Restoring the Reputation of the Much-Maligned TCP

IAB Workshop on Stack Evolution in a Middlebox Internet (SEMI)

26th – 27th January 2015, ETH Zürich, Switzerland

Stuart Cheshire, Apple, Inc.
Ideas expressed here are solely the opinions of the author

At the 79th IETF, in November 2010 in Beijing, I was introduced to the concept of NAT64+DNS64, on the IETF's experimental NAT64 Wi-Fi network. I tried it using on my iPhone, and the results exceeded even my most optimistic expectations. Initially, things did not look good. Every single app I tried, from email to web browsing to stock quotes to weather forecasts to on-line shopping to airline information, displayed an error alert saying, "You have no Internet connection." This was quite disappointing. However, I then discovered something surprising. In every case, once I dismissed the oh-so-helpful alert, the app then proceeded to work fine, using native IPv6 on the iPhone to talk via the NAT64+DNS64 gateway to IPv4 services.

This was extremely encouraging. Success was within our grasp. All we had to do was to get every app to remove the pointless code that displayed the stupid "You have no Internet connection" alert, and we'd be in great shape for the future.

Fast forward to 2015. Where are we now? Current data appears to show that we now have a *smaller* proportion of iOS apps that are IPv6-compatible than we did four years ago.

How did that happen?

Like many things, this one is certainly a combination of factors. Lack of easy access to a NAT64+DNS64 network for testing meant that many app developers, while fully supportive of IPv6 in principle, had no easy way to test their apps on an IPv6 network to make sure they worked.

But another important factor has been the move towards using custom protocols running over UDP. On Apple's iOS there are good address-family-agnostic APIs for applications using HTTP or TCP for connections. For applications using UDP, not so much. Developers are more on their own, relying on gethostbyname() and sendto(). And, being more on their own, without access to a NAT64+DNS64 network for testing, more of these app developers end up writing code that's not IPv6-compatible.

What has caused this trend? I think that two decades of SIGCOMM papers detailing every tiny shortcoming of TCP (usually with a proposed solution), while individually valid and useful in the academic community, collectively have served to smear the reputation of TCP in many minds.

For example, it's widely known that, "When loss occurs that's not a result of congestion, TCP's response of slowing down may be undesirable." Experienced networking people know that 99.9% of the time loss *is* a result of congestion, so 99.9% of the time TCP's response *is* the right thing to do, and it may not even be possible to reliably identify the other 0.1% anyway. However, that subtle analysis is often lost, and "TCP Does The Right Thing After All" doesn't make for a winning SIGCOMM paper title. As a result, the lesson many casual observers take away often ends up simplified to, "When loss occurs TCP does the wrong thing."

Simply put, in many minds, TCP has too many flaws to count. It's an unmitigated disaster.

Consequently many are encouraged to conclude that any protocol they design themselves on top of UDP will definitely work better than TCP. Well, there are precisely zero academic conference papers detailing flaws in their protocol, so that means it must have none, right?

To reverse this trend, we need to revitalize development of TCP. Features like TCP Fast Open and Tail Loss Probe need to get deployed. Without these new features, legitimate concerns about things like the latency induced by the three-way handshake will encourage people to design their own application-specific protocols. Frequently, those application-specific protocols, while allegedly superior to TCP in the one specific area that the developer was giving their attention to, will be inferior to TCP in every other way. The lack of IPv6-compatible implementations and APIs for these application-specific protocols is one example of the many ways they fall short of TCP.