2 A tunnel – the big picture

Within network ***N***, *i.e.*, from inside the tunnel itself, the ingress is a source and the egress is a sink – both are effectively hosts on network ***N*** (Figure 3). From outside the tunnel, to network ***M***, the entire tunnel acts as a single link in network ***M*** (Figure 4). As with any other type of link, a tunnel thus begins and ends at a network interface – *i.e.*, the ingress and egress are more specifically the network interfaces inside routers in network ***M*** (even though we use the terms ingress/egress to also refer to the nodes where they are located). The model of each component (ingress, egress) and the entire system (tunnel) depends on the layer from which you view the tunnel.

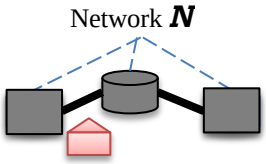

Figure 3 A tunnel as viewed from within the tunnel

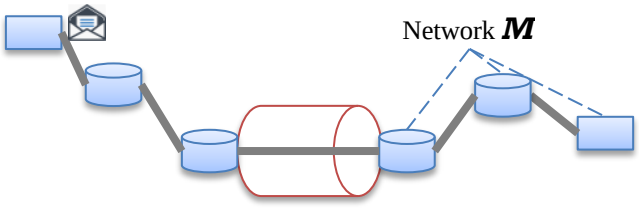

Figure 4 A tunnel, as viewed from the outside

This approach thus highlights a few key features of a tunnel as a network architecture construct:

- Tunnels create a direct link from existing connectivity, *i.e.*, from one or more existing links or paths
- Tunnels rely on devices along that path to act as hosts, to source and sink messages over the path
- A link is a tunnel whose ingress/egress are directly connected by a communication channel [ToFPN]

This model of a tunnel can be used to infer constraints. Because the entire tunnel acts as a link, its ingress and egress should not act directly as either routers or hosts on network ***M***. If the egress is down, an ICMP "host not found" inside the tunnel should be translated as "link down" (or interface down) outside the tunnel, and the router attached to the tunnel should react accordingly. This is why the ingress and egress are typically integrated as virtual interfaces within routers in network ***M***.

This model also highlights how tunnels challenge some of the prevalent assumptions of the Internet architecture. Internet links are generally assumed to have a static MTU, but tunnel MTUs can depend on the path taken in network ***N***. As a result, a tunnel is a link with a variable MTU in network ***M***, and thus network ***M*** needs to accommodate links (not just paths) with varying MTUs. This complicates path MTU discovery.

Networking requirements (such as MTU) and signals (such as ICMP messages) need to be translated from one layer to the other whenever they encounter an ingress or egress. Correctly translating and relaying these properties and behaviors between the layers is as important as properly encapsulating or decapsulating messages at the ingress and egress, and often defines the robustness and completeness of a tunnel solution.

B. *Encapsulation nets as virtual routers*

An encapsulation network is the network equivalent of a multipoint tunnel; messages of network ***M*** are tunneled over network ***N***, using multiple ingresses and egresses.

The Internet has two such models of multipoint tunnels: a Layer 2 subnet (a multipoint link) and a virtual router. Layer 2 subnets (*i.e.*, link layers), are links with more than two endpoints. The Internet assumes Layer 2 subnets support broadcast, *e.g.*, for endpoint name translation (*e.g.*, ARP) or multicast. When broadcast is missing, it needs to be emulated (*e.g.*, LANE [RFC1577] and MARS [RFC2022]).

A more recent model is the virtual router, first proposed in virtual networks (X-Bone *vrouter* [To98][To03]) and later in the IETF at both Layer 2 (rbridges/TRILL [Pe04][RFC5556]) and Layer 3 (LISP [RFC6830]). The X-Bone first proposed that these devices be modeled as a single router. Figure 5 shows such a system, in which network M devices traverse a network N subnet (shaded) *via* encapsulation.

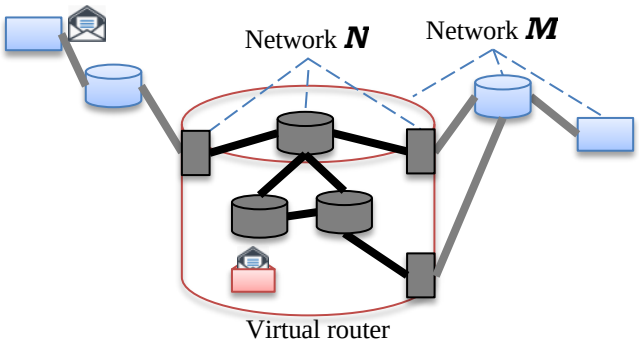

Figure 5 A virtual router – the big picture

Viewed from the outside (in the context of network ***M***), the ingresses and egresses act as virtual network interfaces of the virtual router (Figure 6). Whatever forwarding is required inside the virtual router is hidden from the outside network (***M***). Viewed from inside the virtual router, the ingresses and egresses are hosts in network ***N*** (Figure 7).

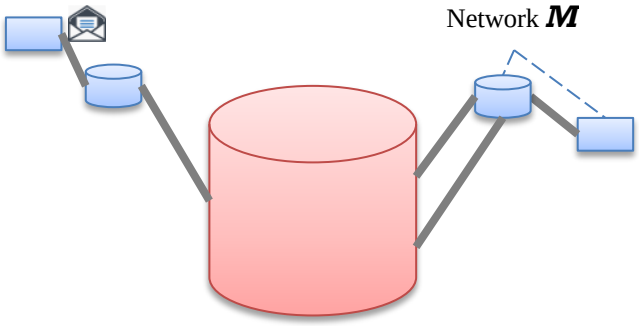

Figure 6 A virtual router viewed from the outside

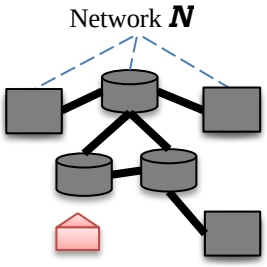

Figure 7 A virtual router viewed from within

Tunnels combine devices to create a virtual link out of a path and virtual routers combine devices to create a router out of a set of paths. Both models assume that network ***N*** can forward messages between ingress and egress using network ***N***'s pre-existing routing.

Using this model, the benefit of virtual routers is clear: they isolate the structure of network ***M*** from network ***N***, *i.e.*, network ***M*** is invisible from within network ***N***.

The virtual router model can be used to resolve protocol issues. A TRILL encapsulation network ("campus") can be modeled as a single, virtual Layer 2 route, *i.e.*, an Ethernet bridge. By that model, bridge discovery (BPDU) messages should be processed by a campus as if by a single bridge; that interpretation would have simplified the TRILL specifications (this was proposed but rejected by the IETF working group and has resulted in substantial and unnecessarily complex BPDU handling rules).

Again, as with tunnels, a virtual router can be viewed as different network architecture components depending on viewpoint. From inside the virtual router, the ingresses and egresses act as hosts and the transits act as routers. From outside the virtual router, the entire system acts as a single router that relays messages between its inputs and outputs. From either viewpoint – inside or outside – the virtual router construct and its components behave consistently with respect to the current Internet architecture.

Also, as with tunnels, translating and potentially relaying the requirements and behaviors between the inside and outside views is critical. Again, the translation between inside and outside is a translation between layers, where the internal network ***N*** of the virtual router acts like a lower layer to network transiting the external network ***M***.

C. Multipoint tunnels vs. virtual routers

The tunnels in the figures shown in Figure 2, Figure 3, and Figure 4 are point-to-point, but tunnels can also be multipoint (Figure 8). A multipoint tunnel is a tunnel with more than two ingress/egress locations.

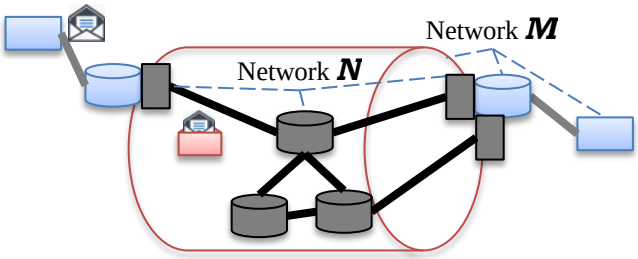

Figure 8 A multipoint tunnel – the big picture

A multipoint link (i.e., a conventional Layer 2 multipoint subnet, *e.g.*, Ethernet), a multipoint tunnel, and a virtual router all require address resolution to provide the information needed for ingress encapsulation. When messages arrive at the ingress of a point-to-point link or tunnel, the egress is always known – there is only one. When a message arrives at the ingress of a multipoint link, multipoint tunnel, or a virtual router, the egress can vary. All three thus require a way to determine the appropriate egress in order to properly encapsulate the message for traversal over network ***N***. For IP messages traversing an Ethernet multipoint link, this egress resolution mechanism provided by ARP.

Just as a point-to-point link is a tunnel whose ingress/egress are connected by a point-to-point communication channel, a multipoint link is a tunnel whose ingresses/egresses are connected by a multi-access communication channel. A multipoint tunnel and a virtual router have one important distinction – they differ in where the encapsulation occurs. A multipoint tunnel encapsulates at virtual interfaces in the "outside" network ***M***, whereas a

virtual router encapsulates using virtual interfaces that belong to the virtual router. Another way of viewing this is that multipoint tunnels interconnect hosts, whereas virtual routers interconnect links or tunnels.

IV. Hybrid Models for Middleboxes

Middleboxes include a wide variety of non-router, non-link devices inside a network, including NATs, traditional proxies, and "transparent" proxies [RFC3234]. Like routers and links, they transit messages – but unlike routers and links, they modify those messages. They cannot be modeled as any single Internet component when viewed from the perspective of a single layer.

A middlebox can be defined as a network device that deliberately modifies the semantics of messages in transit or that sources or sinks messages that mimic the behavior of the endpoints of a message's path.

Although there are a wide variety of middelbox behaviors, they generally fall into three distinct classes:

- **NAT**: a network transit device that modifies messages by translating network layer addresses and/or transport layer port numbers.
- **Traditional proxy**: (or just "proxy") a transport-layer destination host that relays content by acting as a source host
- **Transparent proxy**: a middlebox that is not solely a NAT or traditional proxy, often combining the capabilities of the two or modifying messages in transit other than just the network address and transport port number.

Just as a tunnel and encapsulation subnet can be described using these hybrid models, the following hybrid models for various middlebox types are proposed:

- **NAT:** viewed as a router or link to devices on the private side, but a host on the public side [RFC3022]
- **Proxy:** viewed as different hosts on each side [Sh86]
- **Transparent proxy:** viewed, together with the source, as a single host to the sink but invisible to the source on both sides; these are sometimes called "performance-enhancing proxies" (PEP) [RFC3135]

Unlike our tunnel and virtual router models, these models are not layer-dependent but they are dependent on the location of the viewpoint within the layer. A NAT looks like a router or link on one side and a host on the other, but both views are within the network and transport layers (which, in the Internet, are intertwined). From the link layer, a NAT looks like a host, as would both conventional hosts and network routers.

A. The View of a NAT

A NAT is modeled as a host on the public side because there it sources and sinks messages with its address (Figure 9, right). It is modeled as a router or link to the private side because it the private-side device thinks the NAT is transiting its traffic (Figure 9, left shaded). It is modeled as a router when its private side is connected to a link; it is modeled as a link when its private side is integrated within a router or host, such that there is no separate link connection.

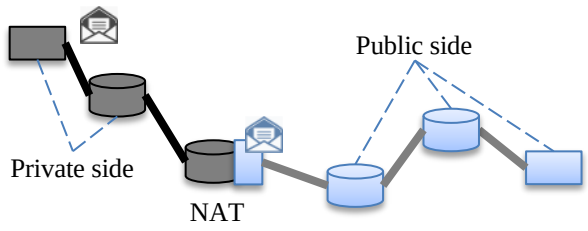

Figure 9 A NAT – the big picture

When viewed from the private side, the NAT is just a router along a path that goes all the way to the public sink host (Figure 10). The NAT is thus either a router or a link (depending on details, as noted before). The host thinks the entire network is homogenous and goes all the way to the destination. This is why a private host can trace the entire path, including routers on the private side and public side, all the way to the public destination host.

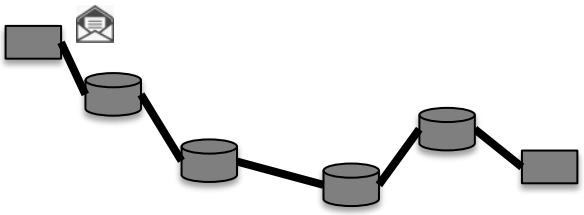

Figure 10 A NAT viewed from the private side

When viewed from the public side, the private source isn't even part of the network (Figure 11). The entire system begins at the NAT and the NAT is a host on the public network. This is why the destination cannot trace back into the private network, and this is also what makes the private side "private".

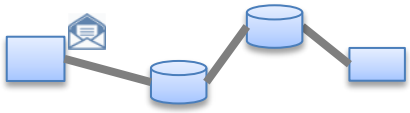

Figure 11 A NAT as viewed from the pubic (Internet) side

This hybrid model of NATs highlights the key challenge of integrating them with the Internet architecture. From the private viewpoint they act as routers along a path, but from the public viewpoint they act as hosts. Within a single network layer they act as two different devices. Requirements (*e.g.*, MTU) and signals (*e.g.*, ICMP) between the two sides need to be translated and potentially

relayed. Additionally, any message terminated on the public side of the NAT might also create a signal or new message to relay to the private side.

B. The View of a Conventional Proxy

A (conventional) proxy acts as different hosts on both sides (Figure 12). On both sides, it sources and sinks messages with its address. Although it transits information between its two sides, neither side knows.

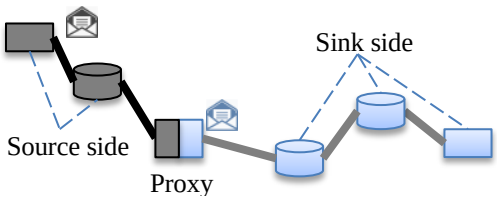

Figure 12 A proxy – the big picture

As a result, from the source side, the sink side does not exist (Figure 13), and from the sink side, the source side does not exist (Figure 14). Messages received by the proxy from either direction can cause new messages to be generated on the other side – this includes not only data that is relayed, but also signals. Requirements of the two different sides, such as MTU limits, can be isolated somewhat, depending on how the proxy is implemented.

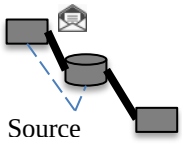

Figure 13 A proxy as viewed from the source side

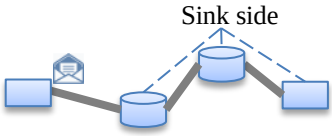

Figure 14 A proxy as viewed from the sink side

The concept of a proxy is not new to the Internet architecture. Layer 3 relays act as Layer 2 proxies, terminating Layer 2 communication on one side and originating it on the other. As such, the behavior of conventional proxies is already part of the current Internet.

C. The View of a Transparent Proxy

Like a conventional proxy, a transparent proxy is modeled as different hosts on both sides (Figure 15). Conventional proxies separate the two sides, which act largely independently at the layer where the proxy is deployed. Transparent proxies mix things up a bit more.

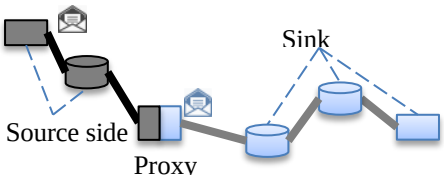

Figure 15 A transparent proxy – the big picture

From the source side, the proxy and sink act as a single host (Figure 16). The behavior of the proxy/sink pair is the device that interacts with the source and the source-side routers and links.

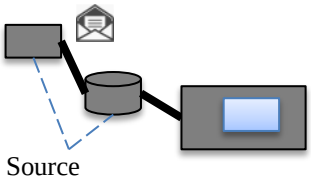

Figure 16 A transparent proxy as viewed from the source

Similarly, from the sink side, the source and the proxy act as a single source (Figure 17). The sink side interacts with the proxy/source pair.

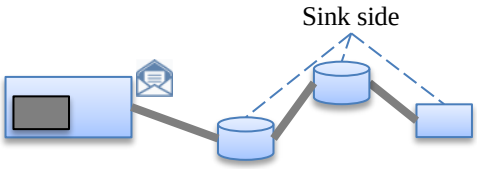

Figure 17 A Transparent proxy as viewed from the sink

A conventional proxy is a simple degenerate case of a transparent proxy, where the contribution of the "foreign" element (the source on the sink side, or the sink on the source side) is eliminated. For example, Figure 17 becomes identical to Figure 14 when the contribution of the source is eliminated. In the conventional case, the proxy is wholly responsible for independent behavior on its two sides.

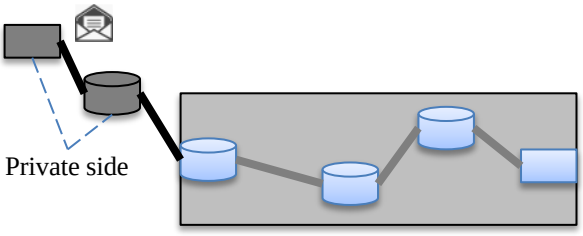

Figure 18 A NAT modeled as a transparent proxy

A NAT is complex degenerate case of a transparent proxy, where the proxy allows the entirety of the public network to be emulated (by the proxy) to the private side (Figure 18). The private-side host functions are now empty, and the only function of the box is to emulate the public network.

In each case, a middlebox is modeled by ***multiple*** components within the same network, a viewpoint that varies based on location. This is the defining property of a middlebox – and perhaps justification for using "middle" in its name: its architectural role ***depends on the location from which it is viewed***, not merely the layer. This represents a significant departure from the simple behavior of conventional network components.

V. THE IMPACT OF THE MIDDLEBOX MODELS

These three distinct middlebox models imply the following common constraints, which are required to ensure compatibility with the Internet architecture:

- **Middleboxes always emulate a host**, independently or in concert with the source, from at least one side's viewpoint
- **Middleboxes act as different components from different directions within the same layer**, either distinct components or different instances of one

Both of these rules imply that middleboxes cannot be deployed without consequence for the endpoints with which they interact, nor can they be implemented as simple, dumb translation devices. In specific, a middlebox can modify messages ***only*** when it ***knows*** it can correctly emulate the intended host, either in solo (for a NAT or proxy) or in tandem with the origin host (for a transparent proxy). ***When the middlebox lacks that certainty, it must not modify transited messages.***

For example, a transparent proxy might want to split or coalesce messages, to more efficiently match MTUs on different links. This is valid only when it knows it can coordinate both directions, *e.g.*, using TCP MSS. A proxy can ignore unknown TCP options only when it acts as if it terminates the connection, *e.g.*, emulating a host in both directions. Middleboxes cannot simply copy unknown options during message modification because they cannot know whether the options interact with the modifications.

To the extent that middleboxes coexist with the Internet, they already obey these rules. The Internet is already experiencing examples where extensions to end-to-end protocols or middlebox capabilities do not comply with these rules and cause problems that might not be resolvable without substantial revision to the Internet architecture.

A. Summary

Hybrid models for NAT, proxy, and transparent proxy middleboxes highlight how middlebox behavior is viewed differently from various locations in the same network, but how they always emulate a host from at least one side's viewpoint. The models also indicate constraints that are necessary to ensure middlebox consistency with the Internet architecture, especially when the middlebox is not certain it can accurately perform host emulation. Middleboxes can and sometimes do coexist with the Internet only when used consistent with these constraints.

ACKNOWLEDGMENTS

The author would like to thank Ted Faber for his detailed feedback. This work is partly supported by USC/ISI's Postel Center.

REFERENCES

[Pe04] Perlman, R., "RBridges: Transparent Routing," Proc. Infocom 2005, Mar. 2004.

[RFC768] Postel, J, "User Datagram Protocol," RFC 768, Aug. 1980.

[RFC791] Postel, J, "Internet Protocol," RFC 791, Sept. 1981.

[RFC793] Postel, J, "Transmission Control Protocol," RFC 793, Sept. 1981.

[RFC1122] Braden, R., (Ed.), "Requirements for Internet Hosts -- Communication Layers," RFC 1122, Oct. 1989.

[RFC1577] Laubach, M., "Classical IP and ARP over ATM," RFC 1577, Jan. 1994.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers," RFC 1812, June 1995.

[RFC2022] Armitage, G., "Support for Multicast over UNI 3.0/3.1 based ATM Networks," RFC 2022, Nov. 1996.

[RFC3022] Srisuresh, P., K. Egevang, "Traditional IP Network Address Translator," RFC 3022, Jan. 2001.

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

[RFC3234] Carpenter, B., S. Brim, "Middleboxes: Taxonomy and Issues," RFC 3234, Feb. 2002.

[RFC5556] Touch, J., R. Perlman, "Transparently Interconnecting Lots of Links (TRILL): Problem and Applicability Statement," RFC 5556, May 2009.

[RFC6830] Farinacci, D., V. Fuller, D. Meyer, D. Lewis, "The Locator/ID Separation Protocol," RFC 6830, Jan. 2013.

[Sa84] Saltzer, J., D. Reed, D. Clark, "End-to-end arguments in system design," *ACM Trans. on Computing Systems*, Nov. 1984.

[Sh86] Shapiro M., "Structure and Encapsulation in Distributed Systems: the Proxy Principle," Int. Conf. on Dist. Comp. Sys. (ICDCS), May 1986

[To98] Touch, J., S. Hotz, "The X-Bone," in Proc. Third Global Internet Mini-Conference, Proc. Globecom '98, Sydney, Australia Nov. 1998.

[To03] Touch, J., Y. Wang, L. Eggert, G. Finn, "Virtual Internet Architecture," USC/ISI Tech. Report 570, Aug. 2003.

[To16] Touch, J., M. Townsley, "Tunnels in the Internet Architecture," draft-ietf-intarea-tunnels (work in progress), July 2016.

[ToFPN] Touch, J., *A First-Principles Approach to Computer Networking*, (in preparation).

[Zi80] Zimmermann, H., "OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection," *IEEE Trans. on Comm.*, Apr. 1980.