Van Jacobson: Content-Centric Networking

Charles Severance

Content-centric networking is much more than caching of content—it also works well for live streams of popular data

As engineers, every once in a while, we need to seriously revisit our underlying assumptions and make sure they still hold true. In terms of the best architecture for the Internet, the four-layer model based on TCP/IP is pretty much accepted as absolute and unquestionable truth.

I recently interviewed Van Jacobson of PARC, and we talked of a major re-architecting of the Internet to deal with the fact that it is increasingly a global content distribution system layered atop a communication model, with computers making virtual "long-distance phone calls" to each other. What if they took a more content-centric approach?

Visit www.computer.org/computingconversations to view a video of this interview.

THE TELEPHONE MODEL

To delve into a possible alternate future, we must first let go of the notion that the current state-of-the-Internet architecture is "right" simply because we're using it and it seems to work:

> If you look at how the Internet has evolved, it started as a telephone system for computers. People wanted their computers to exchange data, and the model we had for communication for 140 years had been the phone system. We said, 'Okay, communication is conversation over long distances,' so let's make protocols and infrastructure that allow computers to have a conversation. The first cut of that was the ARPANET—a network that would handle different bandwidths and didn't require the global clock distribution of a telephony network. Instead, it substituted buffering.

The earliest telephone systems used the dialing of a phone number to physically configure relays and create a temporary "physical" wire that could transmit amplified analog audio signals over long distances:

> A crucial thing coming out of Paul Baran was to not emulate the phone system, where communication was all about building a wire hop-by-hop or link-by-link between two end points, basically instructing the switching system how to make one long wire. Instead, Paul said, 'Just identify the end points and let the network take care of getting the data there.'

Although the ARPANET and the Internet were very different from the telephone network in their implementation and use of physical wires, they were similar in that the ultimate goal of TCP/IP was to allow two distinct applications to "call" each other, get a connection, and let those applications have a conversation. This conversational model between applications was very general and allowed rich research into many different kinds of uses for the Internet. It kept the four-layer architecture pristine and avoided embedding application-specific understanding in routers:

> This really changed the world, but the bulk of that change didn't happen in the 1970s, 80s, or early 90s, when the Net was first growing out—it happened in the late 1990s and 2000s, when the Web took off. The Web had nothing whatsoever to do with computers having a conversation model. It had to do with people creating and consuming content. The Web showed us for the first time what happens if we leave behind this 18th century model of telephony between applications, stop looking at the wires, and instead focus on the information in the wires.

0018-9162/13/$31.00 © 2013 IEEE Published by the IEEE Computer Society

The Web gave us a way to name information, and representational state transfer (REST)-style Web services gave us a namespace for data as content. Today, we enter a URL into our browser to indicate "what we want" instead of "how to get it." We layer the HTTP protocol atop a conversational model, but as moving data and content becomes the dominant use of Internet infrastructure, perhaps the conversational model is becoming our limiting factor:

> We're having massive scaling problems today trying to join together the very information-centric Web model with the very host-centric TCP/IP model. Look at what it takes to build something like YouTube. You can create videos and put them on your own website, but you have to pray that they never become popular. If they get popular, your ISP will almost immediately shut you down because your link will be completely saturated, in what we call the "Slashdot effect." That problem is intrinsic to the conversational model: we don't do broadcast television by making phone calls to the television station because there is no way to scale that. We broadcast it out to everybody who wants to listen, and we don't know who they are--they aren't individually identified.

WORLD SCALE

As world-scale applications and services became the norm on the Internet, it was impossible to have a single connection from something like Google to the rest of the Internet and route all traffic from around the world to a single network connection to a single server room in Mountain View, California. For Google to function effectively, it needs many facilities around the world and many connections between Google facilities and the public Internet.

TCP/IP is architected around the notion that the network number portion of an IP address connects to one and exactly one router in the global Internet. To support world-scale applications, it was increasingly necessary to "lie" to TCP/IP about what's really happening:

> If you look at YouTube, Google, Amazon, Facebook, Twitter, all of these very heavily used services that manifest themselves to the Net as an IP address that looks like a single location, if they're a single location with hundreds of millions of users, the traffic in a conversational model always scales like the popularity, it scales like the number of consumers. If you're doing a Twitter update or creating a video that will be seen by millions, you can't deploy it in a pure conversational model, and so you're forced to spend all your time fooling the Net.

Content-centric networking is much more than caching content—it also works well for live streams of popular data.

In a world-scale Web application like Google or Facebook, the destination IP address in a packet that has little to do with your request's ultimate destination. The address is simply the quickest way to get your request into a nearby datacenter, where Google looks more deeply into the request to figure out what you want so it can virtually route it to the closest copy of the requested information:

> Information that's qualitatively the same is spread randomly across the whole packet. We have source and destination addresses that we conventionally think of as IP, so it's at the front of the packet to be used by the network layer, and then we have ports that are TCP, so they're a little bit deeper in because they're supposed to be used by the end node for its de-multiplexing to get you to a particular application. Inside that, we have sequence numbers that are used when you get to the application to reassemble the larger unit of information, and inside that, we have URLs, which are used by a higher-level part of the application for session meaning and the like. You just have all this information, and it's all fundamentally name information that indicates "What do these bits mean?" If you pull together the source addresses, the ports, the sequence numbers, and the URLs, they all give you context. They're all the "name" of the information.
>
> What if I just said that packets have a name on the front and all that information gets collected to the name? At any point in the network, you look at the name to do your job. If just the front part will work because all you're doing is gross-level steering, just look at the front. If you need to look at more of it, we don't have layers, we have a set of structured information, and we know that we'll be looking at different parts of it for different reasons.

CONTENT IS KING

Once we switch to a naming scheme that uniquely identifies each packet or segment of content, it no longer matters where the data actually comes from. The content segment could come as easily from a nearby router as from the originating source. Packets could then be cached throughout the network in the memory of the routers, and those packets could be reused for popular content.

Content-centric networking is much more than caching of

content—it also works well for live streams of popular data. One of the many prototype applications built on top of CCN's early implementations is a multiuser/multipoint video conferencing system. As the popularity of a live video conference or event scales, the likelihood of packet reuse increases dramatically:

> If you don't care where you're getting the data—if all that matters to you is what the data is, not where it comes from—then all of this memory that has to be in the network as buffering in order to manage the multiplexing suddenly becomes a viable source of data. Having the load scale with popularity is strictly a function of the fact that the data can only originate from one place. If you just care about the data, just start going toward that place, and as soon as you run into that data, now you have a copy and now you're done with your distribution.

Of course, it's one thing to postulate that we need a fundamental paradigm shift in the architecture and yet another to move a new approach into broad world-wide production. Like the shift from voice to data communications from the 1960s to the 1990s, shifting data communications from a conversational model to a content-centric one will also take time and require many experiments. And like the engineering of TCP/IP itself, there will likely be many versions as researchers identify new issues and use cases.

The good news is that an active community is exploring CCN and its applications worldwide. The CCN community met in Sophia Antipolis, France, in September 2012, with 29 presentations, nine demonstrations, 16 poster sessions, and attendees from many different organizations. You can find more information on CCN and related projects at www.ccnx.org.

Perhaps the next time you have a little free time while you wait for a YouTube video to buffer, you might let your mind wander and imagine for a moment how you might re-engineer the Internet architecture to better handle our increasingly content-oriented use of the network.

***Charles Severance**, Computing Conversations column editor and* Computer*'s multimedia editor, is a clinical associate professor and teaches in the School of Information at the University of Michigan. Follow him on Twitter @drchuck or contact him at csev@umich.edu.*