Failure to Thrive: QoS and the Culture of Operational Networking

Gregory Bell
Ernest Orlando Lawrence Berkeley National Laboratory
One Cyclotron Road, Bldg. 50E0101
Berkeley, CA 94720
+1 510-486-6817
grbell@lbl.gov

Abstract
Understanding the culture of operational networking can help to illuminate the question of why QoS has floundered. Network administrators have a well-founded aversion to complexity, in part because they experience failures attributable to design complexity on a regular basis. I argue that IP multicast defines a functional limit-case for deployable complexity in today's Internet. That limit is relevant to the deployment of QoS, since many flavors of QoS entail equal or greater complexity.

The notion of a *functional* constraint on complexity draws attention to the economic, historical, and institutional forces which influence the fate of networking technologies. QoS will not be compelling for most network administrators until its design takes account of these forces.

Categories and Subject Descriptors
C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – *packet-switching networks*

C.2.3 [**Computer-Communication Networks**]: Network Operations – *network management*

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks - *Internet*

General Terms
Design, Economics, Reliability, Human Factors.

Keywords
QoS, complexity, multicast, operational networking.

This work was supported by the Director, Office of Science, U.S. Department of Energy under Contract No. DE-AC03-76SF00098.

*ACM SIGCOMM 2003 Workshop*s, August 25&27, 2003, Karlsruhe, Germany

1. Introduction: Failure to Thrive
When a young child who ought to be gaining weight inexplicably does not, pediatricians sometimes resort to a vague diagnosis, the "failure to thrive." This phrase captures something we take for granted about childhood: active thriving is the norm, and anything short is cause for alarm.

Quality of Service (which I'll define as the effort to engineer an end-to-end alternative to best-effort packet delivery on the Internet)[1] has suffered for years from an analogous failure to thrive. But should that be cause for alarm? It is not obvious, after all, that networking protocols are like children – that they thrive in the normal course of events. On the contrary, given the abundance of protocols which have been standardized and the much smaller number in common use, one might reasonably conclude that withering away, rather than thriving, is the normal fate of the average protocol. In the economy of networking standards, failure is the norm, and only "failure to fail" is exceptional.

Seen in this light, the modest success of QoS is not surprising. And yet the failure of QoS to thrive should interest anyone who creates, analyzes, or implements networking standards. In the first place, that failure is significant because of the professional stature of the architects of QoS, and the sheer volume of their work: all told, the literature includes many dozens of articles, Internet Drafts, RFCs, dissertations, and books. It is also significant because the history of QoS sheds light on a structural rift between networking operations and protocol design, the implications of which extend well beyond QoS.

In this paper, I pursue the second point. I argue that understanding something about the culture of operational networking – about the daily experience of network engineers who manage routers and switches – can illuminate the question of

[1]For insight into the difficulty of defining QoS, see [14], especially section 4. I return to the subject of definition later in this paper.

why QoS has floundered. By way of disclaimer, I acknowledge that my professional experience is wholly operational. I work in a scientific research institution where the local-area network contains about 80 subnets and 10,000 connected devices. My background undoubtedly colors the observations I make and the conclusions I advance; in particular, it explains why I have little to say about the lukewarm reception given QoS by Internet Service Providers.[2]

2. Complexity and Failure

The most important operational objection to QoS is excessive complexity. Why complexity *per se* should be objectionable has been the subject of several recent papers [6, 7, 15, 20]. The authors of RFC 3439 have argued, for example, that the complexity of a given architecture is proportional to the number of components it contains, where "component" may be "a protocol path, a software path, or a physical path." They further observe that the scalability of a highly-complex network such as the Internet is impeded by two properties – amplification and coupling – both associated with nonlinear systems theory. The Amplification Principle states that small, local fluctuations can produce large-scale effects, while the Coupling Principle describes the unexpected interactions sometimes observed between seemingly-isolated features and components. Both properties have the potential to generate failures. Although the Internet is "resilient to designed-for uncertainties" [20], it is also prone, as a result of its complexity, to experience "catastrophic events." To cite a memorable phrase, it is "robust, yet fragile" [20].

The tone of this work is suitably cautionary, but it actually underestimates the full impact of design complexity upon the operational stability of networks, because it tacitly assumes that complex protocols function *as they were designed to function*. The authors conceive of frailty, in other words, as the unintended consequence of otherwise well-behaved systems interacting.

Yet I am certain that most people who administer switches and routers on a daily basis would attribute the frailty they encounter in networks – and they encounter plenty – to poorly-implemented software, rather than unexpected non-linearity.[3] It is the recurring experience of network engineers that complex protocols (especially protocols for which the constituency is small, such as MSDP) are prone to break. Those of us who work with routers encounter serious bugs on a regular basis; we report these bugs to vendors; we upgrade code; and the cycle repeats. As a result, we eventually come to presume that complex protocols will destabilize our networks: we anticipate failure. Outsiders may misinterpret that expectation as simple pessimism, but in fact it is a form of hard-won working knowledge.

It is difficult to sympathize with the operational mindset I have described without living through dozens of debugging sessions. As a poor substitute, I want to describe the flavor of several failures we have encountered at LBNL recently. They fall roughly into two categories. On the one hand, there are isolated problems which seem to strike randomly; on the other hand, there are problems which fall into recurring patterns.

Initially, the following incident seemed to fit squarely within the first category. Abruptly one day, all the subnets served by Router A lost connectivity with destinations outside of LBNL:

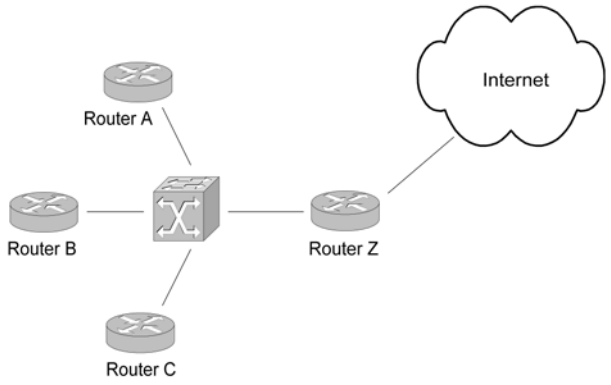

Figure 1: simplified topology

Soon afterwards, subnets on Routers B lost connectivity as well, followed by subnets on Router C. All hosts internal to LBNL could communicate with each other, though. In the process of debugging, we looked carefully at the BGP state on the border Router Z, but there were no signs of trouble. The symptoms strongly suggested a routing problem, but all routing tables were normal. Eventually we discovered the cause of the failure. As unlikely as it seems, the ARP process on Router Z had broken in such a way that the router failed to answer address resolution requests for its "inside" interface address. When the relevant ARP cache entries on Routers A, B and C timed out, each lost the ability to forward packets to its next-hop address for external destinations.[4]

[2] On the other hand, I believe that story is relatively well-documented compared to the corresponding resistance in the enterprise. See [5], for example.

[3] Bug-related failures could be seen as instances of the Amplification Principle, since the input (*eg*, a particular bit pattern in a packet) is trivial compared to the outcome (a router crash). My point, though, is that current discussions of complexity in networks don't conceive of such failures as *systemic*, and therefore as important constraints on deployable complexity. On the insufficiency of current "mental models of the Internet's important properties," see [12].

[4] OSPF state was unaffected by the failure, since the mappings between layer-3 and layer-2 multicast addresses are static. In this narrow sense, IP multicast is more failure-resistant than IP unicast; but see below.

For some time, we attributed this failure to a buggy implementation of the Address Resolution Protocol. When the same symptoms returned two months later, though, we were able to pinpoint the root cause of the ARP failure, which turned out to be a route processor crash triggered by a bug in the code which handles multicast packets. At that point, the failure took on a different character entirely; it fit into one of the recurring patterns I mentioned above. In the preceding twelve months, we had experienced a series of bug-related multicast failures so consistent it could almost be budgeted for in terms of FTE support. We had diagnosed at least 10 serious bugs on 5 hardware platforms, and the time required to resolve these issues was on the order of engineer-weeks.

These bugs were not mere annoyances. On the contrary, they affected router stability. We saw the following symptoms, and many more:

- Router repeatedly reboots when it encounters multicast traffic
- Router reboots when attempting to establish MSDP peering
- Normal PIM packets cause a memory leak
- MSDP router fails to advertise active local sources
- Switch intermittently drops all multicast packets

Such incidents linger stubbornly in the memory of those who debug them. They contribute to an expectation that *every* networking protocol is encumbered with unknown failure modes. Eventually, we concluded that implementing IP multicast required a substantial commitment of engineering resources, and that it would inevitably impair the stability of unicast routing due to the need for frequent OS upgrades and intrusive testing. Many enterprises do not have the mandate, or see a justification, for making this commitment.[5]

I believe our experience with IP multicast is directly relevant to the history of QoS, because IP multicast defines a limit-case for deployable complexity in today's Internet. It is important to stress that the limit is functional. Multicast is not intrinsically "too complex" to be implemented reliably, in other words – but that is not the issue. The issue is whether multicast can be implemented reliably given the extra-technological factors that constrain the success of actual deployments. These factors are numerous, and they include:

- inadequate quality assurance by vendors
- lack of a critical customer mass and a limited market
- finite staff time for troubleshooting
- scarcity of debugging tools
- limited skill-set of operational engineers
- lack of trust between neighboring domains

If IP multicast represents a limit-case for deployable complexity, the implications for QoS are serious. Intuitively, it seems clear that the Integrated Services architecture – with its requirement that routers (even core routers) keep end-to-end per-flow state, and that reservation set-up "be fundamentally designed for a multicast environment" [4] – presupposes a level of complexity significantly higher than that of inter-domain multicast routing. The vexing challenges of multicast apply to Integrated Services, but there is an additional requirement to manage (and inevitably debug) packet scheduling, admission control, classification, and reservation setup.

The relative complexity of the Differentiated Services architecture is harder to assess, largely because of its greater flexibility. Diffserv aims to achieve scalability by "aggregating traffic classification state" through "IP-layer packet marking" [3], and it can be implemented on a very modest scale, perhaps "at a single bottleneck" [8]. Such minimalism is a far cry from the Grand Unified style of QoS exemplified by Integrated Services, and it also presents fewer operational risks.[6] However, it is doubtful that minimalist Diffserv can provide the rich service model envisioned by most QoS advocates and architects. As Geoff Huston points out, the simpler architecture is still incomplete in many respects, especially in regard to "end-to-end signaling facilities" for functions such as resource availability discovery and service requests [14; also see 2 {section 4.2.7}]. Diffserv signaling protocols – which include, in one proposal, a network of "brokers" for end-to-end bandwidth allocation [16] – add significant complexity to the bare, minimalist model, as does the prospect of enlarging the Diffserv domain from a single router to a densely-interconnected set of network clouds under different administrative control.

My aim is not to make a definitive judgment about which of these architectures is more complex, but to suggest that forces currently impeding the scalability of IP multicast routing (including a complicated control plane, a shortage of knowledgeable operational staff, and a scarcity of debugging tools) also impede the deployment of QoS. To be sure, recent QoS architectures are simpler [9, 18], but this reduction in complexity may ultimately be compromised by the necessity to make QoS secure against denial-of-service attacks, as well as resource "theft" [1, 11, 14].

The notion of a *functional* constraint on complexity tends to draw attention to the economic, historical, and institutional forces that influence the fate of networking technologies. These forces – which include the cost of router interfaces, the market for wide-area connectivity, the rate of broadband penetration, the demand for real-time applications, and the average skill-set of network administrators – receive scant attention in the literature on QoS. I think it's fair to say that protocol designers have not been very attentive to them generally. This neglect is particularly ironic, because the professional life of anyone affiliated with computer networking in the past decade has been buffeted by economic and

[5]For more on this point, see [10].

[6]Implementing Diffserv on a limited scale, though, does carry the danger of relocating the offending bottleneck to another link immediately outside the QoS domain [5].

historical surprises: the explosive growth of HTTP traffic, for instance, or the rise and fall of the dot-com bubble.

Attempting to architect QoS without taking into account its economic and institutional context is roughly analogous to designing a city without reference to local culture, climate or geography. In either case, the likely result is a conspicuous failure to thrive. One might argue that architectural design work should remain untainted by such concerns, and to a certain extent this view is appropriate and salutary. But carried to an extreme, it can lead to the standardization of overly-complex technologies with little chance of deployment in working networks.

What would it mean to think about QoS in relation to the forces I have described? Again, let's consider the culture of operational networking, which eschews complexity and anticipates failure. When evaluating complex services, network engineers ask practical questions, and a full understanding of the fate of QoS requires us to take these questions seriously:

1) What does my network (which is currently stable and comfortably-provisioned) have to gain from enabling this technology?

2) Can it be incrementally deployed?

3) Are there good debugging tools?

4) Can I debug it without impacting best-effort service?

5) While debugging, do I need the active cooperation of engineers from other domains?

6) Are the benefits sufficiently compelling to compensate for the potential pain?

7) When it breaks, will I be blamed?

8) Am I likely to be caught in the middle of disputes regarding whose packets receive better-than-best-effort service?

9) Will I be asked to investigate very transient, vaguely-defined symptoms which users attribute to the failure of QoS?

10) Will this project become a black hole for my time, or that of my staff?

QoS will not become compelling for the majority of network administrators until it can offer persuasive answers to these questions. And it will not develop persuasive answers to these questions until its design addresses the operational and economic forces likely to impede its deployment. One such force, for example, is continuing downward pressure on the price of bandwidth, in local and wide-area networks alike.

3. Throwing Bandwidth

The primary operational response to the problems which QoS aims to alleviate – including latency, jitter and packet loss – has been to provision links with sufficient headroom that congestion becomes unlikely. This practice is sometimes characterized as "throwing bandwidth at the problem," but the formulation is misleading, because it implies that the strategy is thoughtless. In fact, the tactic of "throwing bandwidth" has more merit and more staying power than advocates of QoS have been willing to acknowledge.

At LBNL, we adhere to a "10% rule" for provisioning. When the utilization of a router or switch interface begins to exceed 10% of the link speed over a 30-minute average, we upgrade the link. This rule of thumb is, in part, an outgrowth of the observation that our monitoring systems do not tell us very much about transient, peak utilization: we know by comparing actual packet traces with MRTG data that even 30-second load averages underestimate the magnitude of bursts.

The "10% rule" is simple, and it works well in practice. Whether it is economical depends on the market for Ethernet interfaces at the moment the 10% boundary is crossed: are we forced to become early adopters, or can we take advantage of commodity pricing? In practice, the 10% rule generally has *not* committed us to the bleeding edge. For example, we do not anticipate needing to deploy 10 Gigabit Ethernet in production networks for at least nine months, and interface prices are rapidly falling. When all costs are carefully accounted for, we believe that "throwing bandwidth at the problem" is the cheapest antidote to congestion in our network, and that "throwing protocols at the problem" will compromise stability. [7]

Recently, network researchers have begun to explore how to achieve some of the tantalizing benefits promised by QoS through strategic provisioning alone. In other words, they have begun to ask a more formal version of the question "how much bandwidth should we throw at the problem?" [13, 19]. At very high link speeds, that number turns out to be surprisingly low – on the order of 15% above average utilization, by one account [13]. Furthermore, service providers are reporting success in adhering to a similar philosophy in the Internet core [1, 15]. This approach may have been inspired by the dawning realization that the rate of Internet traffic growth has been overestimated in recent years [17]. Whether it can succeed in the long run depends on actual rates of traffic growth, and on rates of decrease in the cost of transmission capacity. Once again, economic and market forces wholly unconnected with the *architecture* of QoS are central to its operational fate.

The tactic described above – *ie,* provisioning conservatively to accommodate bursts – has sometimes received only perfunctory treatment by the architects of QoS:

> "Bandwidth will be Infinite"
>
> The incredibly large carrying capacity of an optical fiber leads some to conclude that in the future bandwidth will be so abundant, ubiquitous, and cheap that there will be no communication delays other than the speed of light, and therefore there

[7] Natrually, ISPs face different economic constraints than enterprises when provisioning link capacity. For a brief overview of current norms in ISP capacity engineering, see [1].

```
will be no need to reserve resources.
However, we believe that this will be
impossible in the short  term and unlikely
in the medium term.  While raw bandwidth
may seem inexpensive, bandwidth provided
as a network service is not likely to
become so cheap that wasting it will be
the most cost-effective design principle.
Even if low-cost bandwidth does eventually
become commonly available, we do not
accept that it will be available
"everywhere" in the Internet.  Unless we
provide for the possibility of dealing
with congested links, then real-time
services will simply be precluded in those
cases.  We find that restriction
unacceptable. [4]
```

This passage from RFC 1633 begins by caricaturing the argument it wants to refute. In fact, no one believes that bandwidth will be infinite; but at any given time, it may be sufficiently plentiful and inexpensive that it creates an attractive alternative to QoS. The authors assert that bandwidth "is not likely to become so cheap that wasting it will be the most cost-effective design principle" – but this claim cannot be persuasive in the absence of data about actual prices and predicted demand. In addition, the authors appear to assume that unused bandwidth is "wasted" in the same sense that uneaten food is wasted, but of course that is not the case. As Meyer and Bush have observed, in "modern Internet backbones the unused capacity is actually protection capacity" for various low-probability events (fiber cuts, extended power outages, equipment failures) [7; also see 1].

The final sentences of the paragraph are quite revealing. It is true that *some* links in the Internet will always be congested, but asserting that "we" need to "provide for the possibility of dealing with congested links" in such cases undermines the justification for a complex, end-to-end QoS architecture, as described in the body of the RFC. In other words, defining Quality of Service as a technique for coping with the possibility of *isolated* congestion reduces the scope of the concept substantially. We are left with something resembling minimalist Diffserv, which falls short of providing the rich service model originally envisioned for QoS.

4. Conclusion

Remarkable intelligence and energy have been lavished upon the architectural design of QoS, but much less attention has been devoted to careful analysis of the relevant problem space from an operational or economic perspective. This discrepancy is symptomatic of a broken (or attenuated) feedback loop between network operations and research. Ideally, there would be constant exchange of information between these two domains, but in practice they tend to be mutually insular. The number of people who are comfortable in both worlds is relatively small, as is the number of institutions designed to bridge the gulf.

This rift has harmed the process of protocol design by shielding it from information about the daily experience of failure in large-scale networks – information that is crucially important in trying to estimate what I have called the functional limits of deployable complexity. Unless the architecture of QoS is calibrated with these limits in mind, it will almost certainly continue to suffer from a chronic failure to thrive.

5. Acknowledgements

For generous feedback on earlier drafts of this paper, I am grateful to Mike Bennett, Jim Leighton, Sally Floyd, and Ion Stoica, as well as the anonymous reviewers. I owe a special debt to Ted Sopher, a steadfast advocate of design simplicity at LBNL, and the origin of many ideas and principles I have articulated here.

6. References

[1] R. Atkinson. An ISP Reality Check. Presentation to IEprep, IETF 55, Nov 2002. http://www.ietf.cnri.reston.va.us/proceedings/02nov/219.htm#slides.

[2] G. Armitage. Quality of Service in IP Networks. Macmillan Technical Publishing, Indianapolis IN, 2000.

[3] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475, An Architecture for Differentiated Services.

[4] R. Braden, D. Clark, and S. Shenker. RFC 1633, Integrated Services in the Internet Architecture: an Overview.

[5] J. Burrescia and J. Leighton. DOE NGI Testbed: Project Close Out Report. http://www.es.net/hypertext/welcome/ pr/DOE_NGI_TESTBED_REPORT3_Dec01.pdf

[6] R. Bush. Complexity – the Internet and the Telco Philosophies, a Somewhat Heretical View. NANOG 29, Oct 2002. http://www.psg.com/~randy/021028.nanog-complex.pdf

[7] R. Bush and D. Meyer. RFC 3439, Some Internet Architectural Guidelines and Philosophy.

[8] B. Carpenter and K. Nichols. Differentiated Services in the Internet. IBM Research Report. http://domino.watson.ibm.com/library/Cyberdig.nsf/398c93678b87a12d8525656200797aca/8efe46a5de7e3b8f85256b5e003c5883?OpenDocument

[9] N. Christin and J. Liebeherr. A QoS Architecture for Quantitative Service Differentiation. IEEE Communications, 41(6), June 2003.

[10] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment Issues for the IP Multicast Service and Architecture. IEEE Network magazine special issue on

Multicasting, 14(1):78-88, January/February 2000. ftp://ftp.sprintlabs.com/diot/xcast-deployment.zip

[11] S. Floyd and R. Atkinson, eds. Concerns with Network Research Funding. Presentation to the IETF 56 plenary. http://www.ietf.org/proceedings/03mar/slides/plenary-19/index.html

[12] S. Floyd and E. Kohler. Internet Research Needs Better Models. First Workshop on Hop Topics in Networks. http://www.icir.org/ models/hotnetsFinal.pdf

[13] C. Fraleigh, F. Tobagi, and C. Diot. Provisioning IP Backbone Networks to Support Latency Sensitive Traffic. http://ipmon.sprintlabs.com/pubs_trs/pubs/cdiot/infocom03_provisionbb.pdf

[14] G. Huston. RFC 2990, Next Steps for the IP QoS Architecture.

[15] D. Meyer. Some Thoughts on QoS and Backbone Networks. Presentation to IEprep, IETF 55, Nov 2002. http://www.ietf.org/proceedings/02nov/slides/ieprep-4.pdf.

[16] K. Nichols, V. Jacobson, and L. Zhang. RFC 2638, A Two-bit Differentiated Services Architecture for the Internet.

[17] A. Odlyzko. Internet Growth: Myth and Reality, Use and Abuse. http://www.dtc.umn.edu/~odlyzko/doc/internet.growth.myth.pdf

[18] I. Stoica. Stateless Core: A Scalable Approach for Quality of Service in the Internet. Ph.D. Dissertation, Carnegie Mellon University, Dec. 2000.

[19] T. Telkamp. Traffic Characteristics and Network Planning. NANOG 26, Oct 2002. http://www.nanog.org/mtg-0210/ppt/telkamp.pdf

[20] W. Willinger and J. Doyle. Robustness and the Internet: Design and Evolution. http://netlab.caltech.edu/pub/papers/part1_vers4.pdf

[21] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. IEEE Network, vol. 7, Sept 1993.