The Discrete Internet and What to do About It

Dan Duchamp

Computer Science Department
Stevens Institute of Technology
Hoboken, NJ 07030 USA
djd@cs.stevens-tech.edu

***Abstract*—**

The original model of the Internet is simple and homogeneous: every interface is reachable via a long-lived address, routers do nothing but move packets as fast as possible over hopefully-stable routes, hosts are responsible for securing themselves, and any complicated function is implemented end-to-end by applications running on hosts.

In recent years both network providers and end-site customers have deployed various forms of network intermediaries with accelerating frequency. Examples of intermediaries include NATs, transparent web caches, firewalls, and load balancers.

The result of these deployments is a more complicated "Discrete Internet:" an Internet composed of discrete subnetworks, attached at a small number of choke points, lacking universal addressability, and including stateful in-network intermediaries as vital components of (what were once end-to-end) services.

This development has anguished many who design networks and applications. These critics make sound technical arguments for why the emerging design of the Internet is undesirable, and some argue for a return to the earlier, simpler model.

This paper explains why the original homogeneous Internet model is gone forever, replaced by the Discrete Internet. A novel technical mechanism – a new "session" layer in the Internet protocol suite, inserted between the application and transport layers – is proposed as a way to manage the new complexity, retrieve some of the lost benefits of the original homogeneous model, and open a variety of new possibilities.

I. The Discrete Internet

The Internet is increasingly being populated by intermediaries. For example, located between a browser and a web server there might be a NAT-ing firewall on the client side, one or more web caches at points in the network, and a workload balancer on the server side.

Many lament the rise of intermediaries – especially NATs, but any stateful component – and argue that the Internet can or will return to its original model wherein the network center simply moves IP packets end-to-end between universally addressable hosts. This nostalgic hope usually takes the form of arguing that (1) NATs exist mostly to cope with the IPv4 address shortage, (2) deployment of IPv6 will solve the address shortage, and (3) a return to universal (IPv6) addressability will allow end-to-end use of IPSec. Roughly, IPv6 plus IPSec equals a secure Internet that obeys the original "end-to-end" model.

As desirable as such a future may be, intermediaries are a fact of life, and are likely to become more, not less, popular for a variety of reasons that are technical, political, and operational:

1) End-system deployment of IPv6 will probably never happen for reasons similar to " the tragedy of the commons:" although switching to IPv6 might be the best thing for the world at large, few if any individual parties will benefit tremendously or quickly from the upgrade. Therefore, each party connected to the Internet will separately calculate that its own best interest is to avoid the upgrade and stick with IPv4 – at least until disaster looms.
   NATs are supposedly an important part of handling the IPv4 address shortage. Since IPv4 is likely to be our long-term IP technology, NATs are likely to become more numerous.
2) A network administrator is wise to be conservative, and he/she may perceive security benefits to maintaining a private address space behind a NAT: attackers do not even know which addresses to attack, or which addresses to mention in their attacks against other parties. This is another argument for NATs.[1]

[1] Many say that increased security through NAT-ing is a specious argument, for two reasons. First, address hiding can be accomplished as well with a "filtering router." Second, a sense of security engendered by a NAT-ing firewall may lead administrators to leave hosts behind the NAT/firewall vulnerable.

Similarly, general increased distrust and more bad actors attached to the Internet make firewalls seem attractive (even if other approaches to security might be more sound).

3) Port-translating NATs (NAPTs, the most widely deployed version of NAT) vastly ease renumbering: an organization needs to renumber only its external NAT interface if it switches ISPs or must return addresses to IANA. Even for NATs that are not NAPT, an organization need renumber only the set of IP addresses used on the public side. For similar reasons, NATs simplify the simultaneous use of multiple ISPs.
4) At least for the next several years, there is likely to be more link/network heterogeneity, as wireless service providers and cable companies experiment with different technologies, standards, and services. Heterogeneity leads to more differing protocols (e.g., WAP) and thence to more protocol-conversion gateways.
5) Network providers want to protect links, not just hosts. In order to provide for billing, they want to prevent even the transmission of packets over their networks until after an attached host has authenticated.
6) Corporations are likely to want substantial control over the use of their intranets to limit employee actions (the Web sites they can visit, the data they can access) to prevent goofing off and avoid liability concerns. Such control is more easily implemented at a central network choke point such as a firewall.
7) ISPs and/or companies that control content (e.g., AOL) will naturally prefer the "walled garden" model wherein they receive enhanced revenue in return for enhanced services available only within their own network. A walled garden is more easily implemented by locking out users at a few network choke points than at each service site.
8) Governments are likely to want to "tap" networks in order to enforce laws governing taxation, wiretapping, import/export of certain types of data, etc. One example is that US Federal law requires telephone networks to be tap-able for criminal investigation; this will affect the design of IP telephony.
9) Finally, the equation for end-to-end latency of a packet is:

$$\begin{aligned} latency &= propagation + \\ & \quad transmission + \\ & \quad processing \end{aligned}$$

Where $propagation$ is the time for the first bit of a packet to travel from one end to the other, and is computed as $\frac{distance}{speed-of-light}$; $transmission$ is the time for the rest of the packet to propagate, computed as $\frac{size}{bandwidth}$; and $processing$ is the time spent at routers, including queuing.

Equipment improvements are making the third term tend to zero, while materials improvements are making the second term smaller, meaning that as time passes the first term is becoming an increasingly large portion of the overall latency. Therefore, improvements in service latency will increasingly depend on locating the service closer to the client. This fact and the desire to provide services to clients worldwide argue for deployment of services in the network.

In contrast to the open and flexible "end-to-end" architecture of the original Internet, the goal of many of the mechanisms that create the Discrete Internet is to keep out users, data, and applications that do not have prior approval of the owner/operator of a discrete subnetwork.

II. Consequences

All the trends mentioned above militate toward an Internet that is more discrete: a set of separately managed subnetworks, attached at few points in order to support monitoring and control, intentionally made different in protocols and services, lacking universal addressability, and including stateful intermediaries as vital service components. Many of the trends – separate ownership, owners motivated to compete with each other, corporate and governmental oversight, service latency depending on distance – are fundamental and likely to accelerate, not decelerate. Therefore, the homogeneous model of the pre-commercial Internet is probably gone forever.

The Discrete Internet model poses some deep technical challenges. Any intermediary that holds state on behalf of either endpoint "shares fate" with that endpoint. Making the intermediary effectively part of the application introduces another component[2] that must be configured and secured, that can fail, and that may introduce new failure modes.

Intermediaries that are transparent bring additional complications. They tend to make fault diagnosis more complicated, since new types of errors happen without any notification to endpoints. There is also the philosophical

[2]One that is located away from the end systems, administered by another party whose interests may not agree with those of the parties running the application/end-systems.

question of whether transparent intermediation should be permitted at all: perhaps the endpoint doesn't want the service (or wouldn't want it if it knew it to be there). Transparent intermediation creates the need to place interception/forwarding devices or the intermediaries themselves at key network choke points. This may exacerbate problems of performance, routing efficiency, and fault tolerance: routes must be established so that all or at least the most significant traffic must pass through just a few points, where significant per-packet processing then takes place. Finally, transparency is an inelegant programming model for intermediation: intermediary code must be conscious of IP fragmentation, packet loss, and ICMP; it must inspect and possibly change payload and/or transport headers, sometimes even falsifying basic information such as the source IP address. Such a method of operation, besides being inelegant, makes intermediate services harder to write, thereby constraining the services that developers build or even contemplate.

Besides sharing all the general disadvantages of stateful transparent intermediaries, NATs (including NAPTs) create further disadvantages unique to themselves. RFC 2993 [1] complains bitterly against NATs, mentioning all the above problems plus many more, most notably that NATing undermines the "well known port" concept and breaks IPSec as well as several prominent legacy apps/protocols (FTP, SIP, RSVP, H.323), thereby requiring application-level gateways (ALGs) to run at the NAT.

III. Proposed Solution

We are engaged in research that seeks to make it easier to write, deploy, and manage network intermediaries through a programming model that is advantageous for both endpoints and intermediaries. We propose a new Internet architecture that includes a "session" protocol at layer 5, between application and transport. In this architecture, the possibility of intermediaries is explicitly recognized and provided for. As part of session establishment, the application approves and disapproves intermediaries, and/or specifies their order; no unapproved services are made part of the session. With the application connecting to a session rather than transport protocol, transport protocols are relieved of the burden of end-to-end delivery verification. The session layer verifies end-to-end delivery in the manner requested by the application, and explicitly identifies which payload or header fields should be visible to which intermediaries.

This new end-to-end protocol runs on top of one or more transport connections, which are connected in series between end systems or intermediaries. For instance, if X and Y were the endpoints, and Z were an intermediary, then two transport connections, X-Z and Z-Y, would be used in this mode of operation. The session protocol would ensure delivery between X and Y, and intermediate node Z would benefit from a cleaner programming model: two terminated transport connections.

With transport connections no longer operating end-to-end, they can be left running permanently, carrying the multiplexed traffic of many sessions, either simultaneously or in sequence. At the limit, each pair of communicating nodes might send all traffic over a single long-lived transport connection between them. Alternatively, a pair of nodes might share a set of transport connections, each of which uses a different protocol and/or delivers a different class of service.

The session concept adds an extra level of indirection between communicating endpoints. It has been said that "in computer science, every problem can be solved with an extra level of indirection." Therefore, one would expect significant advantages from adding a new layer to the TCP/IP suite, and indeed that promises to be the case. Potential advantages of adding a end-to-end session layer include:

1) A cleaner programming model for both applications and intermediate services. The intermediary is presented with two terminated transport connections, one in each direction. The application can specify which services it wants or permits, and order them if it so wishes.
2) In the same vein, most current intermediaries are presented with IP datagrams. They must handle IP fragmentation, and many do not do so properly. The "session layer" architecture removes this as an issue. The intermediate service is presented with a terminated transport connection; it sees pure transport payload.
3) Explicitly naming the desired/allowed intermediate services and their locations eases the placement of intermediate devices – data is sent to them directly rather than as a side effect of routing. Therefore, they can be placed anywhere rather than only at choke points. Similarly, physical network design and routing algorithms are freed of one constraint.
4) Implementing a session with several physically short transport connections rather than a single long one suggests several possibilities for performance improvement:
    a) There is a possibility of improved TCP throughput via pipeline parallelism: if data flows from X to Y through Z, then two TCP

sender-receiver pairs (X-Z and Z-Y) operate over separate paths, simultaneously filling two receive buffers. With proper timing, it is possible that the two transport connections could work in parallel for long enough to deliver data to Y faster than would a single transport connection between X and Y. Indeed, such a result has been reported, discovered by accident [2]. We have investigated this possibility through analysis and simulation [3] and shown that, for reasonable loss rates and round trip times (RTTs), throughput can be up to 50% higher.

b) Each physically shorter transport connection should have, on average, shorter RTTs and hence a faster feedback loop. More rapid feedback allows more rapid response to congestion regardless whether congestion is signaled by ack absence or ECN [4]. Furthermore, by performing congestion control over a physically shorter path, the response to congestion is more localized to the specific problem area. This should result in better overall Internet capacity utilization, since areas away from the bottleneck will not have their traffic reduced unnecessarily. Additionally, at least in principle shorter RTTs should exhibit smaller variation, which leads to a more accurate RTO computation [5].

c) The parameters of each individual transport connection can be more precisely tuned to path characteristics. In particular, a larger average MTU can be expected – on average, over all transport connections – because a bottleneck link will limit just one of several consecutive transport connections. Another possibility is to use specialized retransmission, ack handling, etc. [6].

d) There are well-documented benefits to having transport connections live for a long time [7]. Initialization overheads – such as Path MTU discovery, encryption key generation, etc. – can be amortized further. Also, the transport can use long-term aggregated information for congestion control (as suggested for Ensemble-TCP [8] and the Congestion Manager [9], [10]) rather than re-learning it for each (possibly brief) new transport connection.

5) The extra level of indirection can help to hide certain failures such as change of IP address or host crash. For example, if a host changes its address because it moves, it could tear down all running TCP connections, establish new ones using its new address, and resume its layer 5 conversations over the new transport connections.
6) Relieving the transport protocol of the responsibility of verifying end-to-end delivery should permit more research on transport protocols and help end the over-engineering of TCP. In our session layer architecture, transport design can focus solely on data movement issues, engendering a philosophy of limited transport protocols that solve limited problems.
7) Spreading an end-to-end session across several transport connections can ease and accelerate the deployment of new technology that affects the transport layer and/or below, such as ECN and QoS. In a 5-layer world, new transport technology need not run end-to-end, and can be deployed piecemeal.
8) Error codes sent to endpoints from intermediaries might help diagnose faults. Presently, this is not practical because many intermediaries must operate unseen to the application. The layer 5 protocol would allow applications to know of, approve, and manage the types of intermediate services delivered to their flows.
9) It has often been noted (e.g., [11]) that an architectural failing of IPv4 is that an IP address must serve as both the "endpoint identifier" (identifying a host) and as the "locator" (for routing). The overloading of IPv4 addresses complicates mobility models, encourages unsound security practices such as "authentication" based on source IP, and has probably prevented numerous good ideas from even being contemplated. Session layer connection IDs would act as endpoint identifiers, help to clean up the use of TCP/IP by allowing IPv4 addresses to be used purely for routing.
10) Many papers, on disparate topics, introduce the concept of a "session" in order to provide a more intuitive discussion. Our work makes the session notion explicit, so certain session-oriented ideas perhaps could be folded in naturally or rendered unnecessary. Examples include QoS reservation [12] and various application-level session mechanisms such as HTTP cookies [13].

The primary technical pieces of this work are:

1) A sub-protocol for intermediary discovery. The choices are (1) by chance according to intermediary placement and routing and (2) by prior lookup.
2) A sub-protocol for session setup. This is the most

complicated phase. A connection ID must be generated, and it must have properties of uniqueness and non-forgeability. There must be a syntax for specifying which intermediate services are requested, acceptable if inserted without request, and forbidden. It must also be possible to specify order.
Each intermediary must be named and initially configured, including information about which transport protocol and port to use. This phase should also establish which errors are to be reported, to which endpoint(s). Some errors may be intentionally suppressed; e.g., re-establishing a TCP connection after a prior one fails.

3) A sub-protocol for data transmission. The application should be able to specify which parts of its data are encrypted or not, and which parts are checksummed or not. Most importantly, a variety of end-to-end delivery verification methods should be available. Obvious choices include TCP-style consecutive-byte ID, TCP-style SACK blocks, message IDs, and average bit-rate.
4) A sub-protocol for reconfiguration, error reporting, and diagnosis.
5) An API must be specified.

The major challenge in getting this all to work is the overhead of connection setup. The naive way to start a layer-5 session is a *much* more heavyweight operation than starting a TCP connection. However, it is possible that much of the overhead can be removed in the common case by caching transport connections and accepting default configurations of intermediate services. In such a case, it might be possible to establish a session with a single packet source routed from one endpoint to the other, visiting each intermediate service site successively, at the end of which – assuming defaults and no errors along the way – the session is established.

A session-layer protocol is a framework for handling intermediaries, not a "solution" for all issues raised by the presence of intermediaries. This mechanism can ameliorate some of the problems but not the most fundamental ones, such as the existence of a third party in the network and how to handle in-network state. It does not conflict with and can co-exist with the MIDCOM effort, whose goal is to develop a control protocol that permits application intelligence to be located outside the network element that actually captures packets. The MIDCOM protocol, once developed, will allow communication and control between the packet capture element and the remotely located application-aware intermediary.

IV. Previous Work

There are several categories of related prior work.

One category is work that addresses Internet architecture. Clark and Blumenthal [14] discuss how social and political developments more so than technical developments are killing the original end-to-end Internet model. RFC 2775 [15] and a draft RFC by Carpenter and Austein [16] discuss the reduced end-to-end transparency of the modern Internet. The pros and (mostly) cons of a particular type of intermediary, NATs, are examined at length in RFC 2993 [1]. The limitations imposed by NATs are implicitly revealed in RFC 3235 [17], which describes how applications should be written to cope with NATs. RFC 3234 [18] provides a taxonomy of a wide variety of in-network devices (22 types) and evaluates whether and how each may be considered an intermediary, according to 8 different criteria.

Another category is work whose goal and/or mechanism would be subsumed by a session protocol at layer 5. This includes various proposals for TCP endpoint migration and persistent connections [19], [20], [21], [22], [23], [24], [25], [26], work on "ensemble TCP" (methods for sharing path and/or congestion control information among several TCP endpoints at the same host) [7], [8], [27], a mechanism for explicitly routing traffic to intermediaries [2], and various *ad hoc* application-level "session" mechanisms [13], [28].

A third category of prior work is that which fits into the session-layer framework as a specific intermediate service. This includes NATs, firewalls, "performance-enhancing proxies" (PEPs) [6], transport relays, and possibly QoS reservation [12].

A final category is complementary work, specifically that on "middlebox management" – how endpoints might control intermediaries. This work includes earlier mechanisms for "opening pinholes" (SOCKS [29]), using HTTP as a transport [30], Realm-specific IP [31], [32], and the work of the IETF's MIDCOM working group [33], [34], [35], [36].

V. Status

We have completed the cited study on characterization of the performance of cascaded TCP connections [3]. We have also completed a prototype implementation that contains a layer-5 setup sub-protocol that uses cached TCP connections. Further work on design and implementation is continuing as outlined above.

REFERENCES

[1] T. Hain. *Architectural Implications of NAT.* RFC 2993. November 2000.

[2] P. Rodriguez, S. Sibal and O. Spatscheck. *TPOT: Translucent Proxying of TCP.* Computer Communication, 24(2):249-255, 2001.

[3] A. Sundararaj and D. Duchamp. *Analytical Characterization of the Throughput of a Split TCP Connection.* Submitted for publication.

[4] K. Ramakrishnan, S. Floyd, and D. Black. *The Addition of Explicit Congestion Notification (ECN) to IP.* RFC 3168. September 2001.

[5] V. Paxson and M. Allman. *Computing TCP's Retransmission Timer.* RFC 2988. November 2000.

[6] J. Border et al. *Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations.* RFC 3135. June 2001.

[7] H. Balakrishnan et al. *TCP Behavior of a Busy Internet Server: Analysis and Improvements.* in Proc. IEEE INFOCOM, pp. 252-262, March 1998.

[8] L. Eggert, J. Heidemann, and J. Touch. *Effects of Ensemble TCP.* Computer Communication Review, 30(1):15-29, January 2000.

[9] H. Balakrishnan, H.S. Rahul, and S. Seshan. *An Integrated Congestion Management Architecture for Internet Hosts.* in Proc. ACM SIGCOMM '99, pp. 175-187, August 1999.

[10] H. Balakrishnan and S. Seshan. *The Congestion Manager.* RFC 3124. June 2001.

[11] M. Kaat. *Overview of 1999 IAB Network Layer Workshop.* RFC 2956. November 2000.

[12] R. Braden, ed. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification.* RFC 2205. September 1997.

[13] D. Kristol and L. Montulli. *HTTP State Management Mechanism.* RFC 2965. October 2000.

[14] D.D. Clark and M.J. Blumenthal. *Rethinking the Design of the Internet: The End to End Arguments vs. the Brave New World.* in Proc. 28th Telecommunications Policy Research Conference, September 2000. http://www.tprc.org/abstracts00/rethinking.pdf

[15] B. Carpenter. *Internet Transparency.* RFC 2775. February 2000.

[16] B. Carpenter and R. Austein. *Recent Changes in the Architectural Principles of the Internet.* Work in progress. draft-iab-arch-changes-00.txt February 2002.

[17] D. Senie. *Network Address Translator (NAT)-Friendly Application Design Guidelines.* RFC 3235. January 2002.

[18] B. Carpenter and S. Brim. *Middleboxes: Taxonomy and Issues.* RFC 3234. February 2002.

[19] D. Crocker. *Transport Context Names (TCN): Concepts and Facilities.* Work in progress (expired). draft-crocker-tcn-00.txt February 1995.

[20] C. Huitema. *Multi-homed TCP.* Work in progress (expired). draft-huitema-multi-homed-01.txt May 1995.

[21] Y. Zhang and S. Dao. *A "Persistent Connection" Model for Mobile and Distributed Systems.* in Proc. 4th Intl. Conf. on Computer Communication and Networks (ICCCN), September 1995.

[22] X. Qu, J.X. Yu, and R.P. Brent. *A Mobile TCP Socket.* in Proc. Intl. Conf. on Software Engineering, November 1997.

[23] D.A. Maltz and P. Bhagwat. *MSOCKS: An Architecture for Transport Layer Mobility.* in Proc. IEEE INFOCOM, pp. 1037-1045, March 1998.

[24] V. Zandy and B. Miller. *Reliable Network Connections.* To appear in Mobicom 2002.

[25] A.C. Snoeren and H. Balakrishnan. *An End-to-End Approach to Host Mobility.* in Proc. Mobicom 2000, pp. 155-166, August 2000.

[26] A.C. Snoeren, D.G. Andersen, and H. Balakrishnan. *Fine-Grained Failover Using Connection Migration.* in Proc. 3rd USENIX Symp. on Internet Technologies and Systems (USITS), August 2001.

[27] J. Touch. *TCP Control Block Interdependence.* RFC 2140. April 1997.

[28] S. Spero. *Session Control Protocol (SCP).* http://www.w3.org/Protocols/HTTP-NG/http-ng-scp.html

[29] M. Leech et al. *SOCKS Protocol Version 5.* RFC 1928. March 1996.

[30] K. Moore. *On the Use of HTTP as a Substrate.* RFC 3205. February 2002.

[31] M. Borella et al. *Realm Specific IP: Framework.* RFC 3102. October 2001.

[32] M. Borella et al. *Realm Specific IP: Protocol Specification.* RFC 3103. October 2001.

[33] R.P. Swale et al. *Middlebox Communications (midcom) Protocol Requirements.* Work in progress (expired). draft-ietf-midcom-requirements-05.txt November 2001.

[34] P. Srisuresh et al. *Middlebox Communication Architecture and Framework.* Work in progress (expired). draft-ietf-midcom-framework-07.txt February 2002.

[35] M. Barnes, ed. *Middlebox Communications (MIDCOM) Protocol Evaluation.* Work in progress. draft-ietf-midcom-protocol-eval-02.txt June 2002.

[36] J. Rosenberg et al. *STUN – Simple Traversal of UDP Through Network Address Translators.* Work in progress. draft-ietf-midcom-stun-01.txt July 2002.