Peer-to-Peer vs. the Internet: A Discussion of the Proper and Practical Location of Functionality

James P.G. Sterbenz

Lancaster University, Computing Department
InfoLab 21, South Drive, Lancaster LA1-4WA, UK
University of Massachusetts, Department of Computer Science
140 Governors Drive, Amherst, MA 01003-9264, USA
jpgs@sterbenz.org

Abstract. Peer-to-peer information sharing has become one of the dominant Internet applications, measured not only in the number of users, but also in the network bandwidth consumed. Thus, it is reasonable to examine the location of support functionality such as self-organisation, resource discovery, multipoint-to-multipoint group communication, forwarding, and routing, to provide the needed service to applications while optimising resource usage in the network.
This position paper is intended to stimulate discussion in two related areas: First, where *should* functionality to support peer-to-peer applications be located: in the network, or as an application overlay among end systems. Second, where *can* functionality be located, given the practical constraints of the modern Internet including closed systems and middleboxes, as well as administrative, legal, and social issues. We will discuss the performance implications of these decisions, including whether low latency bounds for delay sensitive peer-to-peer applications (such as distributed network computing) can *ever* be achieved in this environment.

Keywords. network architecture, peer-to-peer, client/server, end-to-end arguments, protocol layering, policy, tussle

1 Introduction and Problem

The dominant driver for Internet traffic in the 1990s was the Web, which is a client/server application that benefits from network support, particularly caching. Peer-to-peer information sharing applications have become one of the dominant Internet applications, measured not only in the number of users, but also in the network bandwidth consumed [1,2].

Thus, it is reasonable to consider where peer-to-peer support functionality such as self-organisation, resource discovery, multipoint-to-multipoint communication, forwarding, and routing *should* reside to provide the needed service to applications, while optimising resource usage in the network. It is also reasonable to consider where such functionality *can* reside given practical issues; social, economic, and administrative constraints frequently do not allow desired placement of functionality.

Dagstuhl Seminar Proceedings 04411
Service Management and Self-Organisation in IP-based Networks
http://drops.dagstuhl.de/opus/volltexte/2005/115

1.1 What Does *in* the Network Mean?

The expression "in the network can" mean three things:[1]

Functional – The functional view is based on the *layer* of the protocol stack: physical, link, and network layer (1–3) functions are *in* the network; the session, transport, and application layers (4–7) are not. An example of a function that may be functionally in the network is multicast; it is a native capability of IP with IGMP implemented at layer 3. Application layer multicast is not in the network, being deployed when network layer multicast is not provided in the network or doesn't provide the required service model (such as reliable multicast).

Topological – The topological view has to do with whether or not functions and resources are *co-located* with network nodes (either core or access networks), located with subscriber end systems, or located between at the network edge. For example, locating caches (topologically) in the network reduces response time and can also reduce aggregate bandwidth usage. Topologically in the network does not imply functionally in the network, however. Caching is an application layer service (not functionally in the network), even if a network layer enhancement such as deep packet header filtering with IP address rewriting or active networking is the enabler of this service.

Administrative – The administrative view relates to what entity has ownership and control of a particular function. For example, a network provider may own caches co-located with network nodes (administratively and topologically in the network), or a third party service provider may manage caches (administratively out of the network and topologically either in or out of the network depending on their location). Administrative placement of functionality and services is frequently dictated more by business model than technical concerns.

In the context discussing the location of peer-to-peer functionality, we are primarily concerned with the *functional* distinction, that is whether functionality is embedded in the network at layer 3, or is an application overlay at layer 7. However, note that an overlay established by end users is not in the network by any of these three criteria. In particular, the overlay is *functionally* above layer 3, is *topologically* outside the network on end systems, and *administratively* outside the network because it is owned and managed by end users.

1.2 Peer-to-Peer vs. Client/Server

It is important to remember that most networks were initially designed as peer-to-peer infrastructure, notably the ARPANET that evolved to the Internet, as well as enterprise networks such as DECNET[2]

[1] These three items are quoted with minor changes from [3], which was based ideas initially formulated in [4].

[2] Note that while SNA began as a star-topology host attach, it evolved to a peer-to-peer mesh.

It was only the convergence of client/server hype and the dominance of the Web (which is inherently client/server) that led to a common perception that the Internet was fundamentally client/server.

This problem was exacerbated by myopic service providers that began to engineer their infrastructure (particularly for ADSL and HFC) in an asymmetric manner under the assumption that network traffic was *fundamentally* asymmetric.

When peer-to-peer file sharing emerged as a significant application, network service providers were surprised because the their previous assumptions about network traffic were challenged. Some researchers, however, weren't a bit surprised that a new application changed traffic patterns, even though we didn't know in advance that it would be Napster that would be the killer app of the year 2000.[3]

An important lesson from this is that just as the client/server model wasn't right for everything, neither is the peer-to-peer model.

2 What *Should* be in the Network

Traditional network layer functions consist of addressing, forwarding, routing, signalling, and (optionally) traffic management. Peer-to-peer information sharing require a number of of these support services, in particular:

Resource discovery of information that is to be shared
Multipoint group communication among groups of peers
Routing to establish paths between peers
Forwarding of information along the path route
Signalling among peers in support of the above functions

Peer-to-peer information sharing applications are almost always constructed as application layer overlays, even though they must perform many of the functions traditionally associated with the network layer. So we will first examine what functionality *should* be in the network, guided by the end-to-end arguments.

2.1 The End-to-End Arguments

The end-to-end arguments [5], as originally expressed, guide us on where functions must and should be placed. The first part of the arguments describe what functions *must* be located end-to-end and is paraphrased as:

> Functions required by communicating applications can be correctly and completely implemented only with the knowledge and help of the applications themselves. Providing these functions as features within the network itself is not possible.

[3] Just as quickly Napster has faded due to futile attempts by the RIAA and MPAA to control the direction of the future. But peer-to-peer file sharing has not faded, and is not likely to.

The reason for this is that a concatenation of hop-by-hop functions doesn't compose end-to-end. The canonical examples of such functions are error control and encryption. In the former case, even if link-level reliability is implemented, packets can still be lost (for example due to congestion). In the latter case, even if link-layer encryption is done, cleartext is passed through switches and can be eavesdropped. Therefore these functions *must* be done at the end systems to provide end-to-end reliability and privacy, respectively; implementation in the network is *functionally* redundant.

The second part of the argument, however, gives guidance on which end-to-end functions may *also* be located hop-by-holy for performance reasons:

> It is beneficial to duplicate an end-to-end function hop-by-hop if the result is an overall (end-to-end) improvement in performance.

For example, error control *must* be performed for a reliable delivery service. But in the case of a local-area lossy wireless link that is part of a wide-area end-to-end path, significant performance benefits result from link-layer reliably, so that the majority of retransmissions are in the short control loop rather than end-to-end.

2.2 End-to-End Transparence

Another *distinct* issue is the end-to-end transparence provided by the original ARPANET and early Internet.[4]

For better or worse, end-to-end address transparency is dead with the advent of NATs (network address translators) and middleboxes in the network. Furthermore, while DHCP provides convenient automatic configuration for mobile clients, it is normally imposed on all clients by consumer Internet service providers, preventing the static IP addresses needed by servers and peer-to-peer clients. This is in part so that the service providers don't need to worry about IP address administration (unless a higher service class is charged) and in part to discourage consumers from running servers (recall that asymmetric networks were designed with the assumption that customers would only be clients in a client/server relationship).

This forces users to implement peer-to-peer applications as an overlay (and not call *anything* a server since this violates most consumer ISP terms of service agreements).

2.3 Functional Placement Alternatives

There are number of alternatives should be considered for placement of peer-to-peer functionality in the network; these relate to the earlier discussion of what is in or out of the network.

[4] This transparency, as well as a stateless core for resiliency are part of the original ARPANET design decisions [6,7] and are distinct from the end-to-end arguments; see [3] for a discussion of this vs. the ARPANET design decisions.

Protocol stack layer – application, network overlay, end-to-end transport, network, link, MAC, all subject to requirements dictated by the end-to-end arguments (functionally in or out of the network)
Plane – data, control, or management
Topology – end system, edge node, access network, core network
Control authority – end user, enterprise, third party application service provider, network service provider, switch vendor (administratively in or out of the network)

The significant number of options demands careful analysis to balance the proper choice of implementation, balancing resource utilisation of processing, memory, bandwidth, energy, and latency, as well as between user and network service provider.

3 What *Can* be in the Network

We have described the range of choices to the research community on where functionality should be placed, but this doesn't necessarily result in a corresponding set of choices on where functionality *can* be placed in the network, due to a number of constraints.

3.1 Lessons from Past Network Deployments

First, it is instructive to look at the lessons from past attempts to place functionality in the network to improve the performance and service delivered to users.

Multicast is clearly a case where deployment could result in significant improvements in aggregate bandwidth utilisation. Thus, there is compelling motivation to deploy multicast in the network (both functionally at layer 3 and topologically). However, in spite of over two decades of research in multicast [8,9] and a well codified Internet standard [10] we have yet to see any significant deployment of intradomain multicast, and no ability to multicast interdomain (except for tunneling hacks).
QOS (quality of service) has compelling motivation for a number of delay, bandwidth, and loss sensitive applications. But despite two decades of research and standards development (inlcluding ATM [11,12], IntServ [13,14] and DiffServ [15]), QOS mechanisms have seen almost no deployment in the Internet. Applications still must rely on significantly over-provisioned networks to perform well; voice over IP frequently relies on network infrastructure *completely* segregated from data traffic.
Routing for large network scale (e.g. [16,17]) and policy [18] has been the subject of research for over a decade , but interdomain routing in the Internet is still a serious problem (as evidenced by the multitude of papers published that propose ways to fix the problems with BGP), with no relief in sight.

IPvN (where $N \neq 4$) as well as other attempts to evolve the Internet architecture in a systematic way have failed so far. The most notable case is IPv6 [19] for which the jury is still out (but may be hung). What we do see is a series of hacks attemping to improve security (see below) and to prevent IP address space exhaustion (CIDR [20] and NATs [21]); the latter may well preclude the need for global IPv6 deployment.

Security The Internet was not designed as a secure infrastructure; the problems are well known and have led to significant research over the last decade. However, even attempts to separately secure some of the fundamental control protocols (such as DNS security [22] and secure BGP [23]) have not led to deployments. The only successful deployments of security mechanisms have been at the edge (e.g. firewalls) or end-to-end (e.g. SSL – [24]), outside the network in all three senses: functionally, topologically, and administratively.

This list of some of the more notable failures is sobering, particularly when comparing the research effort to the real-world impacty. Sadly, it seems that the research community has extremely limited influence on the Internet, excepting the multitude of tweaks and hacks to the existing architecture.

A discussion of the reasons for this lack of influence are beyond the scope of this paper, but clearly network service providers must be convinced that such deployments are in their own financial interest. It is important to note that service providers (and switch vendors) are for-profit businesses that have relatively short-term interests and are not often swayed by long-term strategic arguments.

3.2 Lessons from Past Application Deployments

There are also lessons to be learned from previous large-scale application deployments.

The Web was a marvelous creation, and while its current form is a synthesis of older ideas (in particular hypertext [25] and graphical user interfaces), it revolutionised the way that we use the Internet. It was, however, deployed without longer term thought to overall performance and impact on network infrastructure. Minor improvements were introduced by the persistent state in HTTP/1.1 [26], but fundamental improvements are needed to deliver the subsecond response time needed by interactive applications. These problems include the two round trips needed for a single transaction (in the significant number of cases that persistent state is not reusable), as well as the more recent quantity of dynamic content that prevents caching and prefetching. Application designers generally do not pay attention to the systemic optimisations needed for good performance.

Peer-to-peer file sharing applications need functionality that is not provided by current Internet service offerings, including resource discovery and multipoint group communication. Furthermore, users are frequently prevented by ISPs from having stable IP addresses. The response has been to avoid

these problems by implementing peer-to-peer applications as an application overlay oblivious of, and frequently at odds with the underlying network infrastructure. As in the case of the Web, most of these applications are deployed after rapid development, without the application of research principles or systematic evaluation and choice of alternative mechanisms.

Unfortunately, just as the research community has limited impact on the network infrastructure, it seems to have only limited impact on the deployment of applications.

3.3 Overlays and Translucency

Given that peer-to-peer sharing applications *are* implemented as application overlays, we should examine the implications. Overlays provide an extremely useful abstraction for network virtualisation [27,28]. In the case of current peer-to-peer file sharing applications, however, end-to-end overlays are being used for the wrong reason. In this case the overlay is a hack to impelment function that the network can't or won't provide (but arguably should), rather than because these applications have a fundamental need for network virtualisation.

The problem is that opaque overlays obscure the underlying topology and performance characteristics of the network, making it difficult to provide needed service to applications. For example, it is impossible to establish a path with a needed low-latency bound between Boston and New York if the end-to-end path is on IP overlaid on an ATM PVC through Chicago overlaid on a wide-area SONET ring through Dallas, and the application doesn't know this and can't request QOS guarantees from the network.

What is needed is not opacity, but *translucency*, that is the ability to for a overlay to determine the characteristics of the underlay so that it can adapt its operation accordingly. Similarly, the overlay should be able to convey its desired behaviour to the underlay so that it can deliver the needed service model and performance characteristics. This requires *knobs and dials* [5] in which the dials instrument the underlay to the overlay, which can in turn use knobs to influence the underlay behaviour. This is an overlay–underlay control loop that is needed for the same reasons as interlayer awareness control loops.

While this doesn't solve the problem of functionality that isn't in the network but ought to be, it at least provides the possibility for better performance to the application while inducing less load on the network.

3.4 Performance Prospects

So what does this mean for the prospects to deliver performance needed to applications? There is no systemic performance engineering of the Internet as a whole. Instead, all we have are fragmented piecewise optimisations that frequently have

[5] as described in [3] based on ideas presented in [29]

side-effects and do not interact in a manner to benefit *overall* performance. Individual optimisations move problems elsewhere (at best) and frequently make things worse overall. This is in stark contrast to the traditional PSTN where the Bell System and European PTTs carefully engineered the *entire* system (at least in their geographical sphere of influence). While this is not practical for the global internet, a bit less chaos and anarchy in Internet engineering would be a good thing.

Consider the prospects for the two most important performance metrics:

Bandwidth is a performance attribute that can be purchased. That is, given a consumer with sufficient wealth, a service provider can deliver the needed bandwidth by engineering sufficient link and switch capacity. Even when multiple competing network service providers are involved, it is not difficult to concatenate high-bandwidth network paths. That is, users *can* buy pipes that are as *fat* as they can afford.

Latency bounds are much more difficult to achieve. The current Internet architecture has no mechanism for bounding latency (and recall that QOS mechanisms to achieve this haven't been deployed). It is much harder to imagine how applications that need low latency can obtain this service without dedicated links of known topology from a single service provider. Thus users *cannot* buy pipes that are as *short* as they may need.

Note that not all peer-to-peer applications are file sharing. Some emerging applications, such as distributed computing and process control require low delays that are simply not achievable in the current Internet.

Furthermore, peer-to-peer file sharing applications frequently impose far more load on the network than would be the case for careful implementation in the network, sometimes due to the tussle [30] needed to overcome the resistance of the network to such applications.

4 Conclusions

While the tone of this paper is intentionally pessimistic to be provocative, the problems are real. Even though the research community is capable of analysing where peer-to-peer functionality *should* go, for this analysis to have any real impact we must find some way of influencing both the network providers and application developers.

It may be impossible to deploy future delay-sensitive such as distributed computing on the current Internet, but rather we may see a hack of dedicated extranets for the purpose.

References

1. Sprint ATL: IPMON. http://ipmon.sprintlabs.com (2005)
2. Karagiannis, T., Broido, A., Brownlee, N., Claffy, K., Faloutsos, M.: Is P2P Dying or just Hiding? In: *IEEE GLOBECOM*, IEEE (2004)

3. Sterbenz, J.P., Touch, J.D.: High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication. John Wiley, New York (2001)
4. Sterbenz, J.P.: "What Belongs *in* the Network and What Does *in* the Network Mean?". In: *IFIP/IEEE Protocols for High-Speed Networks PfHSN'96*. (1996)
5. Saltzer, J., Reed, D., Clark, D.: "End-to-End Arguments in System Design". In: *Proceedings of IEEE ICDCS 2*, IEEE (1981) 509–512
6. McQuillan, J.M., Walden, D.: "The ARPA Network Design Decisions". *Computer Networks* **1** (1977) 243–289
7. Clark, D.D.: "The Design Philosophy of the DARPA Internet Protocols". In: *SIGCOMM '88: Symposium Proceedings on Communications Architectures and Protocols*, ACM Press (1988) 106–114
8. Kadaba, B.K., Jaffee, J.M.: "Routing to Multiple Destinations in Computer Networks". *IEEE Transactions on Communications* COM-31 (1977) 343–351
9. Deering, S.E.: "Multicast Routing in Internetworks and Extended LANs". In: *SIGCOMM '88: Symposium Proceedings on Communications Architectures and Protocols*, ACM Press (1988) 55–64
10. Deering, S.E.: *Host Extensions for IP Multicasting*. Internet RFC 1112 / STD 005 (1989)
11. Turner, J.S.: "Design of an Integrated Services Packet Network". *IEEE Journal on Selected Areas in Communications* SAC-4 (1986) 1373–1380
12. ATM Forum, ed.: *Traffic Management Specification* Version 4.0. ATM Forum, New York (1996)
13. Zhang, L., Deering, S.E., Estrin, D., Shenkar, S.: "RSVP: a New Resource reSerVation Protocol". *IEEE Network* **9** (1993) 8–18
14. Bradan, R., Clark, D., Shenker, S.: *Integrated Services in the Internet: an Overviewe*. Internet RFC 1633 (1994)
15. Blake, S., Black, D.L., Carlson, M.A., Wang, Z., Weiss, W.: *An Architecture for Differentiated Services*. Internet RFC 2475 (1998)
16. Castineyra, I., Chiappa, N., Steenstrup, M.: *The Nimrod Routing Architecture*. Internet RFC 1992 (1996)
17. Cornely, T., Oster, G., Cherukuri, R., Dykeman, D., eds.: *Private Network–Network Interface Specification* Version 1.1. ATM Forum, New York (2002)
18. Steenstrup, M.: *An Architecture for Inter-Domain Policy Routing*. Internet RFC 1478 (1993)
19. Blake, S., Black, D.L., Carlson, M.A., Wang, Z., Weiss, W.: *An Architecture for Differentiated Services*. Internet RFC 2475 (1998)
20. Rekhter, Y., Li, T.: *An Architecture for IP Address Allocation with CIDR*. Internet RFC 1518 (1993)
21. Srisuresh, P., Egevang, K.: *Traditional IP Network Address Translator*. Internet RFC 3022 (2001)
22. Eastlake, D.E.: *DNS Security Extensions*. Internet RFC 2535 (1999)
23. Kent, S., Lynn, C., Seo, K.: "Secure Border Gateway Protocol (S-BGP)". *IEEE Journal on Selected Areas in Commuincations* **18** (2000) 582–592
24. Freir, A.O., Karlton, P., Kocher, P.C.: *The SSL Protocol* Version 3.0. Internet Draft draft-freier-ssl-version3-02.txt (work in progress) (1996)
25. Bush, V.: "As We May Think". *Atlantic Monthly* **176** (1945) 101–108
26. Fielding, R.T., Gettys, J., Mogul, J.C., Nielsen, H.F., Masinter, L., Leach, P.J., Berners-Lee, T.: *Hypertext Transfer Protocol – HTTP/1.1*. Internet RFC 2616 (1999)
27. Touch, J.: "Dynamic Internet Overlay Deployment and Management Using the X-Bone". *Computer Networks* **3** (2001) 117–135

28. Scott Shenker, L.P., Turner, J.: "Overcoming the Internet Impasse through Virtualization". In: *ACM Hot Topics in Networks (HotNets) III*, ACM Press (2004)
29. Clark, D.D.: "Protocol Design and Performance". In: INFOCOM '95 tutorial notes. (1995)
30. Clark, D.D., Wroclawski, J., Sollins, K.R., Braden, R.: "Tussle in Cyberspace: Defining Tomorrow's Internet". In: *SIGCOMM 2002: Proceedings of the 2002 conference on Applications, technologies, architectures, and protocols for computer communications*, ACM Press (1988) 347–356