The League of SuperNets

Ken Birman • *Cornell University*

The Internet revolution is over, the debris has been swept away, and we've supposedly recovered from the hangover. Yet the economy remains mired in a listless cycle of anemic recoveries and shallow downturns. The tech sector is soft, and unemployment remains high.

Nonetheless, we can reignite the fires. In this column, I propose a major effort to transform the Internet into a league of SuperNets. Doing so offers the promise of revolutionary new business opportunities for companies large and small, and could save the emerging Web services technology area from a snarl of reliability and security problems. We can slash the costs of operating big networks, roll out new kinds of applications with real-time properties, and start to build other kinds of applications for purposes like controlling the restructured electrical power grid or managing military assets on a battlefield — applications that the Internet just wasn't designed to tackle.

Unfortunately, however, the following proposal departs drastically from the way that the Internet is currently evolving. The technical side of the issue is likely to be the easy part; the daunting problem centers on the politics of the Internet sector and the community that controls its future. Yet, the payoff could be so great that I want to argue for a community response. If we all get behind a common vision, we can make it a reality.

The Shot Heard Round the Web

What would it take to start a new dot-com boom? Viewed almost a decade after the fact, it seems clear that the first boom was triggered by much more than the emergence of Web browsers. Personally, I've always viewed Windows 95's introduction as the watershed event, and not just because it triggered a 25-fold run-up of Microsoft share prices—the first taste of irrational exuberance. The real significance was that literally everyone upgraded to Windows 95, overnight. Microsoft created a new business model, simultaneously opening a major new market for PC software and gaining a tremendous jolt of revenue. It suddenly became clear that even large companies needn't settle for slow market growth. With the right product and story, a company could earn hundreds of billions of dollars overnight.

The Windows 95 rollout became a template from which the subsequent dot-com boom was cut. As the Web caught on, just about every vendor you could name was holding all-hands meetings to strategize about the next new thing. What stands out, in retrospect, is the degree to which the Web played directly into this emerging market dynamic. Suddenly, vendor after vendor rolled out a new Web-enabled upgrade for every product line, and in a flash of Internet time, these new solutions transformed the market.

I believe this was the real core of the dot-com event: a technology shift exciting enough to get everyone to spend money on IT infrastructure and upgrades — not just on new products, but on old ones, too. I'm no economist, but this is the insight that motivates the remainder of this column — a technical proposal that is ultimately aimed at restarting the stalled dot-com revolution, although perhaps without the sky-high share prices.

The Stuff of which Revolutions Are Made

To start the next revolution, a technology must be more than just a good idea or something that would make the world a better place. We need a technology with something for everyone in it because we need more than an incremental shift.

If you wanted to place a bet right now, the horses in this race would have names like Web services, grid computing, and autonomic computing. Some might argue for a new wave of lightweight, wireless embedded sensors instead, and not everyone has given up on peer-to-peer (P2P) technologies. Do any of them have a chance?

Web Services

For those who haven't followed object-oriented computing, Web services are the latest in a continuing revolution. Objects have promoted a tremendous productivity boom because they facilitate code reuse and simplify the task of integrating new applications with older systems, but existing object systems don't go far enough. Web services promise to take a massive step forward by standardizing object interactions around the technologies that were so successful for Web browsers. If we can really pull this off, the implications will be staggering. Could Web services be the next new thing?

Grid Computing

Create a hypothetical blend of SETI@Home and Web services, and you get a new generation of loosely coupled, massively scalable computing systems scattered over the Web. Indeed, IBM is gambling on grid computing's success. Google the term and you'll find numerous articles like the one about a British hospital that farms out 3D nuclear magnetic resonance (NMR) reconstructions to a remote computing grid for use during brain surgery.[1] But the article doesn't explain how the system deals with crashes or infrastructure security, which could present the some problems.

You might argue that this omission really points to a broader problem because many Web services will need security, high availability, automated self-management, and so forth. Moreover, it isn't at all clear that the world really needs massive computing cycles. If just a few applications use the Grid, it might turn out to be a great idea, but not quite what we need.

Autonomic Computing

The world is already a tangle of computer systems talking to one another, and proposals like Web services and grid computing could take interdependence to a whole new level. That's a scary prospect because the Web just isn't very reliable or secure; it's so fragile that if you just wait a little while, it usually freezes up in some arcane way that takes a human expert to untangle. This is no big deal for the people using Web browsers who can always get a cup of coffee or buy that book from someone else. But how's a computer supposed to deal with unexpected outages or bizarre responses? Autonomic computing is intended to make the network smart, and to enable a new kind of distributed system that makes sense of its own state and patches itself when problems arise. Yet, much of the misbehavior stems directly from the Internet itself, so it's hard to see how autonomic computing can fix the real problem. Lacking a good story here, it seems all too likely that Web services systems will be unpredictably balky, jeopardizing the brain surgery application, not to mention many less ambitious services on which corporations, if not critically ill patients, will depend.

Embedded Sensors

Embedding lots of sensors in the environment seems like a good idea, except that it isn't clear what we would want to do with them. The devices are cheap, but we'll need to scatter billions of them to trigger a revolution. That might even be all right, except that they really don't do much — at least not yet; the technology base is immature, and batteries run low rather quickly. So, while I'm excited about the technology area, I don't see it revolutionizing anything soon.

P2P Computing

P2P protocols are breaking all sorts of scalability barriers (scalable event-notification systems and indexing, for example), but few users have well-defined business requirements that span tens of thousands of computers. The old ways of building systems work pretty well for the usual settings with a few hundred computers. I guess that Web services could create a new generation of systems in which tens or hundreds of thousands of computers are dependent on new kinds of services, forcing both servers and clients to monitor the overall system state in real time. One major airline, for example, is running 250,000 PCs off a set of Web services hosted at a single data center in Atlanta. If applications of that size become more common, we'll need P2P technologies to keep things running. Nonetheless, I don't see P2P as the revolution's trigger — just a tool we'll need. (My research group believes that when this happens, Cornell's scalable P2P solutions — Astrolabe, Bimodal Multicast, Kelips — will be exactly what's needed to solve the problem. See www.cs.cornell.edu/Info/Projects/Spinglass/pubs.html.)

The Promise of Web Services

Industry's bet is on Web services. As an example of where things are headed, suppose your favorite online store were to adopt a Web services model. Today, people need browsers to manually surf to a site to purchase items. Tomorrow, Web services could potentially let any computing application, built by any third party, tap into that online store's systems. In effect, their middle-tier and back-end technologies would suddenly become accessible through Web-service interfaces talking to thousands of new applications, built mostly by third parties. Those applications could then sell to users with a single click — the full power of online shopping without the risk of trying to create the next Amazon.com.

Meanwhile, the online store can also offer all sorts of back-end business services: business logic, supply-chain solutions, logistical planning, financial services, inventory, you name it. In fact, it can even offer plug-ins to let third-party developers access these services — applets that are completely analogous to those used to make Web pages extensible. The back-end company thus sees huge growth in its market, and the small software developer who needs a way to help a doctor's office reorder supplies wins, too.

This is just a glimpse of one corner of the world that Web services could enable. There are already hundreds of ideas out there, many as potentially revolutionary as the one I just outlined. Indeed, analysts predict that Web services-based systems will transact as much as US$3 trillion in annual commerce by the end of the decade. Of course, most of this money will be the actual computer-to-computer transactions, but plenty will also be spent on infrastructure to support this commerce. We're looking at tens and perhaps hundreds of billions of dollars a year in expenditures on servers, software platforms, and consulting services.

These are the kinds of numbers that could revive the economy, get our old friends back to work, and really turn things around. But here's the catch: while Web services really do have the promise to finally get everything talking to everything else, the technology will inherit most of the problems we're having with the Web.

Web services systems lack a way to provide high availability, and they need to offer really strong transactional integrity guarantees. Securing the whole infrastructure becomes a real concern, right from the Internet up. Connectivity, bandwidth, and latency are at the Internet's mercy, and this is a serious "gotcha" because not many of the high-availability, high-security, high-integrity technologies we understand best have ever gotten the commercial traction needed to become part of the Web services standards. In fact, the World Wide Web Consortium's (W3C) efforts have been fairly conservative, focusing on best-of-breed Web and database technologies, and are likely to continue that way for many years. W3C is betting that Web services need to run on the *current* Internet, and this is tying their hands.

If we limit ourselves to engineering above the existing Internet, we need to live with the limitations of an infrastructure ideally suited to transferring email, moving files, and handling noncritical Web browsing. The Internet doesn't support quality of service (QoS) guarantees and probably can't. The infrastructure folds under denial-of-service (DoS) attacks, and even minor problems can provoke extended outages. Let's face it: if we use this as the foundation, we will end up with systems that don't work much better than Web browsing. Moreover, each time a condition arises where your Web browser reports that a remote system is unresponsive, or gives a stale response, a Web services computer might experience an unrecoverable error.

We're left with something of a conundrum: Web services really could do the trick if we could make them work reliably, but they won't have the right technical characteristics. To reap real benefits, organizations need to use these technologies pervasively; yet taking that step with Web services would leave their most critical operations dependent on computer-to-computer interactions that were constantly at risk of wedging in ways that require human intervention to untangle. Worse still, once Web services get out there, we will almost certainly hit the scalability issues already seen in large data centers, but not yet obvious to the average enterprise.

We're hitting a wall associated with the end-to-end reliability and security model, which long ago became religion for the IETF communities.

The bottom line is that none of these technologies has much chance of starting a new revolution. The true promise of Web services will probably go unfulfilled, and the real reason — the core problem — is that we're hitting the limitations of the Internet, itself.

Hitting the Wall

For decades, the Cassandra's of the field have predicted the Internet's demise. They've warned that we'll soon run out of IP addresses (fortunately, network address translators came along just in time), that routers couldn't keep up (saved by fast-prefix routing algorithms), or that network routing instabilities would do us in (the guys at University of Michigan, Ann Arbor, saved the day on that one by showing that these were mostly caused by bugs in a couple of router implementations). Today we hear the same thing about email spam and DoS attacks.

Well, I'm an optimist. I don't think any of these problems is really going to get us. My concerns are deeper: I think we're hitting a wall associated with the end-to-end reliability and security model, which long ago became religion for the IETF communities that guide the network's evolution. I also think the problem can be fixed.

The network has changed enormously over the past few decades if you look at it on a large scale. But the change has enshrined the early Internet's most basic assumptions:

- the network is about best-effort reliability, not guarantees,
- it should provide a single route from point A to point B (if that route congests, the user can wait), and
- users can be trusted not to attack the network.

The network is, resolutely, a black box: users are given intermittent connectivity and erratic bandwidth, and told to do their best with it. The problem is the basic model, which hasn't changed in years. It tells us that connectivity isn't necessarily bilateral — I might be able to talk to you, but you might be unable to respond — and failure detection is essentially impossible. Any kind of "consistent" failure detection is deemed way out of scope for the network. Got an opinion about the best way to route your packets? Well, keep it to yourself: the Internet isn't interested in your input. Need something stronger than IPSec? Tough.

Web services confront this Internet with a new generation of applications that might require continuous connectivity (hence, redundant path-independent routing), a high degree of infrastructure security, and the means to defend against DoS attacks. We need to anticipate a new wave of protocols that can't behave in a TCP-friendly manner, and that will make heavy use of multicast to disseminate event notifications and update cached copies of data needed for rapid response. These elements all run directly contrary to the prevailing mindset and existing technology base. What can we do?

Overlay Networks

Nearly everyone has experienced an end-to-end overlay network — we call them "virtual private networks." Basically, you take a chunk of the network and superimpose some other "virtual" network on the same platform. MIT has even started to superimpose a more reactive routing infrastructure on the network to overcome some of the problems I've cited. Many people are starting to believe that overlay networks — including the MIT variety, called resilient overlay networks (RON) — could be the answer. The issue is that when we overlay something on the current Internet, the raw links on which the overlay operates are subject to the same problems we're trying to work around. Moreover, if everyone actually started to use RONs, or a similar technology, the underlying protocol would probably break. But hold that thought because we'll revisit this question in a moment.

QoS Mechanisms

An obvious rejoinder is to point to DiffServ or other QoS mechanisms like RSVP, but I don't think these would help either. They suffer from a similar problem to the overlay networks. Routers disturb the dynamics of a packet flow because they don't know what the user is trying to do, and they see traffic mixed together from many sources. By the time packets traverse a router, they have lost some of the flow properties the sender was trying to offer.

Suppose that A contracts with the network to send 10 packets per second to B and then starts sending a packet precisely every 100 ms. A's data should get through, right? Wrong. Pass that flow through one router, and the inter-packet spacing starts to vary: some packets will be delayed a bit while others zip through with no delay at all. This erratic spacing means the data stream is no longer matched to A's original contract with the network under which DiffServ and RSVP did their planning. Run the stream through a chain of routers, and the original clean spacing will be greatly disrupted. The further we get from A, the more remote the packet stream's dynamics are likely to be from what A contracted to send.

Consider the world as it looks to a router far downstream, which promised to set aside resources for 10 packets per second. Now A seems to be abusing the deal by sending a burst of perhaps 30 packets over a 250-ms period followed by dead time for the next 2.75 seconds. Can we blame the router for tossing out all but two or three of those packets? Of course not — so QoS bites the dust.

The bottom line is that the current Internet just can't support the sorts of QoS properties the new generation of mission-critical applications needs. If we build on a weak foundation, we'll never end up with the kind of rock-solid structures that global corporations can pin their survival on — not to mention ones that rural hospitals can bet your life on.

I haven't even touched on security, but if you ask Steve Kent (BBN Technologies) or Gene Spafford (CERT), they'll assure you that the Internet isn't about to provide the kind of serious security that could brush spammers and hackers off the stage. Corporate users might want to exploit Web services, but the Internet will never let them do it.

Time to Start Fresh /ok?/

Basically, I think it's time to reinvent the Internet — to replace it with a SuperNet that can go where the Internet has never gone before, and isn't likely to go in the future. But now we run into the political problem I mentioned earlier. Suppose that we, as a community, could speak with a single voice to urge the powers that be to take such a step. Even if they were to concede that the idea has some appeal, there are several reasons a pragmatic observer would conclude that it just isn't possible now or anytime in the foreseeable future. For one thing, any new vision of the network still needs to somehow reuse the billions of dollars in existing infrastructure because it just isn't plausible to replace more than a fraction of this hardware at a time. For another, demand for the existing Internet isn't about to go away.

Fortunately, there might be a way out of the dilemma. Let's go back and think about how the current Internet is really implemented. Who provides the wiring? The telecom's, of course, operate tremendous amounts of fiber and set some aside to support the network backbone (many are also regional ISPs, especially for broadband connectivity).

Down at this core level, routers often partition incoming data in a coarse-grained way: data from MCI, data from AT&T, and so on. This lets telecom's implement bilateral deals, in which MCI leases bandwidth to AT&T, for example, and AT&T leases bandwidth to France Telcom. Thus, a form of bandwidth sharing is already in place, partitioning the physical telecommunications network between forms of traffic (voice versus data) and partitioning bandwidth between major vendors. Similarly, large backbone networks have some limited partitioning ability.

I suggest that we leverage this kind of partitioning to split the existing physical infrastructure into a small number of side-by-side virtual networks, each with a share of the original network's total capacity. One of those virtual networks could run the normal Internet protocols, but the others could run modified protocols — whether just slightly or more ambitiously altered — to obtain completely new properties. These new networks could go far beyond the current Internet, precisely because they wouldn't be constrained to use the current generation of Internet protocols.

In effect, we could easily exploit the existing bandwidth-reservation architecture to overlay a small number of networks on shared wiring, and even on shared routers. Much as we time-share modern computers, we could implement time-shared router policies. The potential exists to transform the existing "single" Internet, running on the existing "single" telecommunications network, into a league of SuperNets.

This, I believe, might actually be feasible. We wouldn't need to discard the existing physical network, or impose a drastic change on existing applications. We would simply set aside some of the physical network's current bandwidth for dedicated use by the SuperNets. The current Internet itself would live on, side-by-side with the new ones.

Why SuperNets?

Having set the Internet itself to the side, we won't be forced to run the usual routing policies, or security policies, on the remaining network overlays. Given dedicated, set-aside resources, there are all sorts of options for building mesh-style routing that would guarantee redundant paths between source and destination, hence offering applications much stronger end-to-end properties. This essay is too short to explore the details, but a world in which we can build new networks from the ground up and run them side by side on the existing hardware and links would be a very exciting place.[2]

Given raw, dedicated capacity, we can build SuperNets with better QoS properties (steady real-time data delivery, for example), guaranteed availability (even if individual links or routers fail), or rock-solid security. SuperNets could include mechanisms to explicitly tell applications what to expect in terms of latency, available bandwidth, and what sorts of jitter will be apparent. The scalable P2P technologies my group and others have worked on could endow some SuperNets with other characteristics. Cornell's Astrolabe could let us build a system for large-scale monitoring, management, and control, for example, permitting the development of autonomic control systems for Web services, and solving the data-mining needs of the world's homeland security departments — not to mention those of large corporations seeking to improve their efficiency. Ion Stoica, at University of California, Berkeley, would probably want to implement a SuperNet running his Internet indirection infrastructure (i3), a radical new way of implementing networking over a P2P indexing layer (see http://i3.cs.berkeley.edu/publications/). MIT could unify RON with DiffServ and build a media network that might really work. The armed forces could run a militarily secure overlay (of course, they probably won't tell us about it).

Open the door to innovation at that low level, and we'll see a boom of new kinds of higher-level networks, coexisting on an infrastructure that is currently reputed to be overflowing with black (unused) fiber and not making enough money for the operators.

These overlays will cost real money, but the question of who'll pay to use them is clear: because companies won't get what they need by running Web services on the Internet, a massive wave of demand is about to emerge for Web services hosted on other platforms. Companies will surely pay to get the benefits these powerful technologies will bring.

To develop and test the solutions in the first place, I advocate a bit of government investment to create a kind of entrepreneurial incubator. Researchers could apply to use a small number of government-funded infrastructure networks; after a suitable maturation time, each SuperNet would either grow up and go its own way as a revenue-earning proposition, or the resource would go back into a competitive research pool. In effect, governmental research organizationsshould give clever new SuperNet ideas like i3 a chance, but once they get off the ground, let them make it or fail on their own as for-profit services.The real measure of success will be whether these technologies can attract paying customers.

In the end, we need to reexamine the whole stack because the innovation mustn't stop down in the routers. A SuperNet could offer standard Internet services, but it could also replace or supplant them with its own — by including network-level mechanisms for monitoring component status and reporting failures, for example. Such mechanisms could be wired into TCP, RPC, and other protocols to ensure consistent, trustworthy reporting when a connection fails. We might offer network topology services, designed to report the network's structure and relate it to the real world. We also need accurate, trustworthy time synchronization and protocols that can provide temporal guarantees for applications transmitting media. Each SuperNet could be its own world, catering to the unique needs of a major class of real applications.

Rekindling the Flame

Building a league of SuperNets is definitely feasible. Of course, those with the largest vested interest in the current Internet might drag their feet at first, but they'll jump in once they realize that SuperNets could be the answer to their revenue woes.

It seems to me that a major, deliberate effort to create a league of SuperNets could relaunch the stalled distributed computing revolution. Give us even one SuperNet and we can break through the barriers for Web services. With a few SuperNets, autonomic computing could become a reality, and interactive Web services commonplace. We'll deliver TV-quality video and radio-quality audio, without the dropouts, and all sorts of cooperative workplace tools will follow. We'll be able to design high-availability mechanisms that really work and scalable event mechanisms where "real time" actu ally means something.

It's time to face reality. We've gotten stuck in the moment and can't get out, and this is why so many of our friends and students are under- or unemployed. And the reason we're stuck is that the Internet is being asked to do things it just can't do — things directly at odds with its core design decisions. There is a path forward, however, and it could bring back the dot-com revolution with a vengeance. Let's dust off the dot-com entrepreneurs. It's time to transform the Internet into a league of SuperNets.

References

1. L. Versweyveld, "3D Visualization in the Operating Room to Improve Accuracy of Tumor Removal," *Virtual Medical Worlds Monthly*, Oct. 2002; www.hoise.com/vmw/02/articles/vmw/LV-VM-10-02-2.html.
2. K. Birman, "Technology Requirements for Virtual Overlay Networks," *IEEE Systems*, vol. 31, no 4, July 2001, pp 319-327.

Ken Birman is a professor of computer science at Cornell University. His research interests include reliable distributed computing, reliable information collection or dissemination, and security in complex distributed systems. Birman received a PhD in computer science from University of California, Berkeley. He has authored several books and many papers on reliable distributed computing. Contact him at ken@cs.cornell.edu.