Network Working Group
Request for Comments: 3234
Category: Informational

B. Carpenter
IBM Zurich Research Laboratory
S. Brim

February 2002

Middleboxes: Taxonomy and Issues

Status of this Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document is intended as part of an IETF discussion about "middleboxes" - defined as any intermediary box performing functions apart from normal, standard functions of an IP router on the data path between a source host and destination host. This document establishes a catalogue or taxonomy of middleboxes, cites previous and current IETF work concerning middleboxes, and attempts to identify some preliminary conclusions. It does not, however, claim to be definitive.

Table of Contents

1. Introduction and Goals

1.1. Terminology

The phrase "middlebox" was coined by Lixia Zhang as a graphic description of a recent phenomenon in the Internet. A middlebox is defined as any intermediary device performing functions other than the normal, standard functions of an IP router on the datagram path between a source host and destination host.

In some discussions, especially those concentrating on HTTP traffic, the word "intermediary" is used. For the present document, we prefer the more graphic phrase. Of course, a middlebox can be virtual, i.e., an embedded function of some other box. It should not be interpreted as necessarily referring to a separate physical box. It may be a device that terminates one IP packet flow and originates another, or a device that transforms or diverts an IP packet flow in some way, or a combination. In any case it is never the ultimate end-system of an applications session.

Normal, standard IP routing functions (i.e., the route discovery and selection functions described in [RFC 1812], and their equivalent for IPv6) are not considered to be middlebox functions; a standard IP router is essentially transparent to IP packets. Other functions taking place within the IP layer may be considered to be middlebox functions, but functions below the IP layer are excluded from the definition.

There is some discrepancy in the way the word "routing" is used in the community. Some people use it in the narrow, traditional sense of path selection based on IP address, i.e., the decision-making action of an IP router. Others use it in the sense of higher layer decision-making (based perhaps on a URL or other applications layer string). In either case it implies a choice of outbound direction, not the mere forwarding of a packet in the only direction available. In this document, the traditional sense is always qualified as "IP routing."

1.2. The Hourglass Model, Past and Future

The classical description of the Internet architecture is based around the hourglass model [HOURG] and the end-to-end principle [Clark88, Saltzer]. The hourglass model depicts the protocol architecture as a narrow-necked hourglass, with all upper layers riding over a single IP protocol, which itself rides over a variety of hardware layers.

The end-to-end principle asserts that some functions (such as security and reliability) can only be implemented completely and correctly end-to-end, with the help of the end points. The end-to-end principle notes that providing an incomplete version of such functions in the network itself can sometimes be useful as a performance enhancement, but not as a substitute for the end-to-end implementation of the function. The references above, and [RFC 1958], go into more detail.

In this architecture, the only boxes in the neck of the hourglass are IP routers, and their only function is to determine routes and forward packets (while also updating fields necessary for the forwarding process). This is why they are not classed as middleboxes.

Today, we observe deviations from this model, caused by the insertion in the network of numerous middleboxes performing functions other than IP forwarding. Viewed in one way, these boxes are a challenge to the transparency of the network layer [RFC 2775]. Viewed another way, they are a challenge to the hourglass model: although the IP layer does not go away, middleboxes dilute its significance as the single necessary feature of all communications sessions. Instead of concentrating diversity and function at the end systems, they spread diversity and function throughout the network.

This is a matter of concern for several reasons:

* New middleboxes challenge old protocols. Protocols designed without consideration of middleboxes may fail, predictably or unpredictably, in the presence of middleboxes.

* Middleboxes introduce new failure modes; rerouting of IP packets around crashed routers is no longer the only case to consider. The fate of sessions involving crashed middleboxes must also be considered.

* Configuration is no longer limited to the two ends of a session; middleboxes may also require configuration and management.

* Diagnosis of failures and misconfigurations is more complex.

1.4. Goals of this Document

The principle goal of this document is to describe and analyse the current impact of middleboxes on the architecture of the Internet and its applications. From this, we attempt to identify some general conclusions.

Goals that might follow on from this work are:

* to identify harmful and harmless practices,

* to suggest architectural guidelines for application protocol and middlebox design,

* to identify requirements and dependencies for common functions in the middlebox environment,

* to derive a system design for standardisation of these functions,

* to identify additional work that should be done in the IETF and IRTF.

An implied goal is to identify any necessary updates to the Architectural Principles of the Internet [RFC 1958].

The document initially establishes a catalogue of middleboxes, and cites previous or current IETF work concerning middleboxes, before proceeding to discussion and conclusions.

2. A catalogue of middleboxes

The core of this document is a catalogue of a number of types of middlebox. There is no obvious way of classifying them to form a hierarchy or other simple form of taxonomy. Middleboxes have a number of facets that might be used to classify them in a multidimensional taxonomy.

DISCLAIMER: These facets, many of distinctions between different types of middlebox, and the decision to include or exclude a particular type of device, are to some extent subjective. Not everyone who commented on drafts of this document agrees with our classifications and descriptions. We do not claim that the following catalogue is mathematically complete and consistent, and in some cases purely arbitrary choices have been made, or ambiguity remains. Thus, this document makes no claim to be definitive.

The facets considered are:

1. Protocol layer. Does the box act at the IP layer, the transport layer, the upper layers, or a mixture?

2. Explicit versus implicit. Is the middlebox function an explicit design feature of the protocol(s) in use, like an SMTP relay? Or is it an add-on not foreseen by the protocol design, probably attempting to be invisible, like a network address translator?

3. Single hop versus multi-hop. Can there be only one box in the path, or can there be several?

4. In-line versus call-out. The middlebox function may be executed in-line on the datapath, or it may involve a call-out to an ancillary box.

5. Functional versus optimising. Does the box perform a function without which the application session cannot run, or is the function only an optimisation?

6. Routing versus processing. Does the box simply choose which way to send the packets of a session, or does it actually process them in some way (i.e., change them or create a side-effect)?

7. Soft state versus hard state. If the box loses its state information, does the session continue to run in a degraded mode while reconstructing necessary state (soft state), or does it simply fail (hard state)?

8. Failover versus restart. In the event that a hard state box fails, is the session redirected to an alternative box that has a copy of the state information, or is it forced to abort and restart?

One possible classification is deliberately excluded: "good" versus "evil". While analysis shows that some types of middlebox come with a host of complications and disadvantages, no useful purpose would be served by simply deprecating them. They have been invented for compelling reasons, and it is instructive to understand those reasons.

The types of box listed below are in an arbitrary order, although adjacent entries may have some affinity. At the end of each entry is an attempt to characterise it in terms of the facets identified above. These characterisations should not be interpreted as rigid; in many cases they are a gross simplification.

Note: many types of middlebox may need to perform IP packet fragmentation and re-assembly. This is mentioned only in certain cases.

2.1 NAT

Network Address Translator. A function, often built into a router, that dynamically assigns a globally unique address to a host that doesn’t have one, without that host’s knowledge. As a result, the appropriate address field in all packets to and from that host is

translated on the fly. Because NAT is incompatible with application protocols with IP address dependencies, a NAT is in practice always accompanied by an ALG (Application Level Gateway - see below). It also touches the transport layer to the extent of fixing up checksums.

NATs have been extensively analysed in the IETF [RFC 2663, RFC 2993, RFC 3022, RFC 3027, etc.]

The experimental RSIP proposal complements NAT with a dynamic tunnel mechanism inserting a stateful RSIP server in place of the NAT [RSIP].

{1 IP layer, 2 implicit, 3 multihop, 4 in-line, 5 functional, 6 processing, 7 hard, 8 restart}

2.2 NAT-PT

NAT with Protocol Translator. A function, normally built into a router, that performs NAT between an IPv6 host and an IPv4 network, additionally translating the entire IP header between IPv6 and IPv4 formats.

NAT-PT itself depends on the Stateless IP/ICMP Translation Algorithm (SIIT) mechanism [RFC 2765] for its protocol translation function. In practice, SIIT and NAT-PT will both need an associated ALG and will need to touch transport checksums. Due to the permitted absence of a UDP checksum in IPv4, translation of fragmented unchecksummed UDP from IPv4 to IPv6 is hopeless. NAT-PT and SIIT also have other potential fragmentation/MTU problems, particularly when dealing with endpoints that don’t do path MTU discovery (or when transiting other middleboxes that break path MTU discovery). ICMP translation also has some intractable difficulties.

NAT-PT is a Proposed Standard from the NGTRANS WG [RFC 2766]. The Dual Stack Transition Mechanism adds a second related middlebox, the DSTM server [DSTM].

{1 IP layer, 2 implicit, 3 multihop, 4 in-line, 5 functional, 6 processing, 7 hard, 8 restart}

2.3 SOCKS gateway

SOCKSv5 [RFC 1928] is a stateful mechanism for authenticated firewall traversal, in which the client host must communicate first with the SOCKS server in the firewall before it is able to traverse the firewall. It is the SOCKS server, not the client, that determines the source IP address and port number used outside the firewall. The

client's stack must be "SOCKSified" to take account of this, and address-sensitive applications may get confused, rather as with NAT. However, SOCKS gateways do not require ALGs.

SOCKS is maintained by the AFT (Authenticated Firewall Traversal) WG.

{1 multi-layer, 2 explicit, 3 multihop, 4 in-line, 5 functional, 6 routing, 7 hard, 8 restart}

2.4 IP Tunnel Endpoints

Tunnel endpoints, including virtual private network endpoints, use basic IP services to set up tunnels with their peer tunnel endpoints which might be anywhere in the Internet. Tunnels create entirely new "virtual" networks and network interfaces based on the Internet infrastructure, and thereby open up a number of new services. Tunnel endpoints base their forwarding decisions at least partly on their own policies, and only partly if at all on information visible to surrounding routers.

To the extent that they deliver packets intact to their destinations, tunnel endpoints appear to follow the end-to-end principle in the outer Internet. However, the destination may be completely different from what a router near the tunnel entrance might expect. Also, the per-hop treatment a tunneled packet receives, for example in terms of QoS, may not be what it would have received had the packet traveled untunneled [RFC2983].

Tunnels also cause difficulties with MTU size (they reduce it) and with ICMP replies (they may lack necessary diagnostic information).

When a tunnel fails for some reason, this may cause the user session to abort, or an alternative IP route may prove to be available, or in some cases the tunnel may be re-established automatically.

{1 multi-layer, 2 implicit, 3 multihop, 4 in-line, 5 functional, 6 processing, 7 hard, 8 restart or failover}

2.5. Packet classifiers, markers and schedulers

Packet classifiers classify packets flowing through them according to policy and either select them for special treatment or mark them, in particular for differentiated services [Clark95, RFC 2475]. They may alter the sequence of packet flow through subsequent hops, since they control the behaviour of traffic conditioners.

Schedulers or traffic conditioners (in routers, hosts, or specialist boxes inserted in the data path) may alter the time sequence of packet flow, the order in which packets are sent, and which packets are dropped. This can significantly impact end-to-end performance. It does not, however, fundamentally change the unreliable datagram model of the Internet.

When a classifier or traffic conditioner fails, the user session may see any result between complete loss of connectivity (all packets are dropped), through best-effort service (all packets are given default QOS), up to automatic restoration of the original service level.

{1 multi-layer, 2 implicit, 3 multihop, 4 in-line, 5 optimising, 6 processing, 7 soft, 8 failover or restart}

2.6 Transport relay

Transport relays are basically the transport layer equivalent of an ALG; another (less common) name for them is a TLG. As with ALGs, they're used for a variety of purposes, some well established and meeting needs not otherwise met. Early examples of transport relays were those that ran on MIT's ITS and TOPS-20 PDP-10s on the ARPANET and allowed Chaosnet-only hosts to make outgoing connections from Chaosnet onto TCP/IP. Later there were some uses of TCP-TP4 relays. A transport relay between IPv6-only and IPv4-only hosts is one of the tools of IPv6 transition [TRANS64]. TLGs are sometimes used in combination with simple packet filtering firewalls to enforce restrictions on which hosts can talk to the outside world or to kludge around strange IP routing configurations. TLGs are also sometimes used to gateway between two instances of the same transport protocol with significantly different connection characteristics; it is in this sense that a TLG may also be called a TCP or transport spoofer. In this role, the TLG may shade into being an optimising rather than a functional middlebox, but it is distinguished from Transport Proxies (next section) by the fact that it makes its optimisations only by creating back-to- back connections, and not by modification or re-timing of TCP messages.

Terminating one TCP connection and starting another mid-path means that the TCP checksum does not cover the sender's data end-to-end. Data corruptions or modifications may be introduced in the processing when the data is transferred from the first to the second connection. Some TCP relays are split relays and have even more possibility of lost data integrity, because the there may be more than two TCP connections, and multiple nodes and network paths involved. In all cases, the sender has less than the expected assurance of data integrity that is the TCP reliable byte stream service. Note that

this problem is not unique to middleboxes, but can also be caused by checksum offloading TCP implementations within the sender, for example.

In some such cases, other session layer mechanisms such as SSH or HTTPS would detect any loss of data integrity at the TCP level, leading not to retransmission as with TCP, but to session failure. However, there is no general session mechanism to add application data integrity so one can detect or mitigate possible lack of TCP data integrity.

{1 Transport layer, 2 implicit, 3 multihop, 4 in-line, 5 functional (mainly), 6 routing, 7 hard, 8 restart}

2.7. TCP performance enhancing proxies

"TCP spoofer" is often used as a term for middleboxes that modify the timing or action of the TCP protocol in flight for the purposes of enhancing performance. Another, more accurate name is TCP performance enhancing proxy (PEP). Many TCP PEPs are proprietary and have been characterised in the open Internet primarily when they introduce interoperability errors with standard TCP. As with TLGs, there are circumstances in which a TCP PEP is seen to meet needs not otherwise met. For example, a TCP PEP may provide re-spacing of ACKs that have been bunched together by a link with bursty service, thus avoiding undesireable data segment bursts. The PILC (Performance Implications of Link Characteristics) working group has analyzed types of TCP PEPs and their applicability [PILCPEP]. TCP PEPs can introduce not only TCP errors, but also unintended changes in TCP adaptive behavior.

{1 Transport layer, 2 implicit, 3 multihop, 4 in-line, 5 optimising, 6 routing, 7 hard, 8 restart}

2.8. Load balancers that divert/munge packets.

There is a variety of techniques that divert packets from their intended IP destination, or make that destination ambiguous. The motivation is typically to balance load across servers, or even to split applications across servers by IP routing based on the destination port number. Except for rare instances of one-shot UDP protocols, these techniques are inevitably stateful as all packets from the same application session need to be directed to the same physical server. (However, a sophisticated solution would also be able to handle failover.)

To date these techniques are proprietary and can therefore only be applied in closely managed environments.

{1 multi-layer, 2 implicit, 3 single hop, 4 in-line, 5 optimising, 6 routing, 7 hard, 8 restart}

2.9. IP Firewalls

The simplest form of firewall is a router that screens and rejects packets based purely on fields in the IP and Transport headers (e.g., disallow incoming traffic to certain port numbers, disallow any traffic to certain subnets, etc.)

Although firewalls have not been the subject of standardisation, some analysis has been done [RFC 2979].

Although a pure IP firewall does not alter the packets flowing through it, by rejecting some of them it may cause connectivity problems that are very hard for a user to understand and diagnose.

"Stateless" firewalls typically allow all IP fragments through since they do not contain enough upper-layer header information to make a filtering decision. Many "stateful" firewalls therefore reassemble IP fragments (and re-fragment if necessary) in order to avoid leaking fragments, particularly fragments that may exploit bugs in the reassembly implementations of end receivers.

{1 IP layer, 2 implicit, 3 multihop, 4 in-line, 5 functional, 6 routing, 7 hard, 8 restart}

2.10. Application Firewalls

Application-level firewalls act as a protocol end point and relay (e.g., an SMTP client/server or a Web proxy agent). They may

(1) implement a "safe" subset of the protocol,

(2) perform extensive protocol validity checks,

(3) use an implementation methodology designed to minimize the likelihood of bugs,

(4) run in an insulated, "safe" environment, or

(5) use some combination of these techniques in tandem.

Although firewalls have not been the subject of standardisation, some analysis has been done [RFC 2979]. The issue of firewall traversal using HTTP has been discussed [HTTPSUB].

{1 Application layer, 2 implicit, 3 multihop, 4 in-line, 5 functional, 6 processing, 7 hard, 8 restart}

2.11. Application-level gateways

These come in many shapes and forms. NATs require ALGs for certain address-dependent protocols such as FTP; these do not change the semantics of the application protocol, but carry out mechanical substitution of fields. At the other end of the scale, still using FTP as an example, gateways have been constructed between FTP and other file transfer protocols such as the OSI and DECnet (R) equivalents. In any case, such gateways need to maintain state for the sessions they are handling, and if this state is lost, the session will normally break irrevocably.

Some ALGs are also implemented in ways that create fragmentation problems, although in this case the problem is arguably the result of a deliberate layer violation (e.g., mucking with the application data stream of an FTP control connection by twiddling TCP segments on the fly).

{1 Application layer, 2 implicit or explicit, 3 multihop, 4 in-line, 5 functional, 6 processing, 7 hard, 8 restart}

2.12. Gatekeepers/ session control boxes

Particularly with the rise of IP Telephony, the need to create and manage sessions other than TCP connections has arisen. In a multimedia environment that has to deal with name lookup, authentication, authorization, accounting, firewall traversal, and sometimes media conversion, the establishment and control of a session by a third-party box seems to be the inevitable solution. Examples include H.323 gatekeepers [H323], SIP servers [RFC 2543] and MEGACO controllers [RFC 3015].

{1 Application layer, 2 explicit, 3 multihop, 4 in-line or call-out, 5 functional, 6 processing, 7 hard, 8 restart?}

2.13. Transcoders

Transcoders are boxes performing some type of on-the-fly conversion of application level data. Examples include the transcoding of existing web pages for display on hand-held wireless devices, and transcoding between various audio formats for interconnecting digital mobile phones with voice-over-IP services. In many cases, such transcoding cannot be done by the end-systems, and at least in the case of voice, it must be done in strict real time with extremely rapid failure recovery.

Not all media translators are mandatory. They may simply be an optimisation. For example, in the case of multicast, if all the low-bandwidth receivers sit in one "corner" of the network, it would be inefficient for the sender to generate two streams or send both stream all the way across the network if the "thin" one is only needed far away from the sender. Generally, media translators are only useful if the two end systems don't have overlapping codecs or if the overlapping set is not a good network match.

{1 Application layer, 2 explicit or implicit, 3 single hop, 4 in-line, 5 functional, 6 processing, 7 hard?, 8 restart or failover}

2.14. Proxies

HTTP1.1 [RFC 2616] defines a Web proxy as follows:

> "An intermediary program which acts as both a server and a client for the purpose of making requests on behalf of other clients. Requests are serviced internally or by passing them on, with possible translation, to other servers. A proxy MUST implement both the client and server requirements of this specification. A "transparent proxy" is a proxy that does not modify the request or response beyond what is required for proxy authentication and identification. A "non-transparent proxy" is a proxy that modifies the request or response in order to provide some added service to the user agent, such as group annotation services, media type transformation, protocol reduction, or anonymity filtering."

A Web proxy may be associated with a firewall, when the firewall does not allow outgoing HTTP packets. However, HTTP makes the use of a proxy "voluntary": the client must be configured to use the proxy.

Note that HTTP proxies do in fact terminate an IP packet flow and recreate another one, but they fall under the definition of "middlebox" given in Section 1.1 because the actual applications sessions traverse them.

SIP proxies [RFC 2543] also raise some interesting issues, since they can "bend" the media pipe to also serve as media translators. (A proxy can modify the session description so that media no longer travel end-to-end but to a designated intermediate box.)

{1 Application layer, 2 explicit (HTTP) or implicit (interception), 3 multihop, 4 in-line, 5 functional, 6 processing, 7 soft, 8 restart}.

Note: Some so-called Web proxies have been implemented as "interception" devices that intercept HTTP packets and re-issue them with their own source address; like NAT and SOCKs, this can disturb address-sensitive applications. Unfortunately some vendors have caused confusion by mis-describing these as "transparent" proxies. Interception devices are anything but transparent. See [WREC] for a full discussion.

2.15. Caches

Caches are of course used in many shapes and forms in the Internet, and are in principle distinct from proxies. Here we refer mainly to content caches, intended to optimise user response times. HTTP makes provision for proxies to act as caches, by providing for both expiration and re-validation mechanisms for cached content. These mechanisms may be used to guarantee that specific content is not cached, which is a requirement for transient content, particularly in transactional applications. HTTP caching is well described in Section 13 of [RFC 2616], and in the HTTP case caches and proxies are inextricably mixed.

To improve optimisation, caching is not uniquely conducted between the origin server and the proxy cache directly serving the user. If there is a network of caches, the nearest copy of the required content may be in a peer cache. For this an inter-cache protocol is required. At present the most widely deployed solution is Internet Cache Protocol (ICP) [RFC 2186] although there have been alternative proposals such as [RFC 2756].

It can be argued that caches terminate the applications sessions, and should not be counted as middleboxes (any more than we count SMTP relays). However, we have arbitrarily chosen to include them since they do in practice re-issue the client’s HTTP request in the case of a cache miss, and they are not the ultimate source of the application data.

{1 Application layer, 2 explicit (if HTTP proxy caches), 3 multihop, 4 in-line, 5 functional, 6 processing, 7 soft, 8 restart}

2.16. Modified DNS servers

DNS servers can play games. As long as they appear to deliver a syntactically correct response to every query, they can fiddle the semantics. For example, names can be made into "anycast" names by arranging for them to resolve to different IP addresses in different parts of the network. Or load can be shared among different members of a server farm by having the local DNS server return the address of

different servers in turn. In a NAT environment, it is not uncommon for the FQDN-to-address mapping to be quite different outside and inside the NAT ("two-faced DNS").

Modified DNS servers are not intermediaries in the application data flow of interest. They are included here because they mean that independent sessions that at one level appear to involve a single host actually involve multiple hosts, which can have subtle effects. State created in host A.FOR.EXAMPLE by one session may turn out not to be there when a second session apparently to the same host is started, because the DNS server has directed the second session elsewhere.

If such a DNS server fails, users may fail over to an alternate DNS server that doesn't know the same tricks, with unpredicatble results.

{1 Application layer, 2 implicit, 3 multihop, 4 in-line (on DNS query path), 5 functional or optimising, 6 processing, 7 soft, 8 failover}

2.17. Content and applications distribution boxes

An emerging generalisation of caching is content distribution and application distribution. In this model, content (such as static web content or streaming multimedia content) is replicated in advance to many widely distributed servers. Further, interactive or even transactional applications may be remotely replicated, with some of their associated data. Since this is a recent model, it cannot be said that there is an industry standard practice in this area. Some of the issues are discussed in [WREC] and several new IETF activities have been proposed in this area.

Content distribution solutions tend to play with URLs in one way or another, and often involve a system of middleboxes - for example using HTTP redirects to send a request for WWW.EXAMPLE.COM off to WWW.EXAMPLE.NET, where the latter name may be an "anycast" name as mentioned above, and will actually resolve in DNS to the nearest instance of a content distribution box.

As with caches, it is an arbitrary choice to include these devices, on the grounds that although they terminate the client session, they are not the ultimate origin of the applications data.

{1 Application layer, 2 implicit or explicit, 3 multihop, 4 in-line or call-out, 5 optimising, 6 routing or processing, 7 soft, 8 restart?}

2.18. Load balancers that divert/munge URLs

Like DNS tricks, URL redirects can be used to balance load among a pool of servers - essentially a local version of a content distribution network. Alternatively, an HTTP proxy can rewrite HTTP requests to direct them to a particular member of a pool of servers.

These devices are included as middleboxes because they divert an applications session in an arbitrary way.

{1 Application layer, 2 explicit, 3 single hop, 4 in-line, 5 functional, 6 routing, 7 soft, 8 restart}

2.19. Application-level interceptors

Some forms of pseudo-proxy intercept HTTP packets and deliver them to a local proxy server instead of forwarding them to the intended destination. Thus the destination IP address in the packet is ignored. It is hard to state whether this is a functional box (i.e., a non-standard proxy) or an optimising box (i.e., a way of forcing the user to use a cache). Like any non-standard proxy, it has undefined consequences in the case of dynamic or non-cacheable content.

{1 Application layer, 2 implicit, 3 single hop, 4 in-line, 5 functional or optimising, 6 routing, 7 hard, 8 restart}

2.20. Application-level multicast

Some (mainly proprietary) applications, including some approaches to instant messaging, use an application-level mechanism to replicate packets to multiple destinations.

An example is given in [CHU].

{1 Application layer, 2 explicit, 3 multihop, 4 in-line, 5 functional, 6 routing, 7 hard, 8 restart}

2.21. Involuntary packet redirection

There appear to be a few instances of boxes that (based on application level content or other information above the network layer) redirect packets for functional reasons. For example, more than one "high speed Internet" service offered in hotel rooms intercepts initial HTTP requests and diverts them to an HTTP server that demands payment before opening access to the Internet. These boxes usually also perform NAT functions.

{1 multi-layer, 2 implicit, 3 single hop, 4 call-out, 5 functional, 6 routing, 7 hard, 8 restart}

2.22. Anonymisers

Anonymiser boxes can be implemented in various ways that hide the IP address of the data sender or receiver. Although the implementation may be distinct, this is in practice very similar to a NAT plus ALG.

{1 multi-layer, 2 implicit or explicit, 3 multihop, 4 in-line, 5 functional, 6 processing, 7 hard, 8 restart}

2.23. Not included

Some candidates suggested for the above list were excluded for the reasons given below. In general, they do not fundamentally change the architectural model of packet delivery from source to destination.

Bridges and switches that snoop ARP, IGMP etc. These are below the IP layer, but use a layer violation to emulate network layer functions. They do not change IP layer functions.

Wiretaps and snoopers in general - if they are working correctly, they have no impact on traffic, so do not require analysis.

Mobile IP home agents are intended to assist packet delivery to the originally desired destination, so they are excluded on the same grounds as standard routers.

Relays in interplanetary networks - although these would certainly appear to be middleboxes, they are not currently deployed.

2.24. Summary of facets

By tabulating the rough classifications above, we observe that of the 22 classes of middlebox described:

17 are application or multi-layer
16 are implicit (and others are explicit OR implicit)
17 are multi-hop
21 are in-line; call-out is rare
18 are functional; pure optimisation is rare
Routing & processing are evenly split
16 have hard state
21 must restart session on failure

We can deduce that current types of middlebox are predominantly application layer devices not designed as part of the relevant protocol, performing required functions, maintaining hard state, and aborting user sessions when they crash. Indeed this represents a profound challenge to the end-to-end hourglass model.

3. Ongoing work in the IETF and elsewhere

Apart from work cited in references above, current or planned work in the IETF includes:

MIDCOM - a working group with focus on the architectural framework and the requirements for the protocol between a requesting device and a middlebox and the architectural framework for the interface between a middlebox and a policy entity [MIDFRAME, MIDARCH]. This may interact with session control issues [SIPFIRE].

Work is also proceeding outside the MIDCOM group on middlebox discovery [MIDDISC].

WEBI (Web Intermediaries) - a working group that addresses specific issues in the world wide web infrastructure (as identified by the WREC working group), by providing generic mechanisms which are useful in several application domains (e.g., proxies, content delivery surrogates). Specific mechanisms will be Intermediary Discovery and Description and a Resource Update Protocol.

Intermediaries are also an important focus in the development of XML Protocol by the World-Wide Web Consortium, who have published an interesting analysis [XMLPI].

OPES (Open Pluggable Extension Services) - a proposed working group whose output will enable construction of services executed on application data by participating transit intermediaries. Caching is the most basic intermediary service, one that requires a basic understanding of application semantics by the cache server.

CDI (Content Distribution Internetworking) is a potential working group for allowing cooperation between different Content Distribution Networks and cache clusters [CDNP].

RSERPOOL (Reliable Server Pooling) is a working group that will define architecture and requirements for management and access to server pools, including requirements from a variety of applications, building blocks and interfaces, different styles of pooling, security requirements and performance requirements, such as failover times and coping with heterogeneous latencies.

4. Comments and Issues

A review of the list in Section 2 suggests that middleboxes fit into one or more of three broad categories:

1) mechanisms to connect dissimilar networks to enable cross-protocol interoperability;

2) mechanisms to separate similar networks into zones, especially security zones;

3) performance enhancement.

As observed in [RFC 2775], the rise of middleboxes puts into question the general applicability of the end-to-end principle [RFC 1958]. Middleboxes introduce dependencies and hidden points of failure that violate the fate-sharing aspect of the end-to-end principle. Can we define architectural principles that guarantee robustness in the presence of middleboxes?

4.1. The end to end principle under challenge

Many forms of middlebox are explicitly addressed at the IP level, and terminate a transport connection (or act as a final destination for UDP packets) in a normal way. Although they are potential single points of failure, they do not otherwise interfere with the end to end principle [RFC 1958]. (This statement does not apply to transport relays or TCP spoofers; they do not terminate a transport connection at the expected destination in the normal way.)

However, there is a general feeling that middleboxes that divert an IP packet from its intended destination, or substantively modify its content on the fly, are fundamentally different from those that correctly terminate a transport connection and carry out their manipulations at applications level. Such diversion or modification violates the basic architectural assumption that packets flow from source to destination essentially unchanged (except for time-to-live and QOS-related fields). The effects of such changes on transport and applications is unpredictable in the general case. Much of the analysis that applies to NAT [RFC 2993, RFC 3027] will also apply to RSIP, NAT-PT, DSTM, SOCKS, and involuntary packet redirectors. Interception proxies, anonymisers, and some types of load balancer can also have subtle effects on address-sensitive applications, when they cause packets to be delivered to or from a different address. Transport relays and TCP spoofers may deceive applications by delivering an unreliable service on a TCP socket.

We conclude that:

> Although the rise of middleboxes has negative impact on the end to end principle at the packet level, it does not nullify it as a useful or desirable principle of applications protocol design. However, future application protocols should be designed in recognition of the likely presence of network address translation, packet diversion, and packet level firewalls, along the data path.

4.2. Failure handling

If a middlebox fails, it is desirable that the effect on sessions currently in progress should be inconvenient rather than catastrophic. There appear to be three approaches to achieve this:

> Soft state mechanisms. The session continues in the absence of the box, probably with reduced performance, until the necessary session state is recreated automatically in an alternative box (or the original one, restarted). In other words the state information optimises the user session but is not essential. An example might be a true caching mechanism, whose temporary failure only reduces performance.
>
> Rapid failover mechanisms. The session is promptly redirected to a hot spare box, which already has a copy of the necessary session state.
>
> Rapid restart mechanisms. The two ends of the session promptly detect the failure and themselves restart the session via a spare box, without being externally redirected. Enough session state is kept at each end to recover from the glitch.

It appears likely that "optimising" middleboxes are suitable candidates for the soft state approach and for non-real-time data streams, since the consequence of failure of the box is not catastrophic for the user. (Configured HTTP proxies used as caches are an awkward case, as their failure causes client failure.) On the other hand, "functional" middleboxes must be present for the session to continue, so they are candidates for rapid failover or rapid restart mechanisms. We conclude that:

> Middlebox design should include a clear mechanism for dealing with failure.

4.3. Failures at multiple layers

Difficulties occur when middlebox functions occur at different layers, for example the following situation, where B and C are not in the same physical box:

```
Apps layer:     A ------------------------> C ------> D

Lower layer:    A -----> B -------------------------> D
```

When all is well, i.e., there is an IP path from A to B to C to D and both B and C are working, this may appear quite workable. But the failure modes are very challenging. For example, if there is a network failure between C and D, how is B instructed to divert the session to a backup box for C?. Since C and B function at different protocol layers, there is no expectation that they will have coordinated failure recovery mechanisms. Unless this is remedied in some general way, we conclude that

> Middlebox failure recovery mechanisms cannot currently assume they will get any help from other layers, and must have their own means of dealing with failures in other layers.
>
> In the long term future, we should be able to state clearly for each middlebox function what it expects from its environment, and make recommendations about which middlebox functions should be bound together if deployed.

4.4. Multihop application protocols

We can also observe that protocols such as SMTP, UUCP, and NNTP have always worked hop-by-hop, i.e., via multiple middleboxes. Nobody considers this to be an issue or a problem. Difficulties arise when inserting a middlebox in an application protocol stream that was not designed for it. We conclude that:

> New application protocol designs should include explicit mechanisms for the insertion of middleboxes, and should consider the facets identified in Section 2 above as part of the design.

A specific challenge is how to make interactive or real-time applications ride smoothly over middleboxes. This will put particular stress on the failure handling aspects.

4.5. Common features

Given that the IP layer - the neck of the hourglass - is no longer alone in its role supporting end-to-end connectivity, it would be desirable to define requirements and features that are common to middlebox intermediaries. It would then be possible to implement middleboxes, and in particular the protocols that communicate with them, fully from the stance of supporting the end-to-end principle. Conceptually, this would extend the neck of the hourglass upwards to include a set of common features available to all (or many) applications. In the context of middleboxes and multihop protocols, this would require common features addressing at least:

Middlebox discovery and monitoring
Middlebox configuration and control
Call-out
Routing preferences
Failover and restart handling
Security, including mutual authentication

As far as possible, the solutions in these areas being developed in the IETF and W3C should be sufficiently general to cover all types of middlebox; if not, the work will be done several times.

5. Security Considerations

Security risks are specific to each type of middlebox, so little can be said in general. Of course, adding extra boxes in the communication path creates extra points of attack, reduces or eliminates the ability to perform end to end encryption, and complicates trust models and key distribution models. Thus, every middlebox design requires particular attention to security analysis. A few general points can be made:

1. The interference with end-to-end packet transmission by many types of middlebox is a crippling impediment to generalised use of IPSEC in its present form, and also invalidates transport layer security in many scenarios.

2. Middleboxes require us to move definitively from a two-way to an N-way approach to trust relationships and key sharing.

3. The management and configuration mechanisms of middleboxes are a tempting point of attack, and must be strongly defended.

These points suggest that we need a whole new approach to security solutions as the middlebox paradigm ends up being deployed in lots of different technologies, if only to avoid each new technology

designing a end-to-end security solution appropriate to its particular impact on the data stream.

Additionally, content caches and content distribution mechanisms raise the issue of access control for content that is subject to copyright or other rights. Distributed authentication, authorisation and accounting are required.

6. Acknowledgements

Steve Bellovin, Jon Crowcroft, Steve Deering, Patrik Faltstrom, Henning Schulzrinne, and Lixia Zhang all gave valuable feedback on early versions of this document. Rob Austein and Allison Mankin drafted the text on transport relays and TCP spoofers, and Rob Austein made other substantial contributions. Participants in the MIDTAX BOF at the 50th IETF and on the MIDTAX mailing list, including Harald Alverstrand, Stanislav Shalunov, Michael Smirnov, Jeff Parker, Sandy Murphy, David Martin, Phil Neumiller, Eric Travis, Ed Bowen, Sally Floyd, Ian Cooper, Mike Fisk and Eric Fleischman gave invaluable input. Mark Nottingham brought the W3C work to our attention. Melinda Shore suggested using a facet-based categorization. Patrik Faltstrom inspired section 4.3.

7. References

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

[RFC 1928] Leech, M., Ganis, M., Lee, Y., Kuris, R., Koblas, D. and L. Jones, "SOCKS Protocol Version 5", March 1996.

[RFC 1958] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC 2186] Wessels, D. and K. Claffy, "Internet Cache Protocol (ICP), version 2", RFC 2186, September 1997.

[RFC 2475] Blake, S., Black, D., Carlson, M., Davies, E., Wang, Z. and W. Weiss, "An Architecture for Differentiated Service", RFC 2475, December 1998.

[RFC 2543] Handley, M., Schulzrinne, H., Schooler, E. and J. Rosenberg, "SIP: Session Initiation Protocol", RFC 2543, March 1999.

[RFC 2616] 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.

[RFC 2663] Srisuresh, P. and M. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, August 1999.

[RFC 2756] Vixie, P. and D. Wessels, "Hyper Text Caching Protocol (HTCP/0.0)", RFC 2756, January 2000.

[RFC 2766] Tsirtsis, G. and P. Srisuresh, "Network Address Translation - Protocol Translation (NAT-PT)", RFC 2766, February 2000.

[RFC 2775] Carpenter, B., "Internet Transparency", RFC 2775, February 2000.

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

[RFC 2983] Black, D., "Differentiated Services and Tunnels", RFC 2983, October 2000.

[RFC 2993] Hain, T., "Architectural Implications of NAT", RFC 2993, November 2000.

[RFC 3015] Cuervo, F., Greene, N., Rayhan, A., Huitema, C., Rosen, B. and J. Segers, "Megaco Protocol 1.0", RFC 3015, November 2000.

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

[RFC 3027] Holdrege, M. and P. Srisuresh, "Protocol Complications with the IP Network Address Translator", RFC 3027, January 2001.

[CHU] Y. Chu, S. Rao, and H. Zhang, A Case for End System Multicast, SIGMETRICS, June 2000. http://citeseer.nj.nec.com/chu00case.html

[CLARK88] The Design Philosophy of the DARPA Internet Protocols, D.D.Clark, Proc SIGCOMM 88, ACM CCR Vol 18, Number 4, August 1988, pages 106-114 (reprinted in ACM CCR Vol 25, Number 1, January 1995, pages 102-111).

[CLARK95] "Adding Service Discrimination to the Internet", D.D. Clark, Proceedings of the 23rd Annual Telecommunications Policy Research Conference (TPRC), Solomons, MD, October 1995.

[CDNP] M. Day, et al., "A Model for Content Internetworking (CDI)", Work in Progress.

[DSTM] J. Bound, L. Toutain, F. Dupont, O. Medina, H. Afifi, A. Durand, "Dual Stack Transition Mechanism (DSTM)", Work in Progress.

[H323] ITU-T Recommendation H.323: "Packet Based Multimedia Communication Systems".

[HOURG] "Realizing the Information Future: The Internet and Beyond", Computer Science and Telecommunications Board, National Research Council, Washington, D.C., National Academy Press, 1994. However, the "hourglass" metaphor was first used by John Aschenbrenner in 1979, with reference to the ISO Open Systems Interconnection model.

[HTTPSUB] Moore, K., "On the use of HTTP as a Substrate", BCP 56, RFC 3205, February 2002.

[MIDARCH] E. Lear, "A Middlebox Architectural Framework", Work in Progress.

[MIDDISC] E. Lear, "Requirements for Discovering Middleboxes", Work in Progress.

[MIDFRAME] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, A. Rayhan, "Middlebox Communication: Framework and Requirements", Work in Progress.

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

[RSIP] Borella, M., Lo, J., Grabelsky, D. and G. Montenegro, "Realm Specific IP: Framework", RFC 3102, October 2001.

[SALTZER] End-To-End Arguments in System Design, J.H. Saltzer, D.P.Reed, D.D.Clark, ACM TOCS, Vol 2, Number 4, November 1984, pp 277-288.

[SIPFIRE] S. Moyer, D. Marples, S. Tsang, J. Katz, P. Gurung, T. Cheng, A. Dutta, H. Schulzrinne, A. Roychowdhury, "Framework Draft for Networked Appliances Using the Session Initiation Protocol", Work in Progress.

[SOCKS6] Kitamura, H., "A SOCKS-based IPv6/IPv4 Gateway Mechanism", RFC 3089, April 2001.

[TRANS64] "Overview of Transition Techniques for IPv6-only to Talk to IPv4-only Communication", Work in Progress.

[WREC] Cooper, I, Melve, I. and G. Tomlinson, "Internet Web Replication and Caching Taxonomy", RFC 3040, January 2001.

[XMLPI] Intermediaries and XML Protocol, Mark Nottingham, Work in Progress at http://lists.w3.org/Archives/Public/xml-dist-app/2001Mar/0045.html

Authors' Addresses

Brian E. Carpenter
IBM Zurich Research Laboratory
Saeumerstrasse 4 / Postfach
8803 Rueschlikon
Switzerland

EMail: brian@hursley.ibm.com

Scott W. Brim
146 Honness Lane
Ithaca, NY 14850
USA

EMail: sbrim@cisco.com

Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.