The Internet Protocol Journal

December 2014 Volume 17, Number 2

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at: **www.protocoljournal.org**

ISSN 1944-1134

From The Editor

As announced in September, *The Internet Protocol Journal* (IPJ) has been re-launched with the generous support of numerous organizations and individuals. You will find a complete list of our sponsors and supporters on page 47. As we bring you this second issue, I want to say a few words about subscriptions to this journal. If you are already a subscriber to IPJ and in the past have received a printed copy by mail, you will find your Subscription ID printed on the back page along with your delivery address. We will send you an account activation e-mail in the near future that will allow you to update and renew your subscription. However, if you have changed e-mail address you can contact us with the new information by sending a message to **ipj@protocoljournal.org**. If, on the other hand, you picked up a copy of IPJ at a conference or other event and you wish to create a *new* subscription, just follow the instructions on our website at **www.protocoljournal.org**, where you will also find all our back issues, index files, and sponsorship information. Subscriptions to IPJ are free of charge.

The growth of the Internet has been a recurring theme in this journal since its inception in 1998. We've covered various aspects of IPv4 address-space depletion and IPv6 deployment, and of course explored many challenges related to *Network Address Translation* (NAT). But address depletion isn't the only scaling issue facing the Internet. In this issue, Geoff Huston explains what happened in August 2014 when the number of reachable networks as announced by the *Border Gateway Protocol* (BGP) exceeded 512,000 for a time.

The Internet is also growing in a different arena, namely that of intelligent embedded systems that use Internet protocols for communication. This emerging technology area is referred to as *The Internet of Things*. In our second article, Douglas Comer describes the *ZigBee IP Protocol Stack,* which is under development and standardization.

As always, we would love your feedback on anything you read in this journal. With your permission we can include your comments in the form of a Letter to the Editor, or you may consider writing a Book Review. Send your message to **ipj@protocoljournal.org**

—Ole J. Jacobsen, Editor and Publisher
ole@protocoljournal.org

What's So Special About 512?

by Geoff Huston, APNIC

August 12, 2014 was widely reported as a day when the Internet "collapsed." Despite the sensational media reports, the condition was not fatal for the Internet, and perhaps it could be more reasonably reported that some parts of the Internet were having a "bad hair day." The root cause of this event was the Internet routing system, and in this article I will review the behavior of this system and the relationship between the routing system and routing hardware.

A Lightning Introduction to the Internet Routing System

The Internet is a collection of some 50,000 component networks. Some are large, spanning multiple continents, while others are the size of a small office. Most of these networks sit somewhere between these two extremes. All networks announce those IP addresses (or "network address prefixes") that are located within their network (or "originate" from their network).

The routing protocol used to disseminate these announcements across the Internet is the *Border Gateway Protocol* (BGP)[0]. To simplify the operation of BGP, these announcements of reachable network address prefixes are not made to each of the 50,000 other networks, but instead are made to their immediately connected adjacent network neighbors (or routing "peers"). If one (or more) of these peer networks is willing and capable of passing traffic through its network to reach the originating network (that is, act as "transit"), then these networks will further announce these network address prefixes to their peers, and so on. In this way a BGP speaker will hear, via its immediately connected peers, announcements for all reachable network address prefixes, and can assemble a complete picture of all reachable addresses on the Internet.

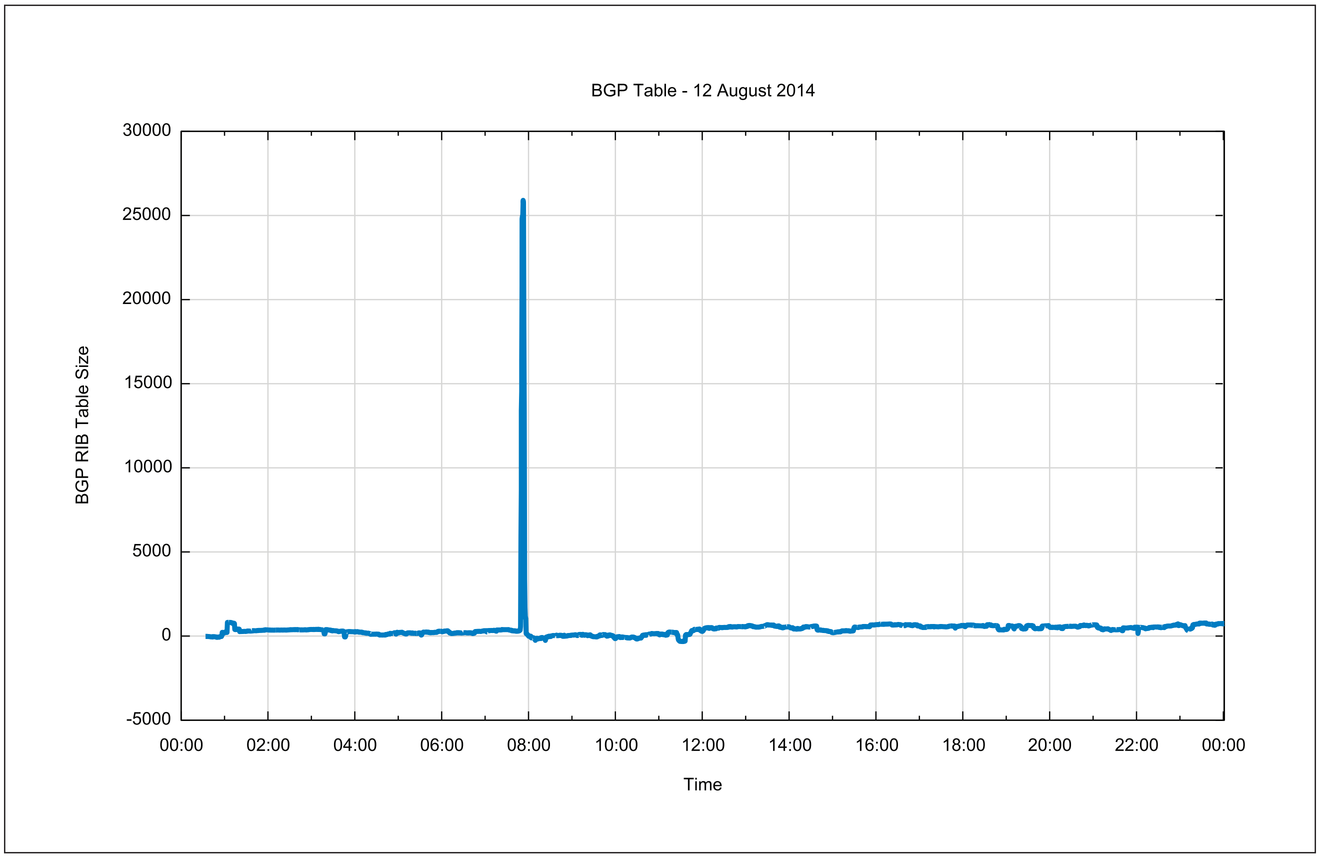
When a network can no longer reach a network address prefix, it needs to inform its peers that it is no longer a useful transit to reach these addresses. The network informs its peers by passing a *withdraw* message to them, withdrawing the network address prefix that it can no longer reach. When a BGP speaker receives a withdraw message, it checks to see if any other peers are still announcing reachability to that address. If so, it adjusts its internal record of the preferred path to reach those addresses to be via a peer that has not withdrawn its path to those addresses, and updates its peers with this new path. If no alternate path to the address exists, the BGP speaker marks those addresses as unreachable and passes a withdrawal message to its BGP peers in turn.

Given that there are some 520,000 network address prefixes and 50,000 component networks, this whole process sounds extremely chatty in terms of protocol interaction. However, BGP uses the *Transmission Control Protocol* (TCP) as its transport protocol, and because TCP provides reliable carriage services, BGP does not repeat its announcements or withdrawals. A BGP protocol session carries only the changes that occur in reachability to network address prefixes. Secondly, BGP uses internal timers to damp the frequency of updates sent to each peer, so each BGP speaker waits for its peers to stabilize before propagating further reachability changes for each network address prefix. The result is surprising stability most of the time. But, from time to time, exceptional events occur.

August 12, 2014

Analysis of BGP activity on August 12, 2014, in terms of the net change in size of the routing table from midnight of that day, shows that something certainly happened just before 08:00 *Coordinated Universal Time* (UTC) (Figure 1).

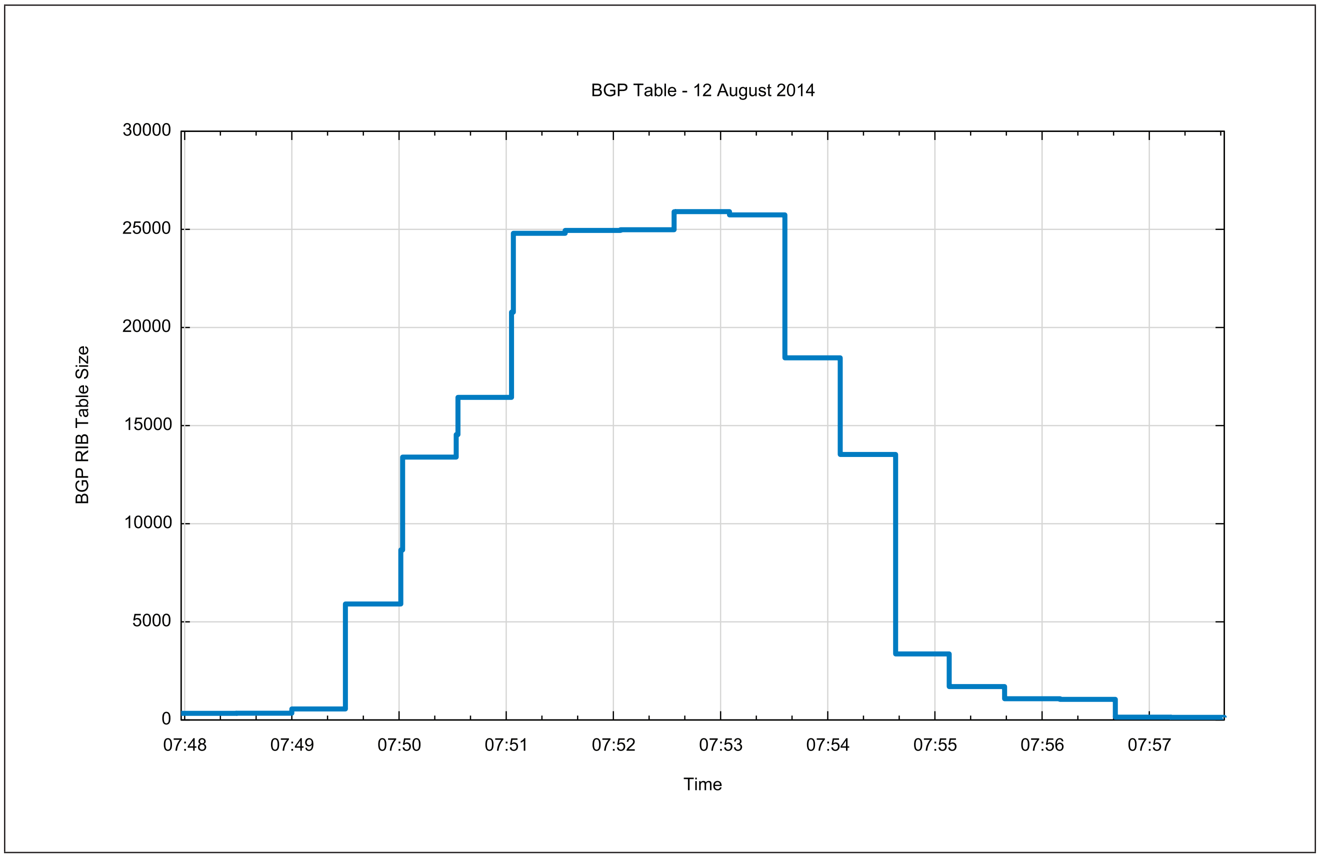
Figure 1: BGP Table Size for August 12, 2014



What dominates this picture is the spike that occurred a few minutes before 08:00 UTC on that day, when the Internet was flooded with what appears from the graph to be 26,000 new prefixes, which were withdrawn very rapidly thereafter. All these new routes shared a common origin, *Autonomous System 701* (AS 701). They did not convey any change in routing information, because they were announcements of more specific prefixes of already announced network prefixes. The announcements were short-lived, and were withdrawn soon after their announcement. The most likely explanation of this event was a *route leak,* where routing detail that was internal to this network was inadvertently leaked into the larger inter-AS routing space, either as a result of a filter reset or a BGP community tag failure, or other forms of transient failure within the route control apparatus of the network.

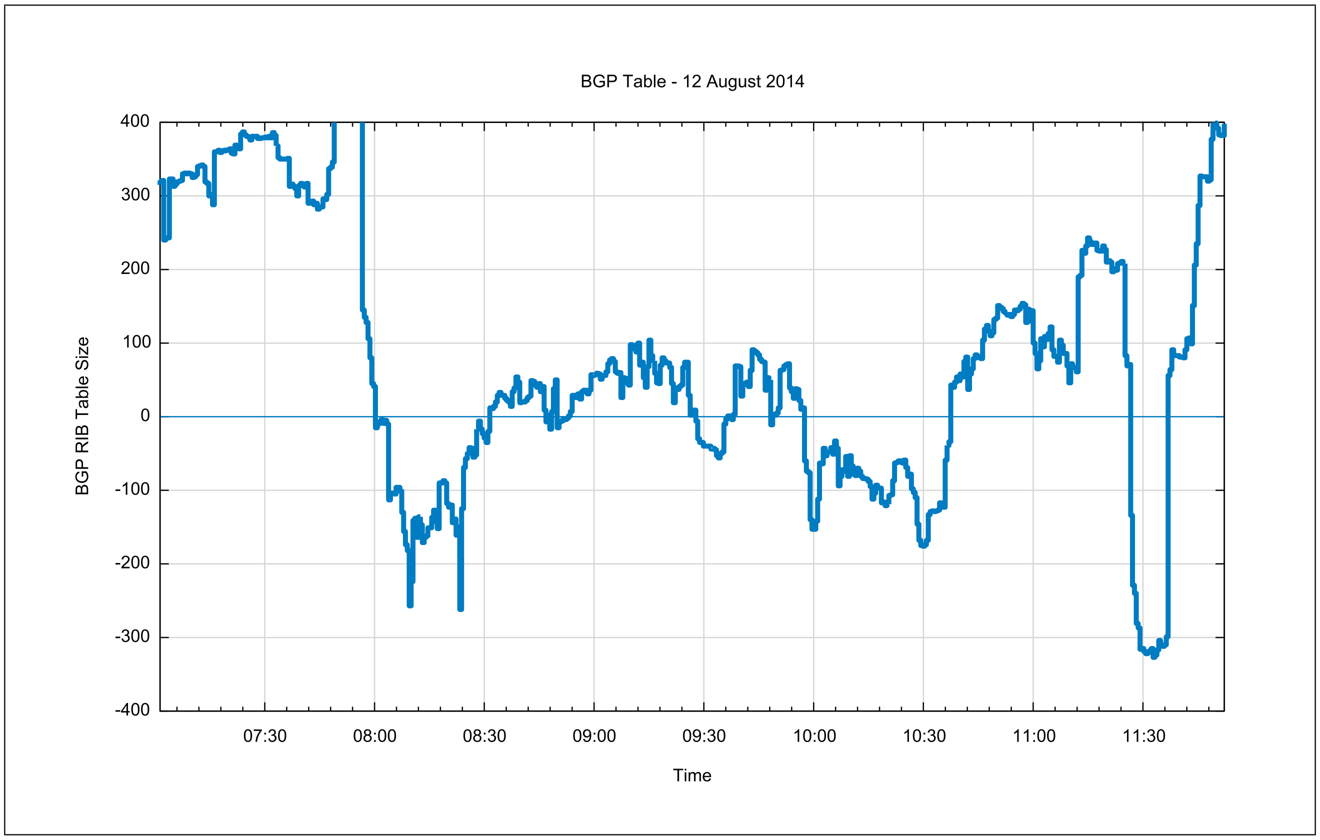
Figure 2 gives us a closer look at that route leak, as seen at AS 131072. An initial burst of 25,000 prefixes was received over a 150-second interval starting 07:49:30 UTC. The leaked routes remained in place for 200 seconds, and then were withdrawn over the ensuing 150 seconds. The routing table returned to its earlier state by 07:57 UTC.

Figure 2: BGP Routing Table: 08:00 August 12, 2014



But while the scale of the data in Figure 2 makes it look as if all returned to "normal" at the end of the route leak, that's not actually what happened. Figure 3 shows the BGP activity profile up to midday UTC of that day, and it appears that in the immediate aftermath of the route leak a further 550 routes were missing. In the period from midnight until the second immediately prior to the route leak, we had seen a net gain of 300 routes added to the routing table. When the leaked routes were removed, the table size dropped to a net loss of 250 routes for the day. That is, some 600 routes that were present immediately prior to the route leak were missing immediately following the route leak. Many (350) of these routes were subsequently restored over the ensuing 90 minutes, and then a further period of instability involving some 500 routes occurred, until the routing table was at its pre-leak level just before midday.

Figure 3: BGP Routing Table: 07:00–12:00 August 12, 2014



Is this event uncommon? Unfortunately, it's relatively common. If you look closely at the behavior of the interdomain routing system across any week in the Internet, you will see evidence of some route leaks. If route leaks are so common, then why was the leak on the 12th of August so special?

The first part of the answer to that question concerns the origin of this route leak. AS 701 is one of the so-called "Tier 1" service providers. AS 701 does not purchase upstream transit from any other provider, and, more critically in this context, its route advertisements are, in general, not filtered. Further down in the transit and peering hierarchy the chances of having a filter applied to your advertised routes is high, and the implication is that any form of route leak is quickly suppressed. But if AS 701 is the origin of a route leak, then no other *Internet Service Provider* (ISP) filters the leak, and the advertisements flow across the entire Internet. So the first factor of this leak was that every nondefault BGP speaker was exposed to the route leak.

Secondly, it was a large route leak, in which an additional 26,000 prefixes were added to the Internet routing table for the duration of the leak. While leaks of a few thousand routes are commonplace, they are generally locally contained and appear to be readily absorbed, but 26,000 routes is a somewhat different proposition. It's a significantly large leak.

The third factor is the Internet routing table. At this time many BGP speakers were holding routing tables of around 500,000 routes. There is no single view of the BGP routing system; every BGP speaker gathers its own view, but there is a common core of routes, and at the time of this route leak the common core of advertised routes was around 500,000 routes for most BGP speakers. The leak of 27,000 additional routes on August 12th pushed most BGP speakers to carry in excess of 512,000 routes for a short period of time.

This size of the routing table (512,000 routes) is a default limit point for some commonly deployed items of equipment. The specifications from some commonly used switching equipment have some references to the number 512,000 in the fine print as a default setting for the number of IPv4 entries that are carried in a high-speed lookup cache.

When the number of routes in the routing table exceeds this number, numerous potential scenarios are possible. Note that I am not describing the exact behavior of any particular equipment or configuration here, just the options for failure.

The worst possible option is that the unit crashes and awaits an operator intervention before rebooting; this option may be related to the additional withdrawals that are seen in Figure 3.

Another option is that the condition triggers a reset of the equipment. In the case of the route leak, the reset of the local equipment would take longer than the period of the route leak, and as the equipment came back on line it would be loaded with the "normal" load of some 500,000 routes, and it would functional normally once more.

Another possibility is that new and updated routes are simply discarded by the unit in its forwarding caches. This action would result in a rather subtle condition where, for packets addressed to a relatively small number of prefixes, the equipment would silently discard the packet. However, the operating BGP process on the equipment would not necessarily be aware of this action and would report that all was normal.

Something to note about this particular event is that it is more in the way of a warning of what is to come. The continued growth of the routing table is basically inevitable, and by late November 2014 the pool of common BGP routes was passing across the same 512,000 level, and this time it didn't recede, but continued to grow. So in some ways the route leak of August was a warning of what was to follow in a few months.

But before looking at the dynamics of routing-table growth, it's useful to ask why is this particular value—512,000 routes—presents such a problem for some items of routing equipment. And to do that we need to look inside a router.

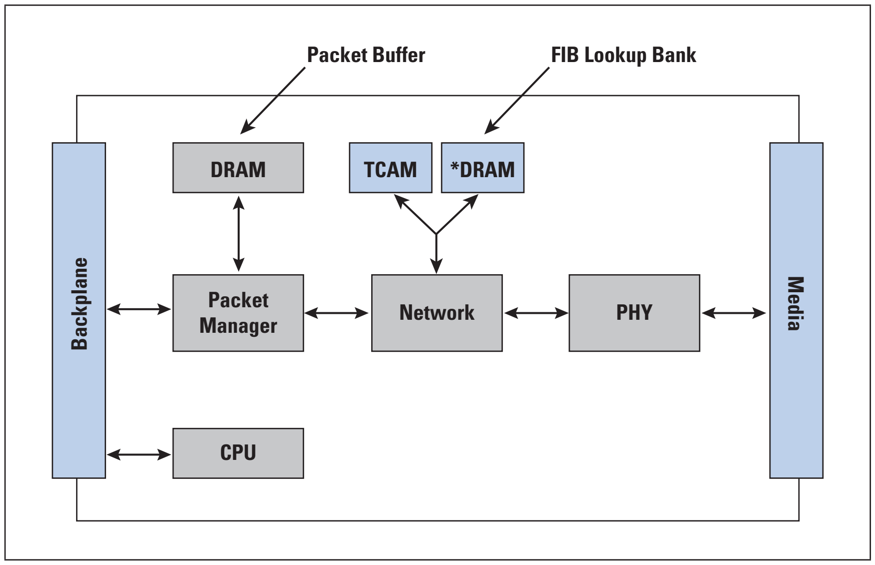
Router Internals

How do these routing-table limits, like 512,000, arise in routing equipment in the first place? Why isn't it possible to design routing equipment that is not arbitrarily limited in this manner?

The internal design of a high-speed router can be described in an analogous way as similar to the old mainframe computer architectures: as a set of specialized modules attached by a common *backplane*. These modules include a *controller,* a *switch fabric card,* and a collection of *line interface cards*.

The purpose of each line interface card is to perform as much as it can in an autonomous manner. It is designed to minimize the load that is imposed on other components of the unit, meaning that each line interface card has many roles to perform; these roles can be summarized as a *Physical Interface,* a *Network Forwarding* unit, and a *Packet Manager* (Figure 4). The Physical Interface unit includes the digital signal processing units that support the interface to the physical media. The interface that this unit presents to the remainder of the line interface card is essentially one of an assembled data packet. For incoming data packets, the network unit performs the initial part of the switching function, where for each received packet the line card looks up a local forwarding table, using the destination parameters from the packet as the lookup key.

Figure 4: Logical Structure of a Line Interface Card



The result of the forwarding-table lookup is the hardware address of the outgoing interface. If this interface is located on the same line card, then the packet is queued to the output structure associated with that local interface. If the interface is located on another card, then the packet is passed to the packet manager for transmission along the backplane to the switching unit to be passed to the outbound line interface card.

A critical aspect of the design of the line interface card is the memory structure used to support the network-level destination-address lookup. This lookup must be completed within the time defined by the maximal packet arrival rate, so for high-speed line cards the performance of this forwarding-table memory structure is critical.

An approach used in some routers is to use *Ternary Content Addressable Memory* (TCAM). TCAMs store a routing prefix in each memory "slot," using a ternary-state representation of the bits within the prefix ("ternary" because the stored values in the table are either "1," "0," or "don't care"). When presented with an IP address, the TCAM module returns the address of the router interface slot that is the longest match network prefix against the destination address. The advantage of TCAM is that it is consistent, in that every lookup takes just one TCAM cycle time. However, TCAM memory requires a significantly higher gate count per stored bit (up to 24 gates per bit), and the storage structure can be somewhat inefficient, so although TCAM offers consistent performance, it is expensive and consumes a significant level of power on the line card. TCAM is an expensive approach.

An alternate approach is a *trie* (a radix-tree lookup structure) lookup using conventional memory and an *Application-Specific Integrated Circuit* (ASIC) front end. The advantage of this approach is that the routing table can be stored in conventional high-speed *Dynamic Random-Access Memory* (DRAM), which is much cheaper than TCAM, but it does require an ASIC front end. The lookup function also requires multiple comparisons, and the number of comparisons to complete an address search is variable, so this approach does not provide an answer within a consistent time interval. In general, the larger the overall table, the slower the lookup, but the exact performance of a trie depends on the distribution of prefixes in the table, the choice of trie structure, and the specific lookup algorithm that is built into the ASIC.

The question when designing a line card is how much lookup memory should be provisioned on the card, how fast the memory should be, and whether to use a TCAM or a trie structure. The larger the memory and the faster the lookup, the higher the cost, so a trade-off is made between provisioning enough memory and fast enough memory for the expected service life of the unit and at the same time avoiding the cost of overprovisioning.

Two important questions must be answered when looking at this aspect of router design. How quickly will the routing tables grow in the coming years? And how quickly will transmission speeds grow? The answer to the first will influence the size of the forwarding tables in the line interface cards, and the answer to the second will influence the desired memory cycle time.

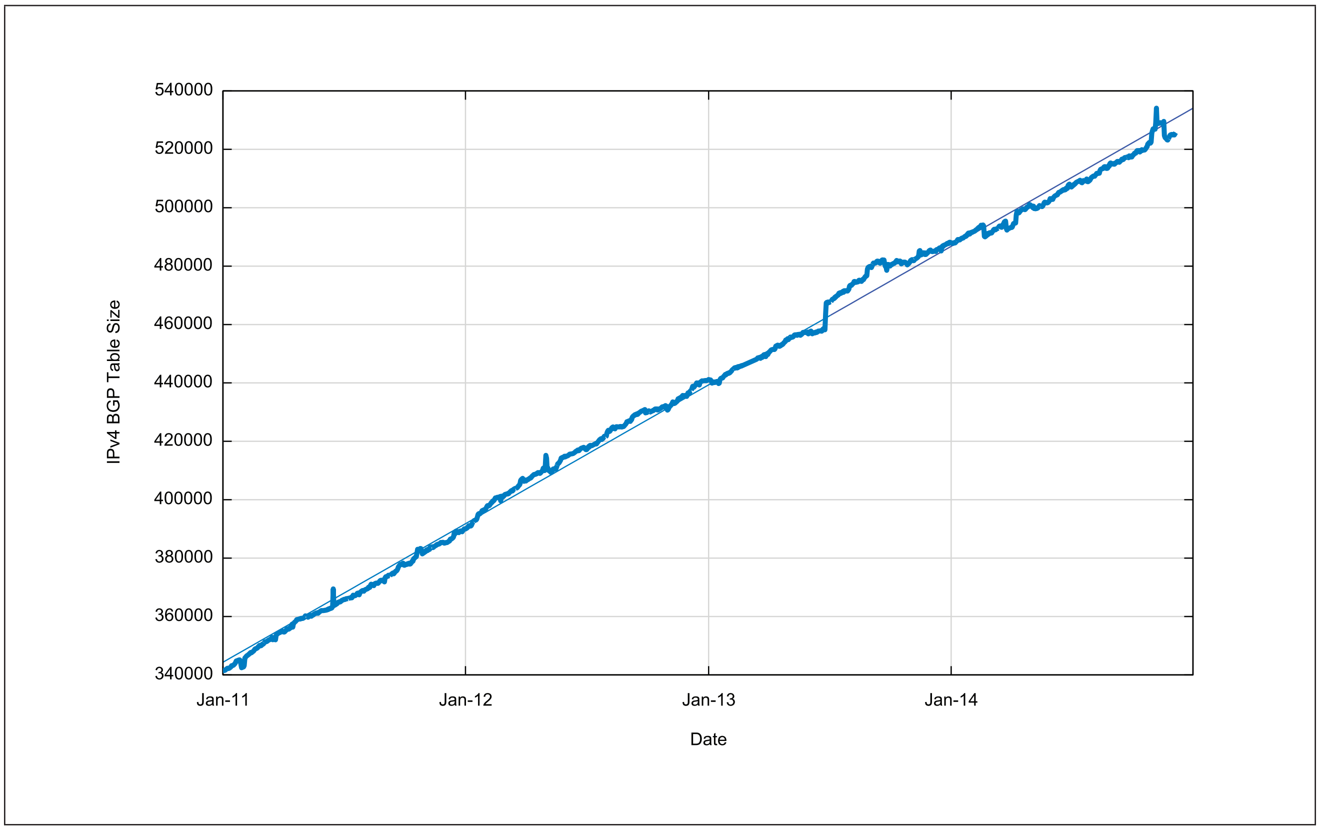
Let's looks at these two questions in further detail.

Predicting Routing-Table Growth

Sometimes these table-overflow events are unpredictable, and the route leak on the morning of August 12, 2014, certainly falls into the category of an unpredicted event. But the subsequent growth of the common pool of advertised routes is a predictable event. How quickly is the routing table growing?

Since January 2011 the Internet routing table has increased from some 355,000 entries to the current (late November 2014) level of some 523,000 entries. As can be seen in Figure 5, the overall trend of growth in the past 3 years is that of constant, or linear, growth. What is perhaps anomalous here is that during this same period three of the five *Regional Internet Registries* (RIRs) exhausted their general use pools of IPv4 addresses, yet the momentum of growth in the routing table was largely unaffected by these events. We saw neither a massive change to a large number of more specific advertisements being added to the routing table nor a marked decline in the number of new prefixes appearing.

Figure 5: IPv4 BGP Table Growth 2011–2014



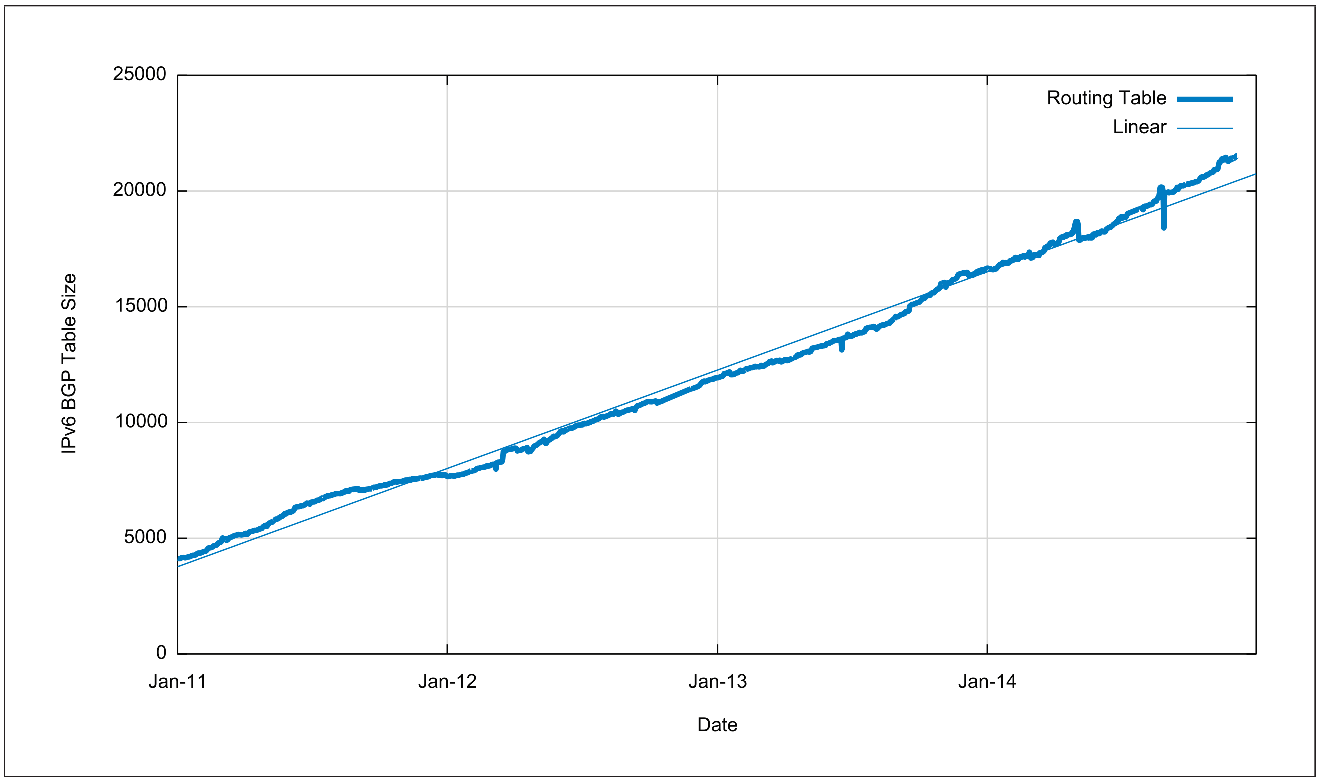
The overall metrics of Internet IPv4 routing-table growth in this period are a modest level of between 8 to 9% per year for most of the basic routing metrics (Table 1). Of course this point is the one where address exhaustion is a little more visible, and the growth in the total span of addresses has grown at a far lower rate of just 2%.

Table 1: IPv4 Routing Metrics 2013–2014

IPv4 Routing Table	Jan-13	Nov-14	Relative Growth (p.a.)
Prefix Count	440,000	523,000	9%
Root Prefixes	216,000	255,000	9%
More Specifics	224,000	268,000	9%
Address Span	156/8s	162/8s	2%
AS Count	43,000	49,000	7%
Transit ASs	6,100	7,200	9%
Stub ASs	36,900	41,800	7%

These days any consideration of the overall routing environment must also include consideration of the IPv6 network. Since the start of 2010 the IPv6 routing table has expanded fivefold, from some 4,000 entries to more than 20,000 entries at the end of 2014. However, this growth has also been predominately a linear growth since 2011, with the table size growing by approximately 4,000 entries per year over this period (Figure 6).

Figure 6: IPv6 BGP Table Growth 2011–2014



The overall metrics of growth in the IPv6 routing table since January 2013 are shown in Table 2.

Table 2: IPv6 Routing Metrics 2013–2014

IPv6 Routing Table	Jan-13	Nov-14	Relative Growth (p.a.)
Prefix Count	11,500	20,580	31%
Root Prefixes	8,451	14,030	27%
More Specifics	3,049	6,550	41%
Address Span	65,127	73,936	7%
AS Count	6,560	9,038	17%
Transit ASs	1,260	1,728	17%
Stub ASs	5,300	7,300	17%

Over this period, when the IPv4 network added a further 172,000 routing entries, the IPv6 network grew at a somewhat more modest level, at least in absolute terms. The number of routing entries grew from 11,500 to 20,500 routes, adding an additional 9,000 prefixes over this period. However, in relative terms this growth represents an annual growth rate of some 31%, which is considerably higher than the equivalent metric in IPv4.

Since 2011 the average growth of routing entries in the routing table has been relatively consistent at a long-term average of some 140 net additional entries per day. In relative terms this growth represents a steady decline in relative growth, falling from a relative growth rate of some 15% per year in 2011 to around 9% by the third quarter of 2014. This slowing down of growth in the IPv4 network could be attributed to market saturation factors in many markets in the developed world, or possibly due to the exhaustion of IPv4 addresses, which has pushed much of the growth activity behind various forms of *Network Address Translators* (NATs). What these figures indicate is that the outlook for IPv4 table growth would be best modeled on a simple linear model, looking at a medium-term growth rate of some 50,000 additional entries per year. This model implies a prediction of the IPv4 routing table reaching some 750,000 entries 5 years from now, in 2019.

However, it must be stated that this model *is* just a model, and it assumes continuity of the environment that accelerates routing-table growth, and of course continuity is simply not going to happen. In what I could describe as the most rational direction for the Internet, the momentum of IPv6 adoption should gather pace, and at some stage within this 5-year outlook, there will be a critical mass of IPv6 deployment such that an IPv6-only end client will have a seamless experience when using the Internet. At that point the momentum behind further IPv4 growth should taper off, and then we will see the IPv4 network shrink as IPv6 assumes the role of the protocol platform for further growth of the Internet. But such a rational perspective of the medium-term future has been constant over the past 5 years at least, and yet it has not eventuated so far. We have to recognize the possibility that we will continue to use IPv4 over the coming 5 years, and absorb the growth pressures through more efficient use of addresses. This paradigm would imply increasing the pressures in address sharing in NATs, looking at ways to intensify the use of public address pools across larger populations of served clients, but may also imply the emergence of fine-grained routing advertisements.

The current convention of a minimum advertised routing prefix size in the default-free zone of the Internet of /24 routes is indeed a common convention across network operators, and it is conceivable that the increasing address scarcity pressures may alter this convention. If we move to an Internet that can support the common acceptance of /25 routes, and even /32 routes, the predictions of the resultant routing-table size are of course far more uncertain.

The growth rates for the IPv6 routing table have increased from an early rate of less than 1 entry per day in 2006 to an average rate of some 17 new entries per day at present, with admittedly a high rate of variance. In relative terms, when this growth is expressed as a proportion of the routing table, the growth rate is slowing down, and the current relative growth rate is somewhere between 20 and 40% p.a. for IPv6.

Within the obvious bounds of uncertainty that accompany any such predictions, these numbers are not particularly alarming in terms of requirements for routing hardware. The routing table is stored in a memory structure, and the capacity and price of memory are related to the number of gates that can be placed into a single integrated circuit. So far *Moore's Law,* postulated some 50 years ago, continues to hold sway, and the silicon industry has been able to double the number of gates on an integrated circuit chip every 18 months or so. If the routing space were growing at a faster rate than this, then there may be some cause for concern about the future cost-effectiveness of routers, but in the IPv4 network it is simply not happening. In IPv4 the linear growth model is far lower than the exponential growth model of Moore's Law, so there is little cause for concern in that domain.

For IPv6 the numbers are a little closer to Moore's Law; if we take a model of the IPv6 routing table doubling in size every 2 years, then the IPv6 routing table is growing at a comparable pace. The mitigating factor here is that the absolute size of the IPv6 table is relatively small, and a 5-year growth outlook from 20,000 entries to some 120,000 entries is not an overly concerning prospect.

Predicting BGP Routing Update Growth

Are there other aspects of the growth of the routing system that we should be concerned about? The BGP protocol is a distance vector protocol, and a common weakness of such protocols is that the protocol reaches convergence by a process of repeated iteration of communication of updates between peer BGP speakers. Each time a BGP speaker receives information of a better path to a destination, it passes this updated information to each of its other peers.

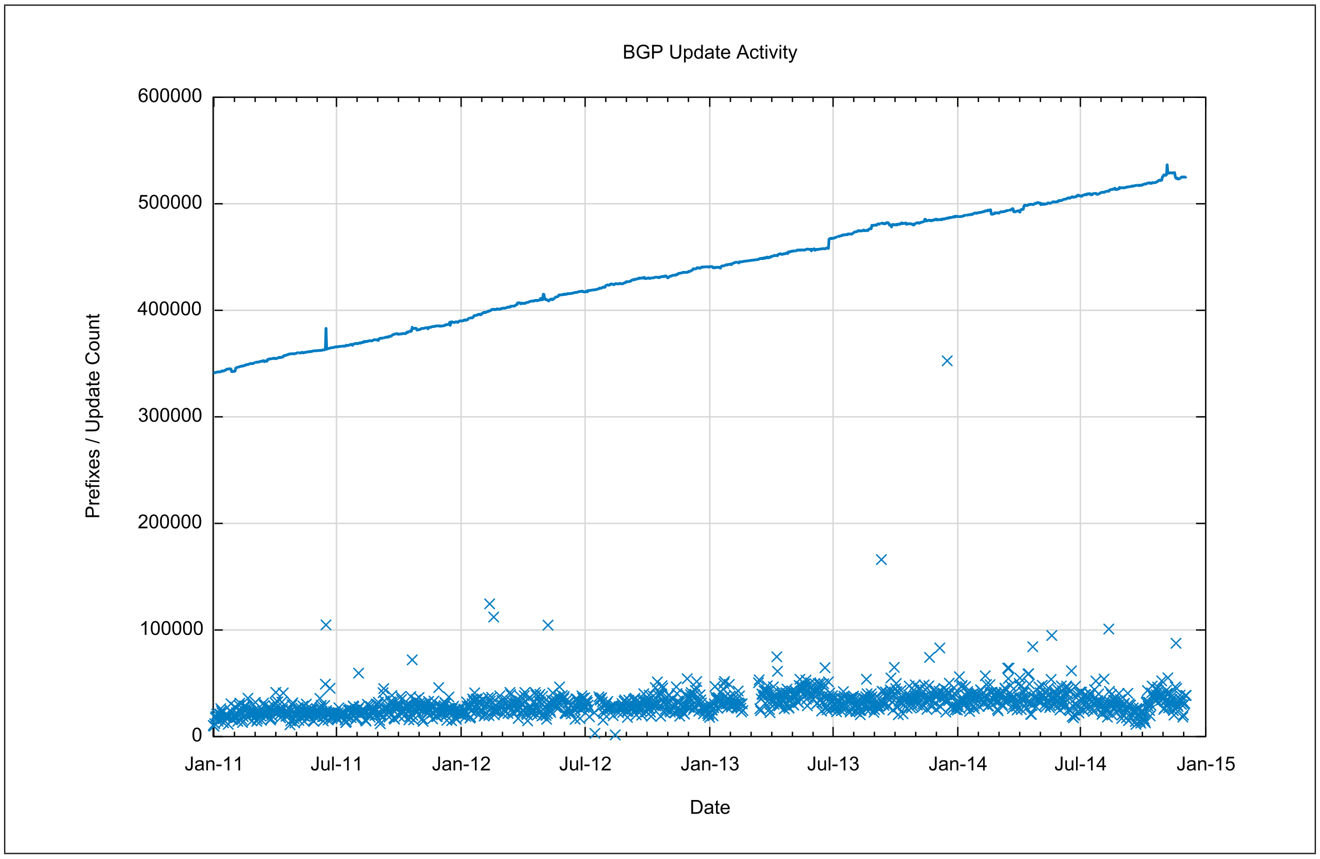
One would expect that as the number of routed entities increases, and as the number of Autonomous Systems (BGP "networks") increases, then the number of updates in BGP should increase at a comparable rate. Of course BGP has many attributes that damp this growth in updates, including the use of TCP as a transport protocol, which removes the need for periodic flooding updates between peers; the use of a *Minimum Route Advertisement Interval* (MRAI) timer, which damps the update rate between BGP speakers; and the use of the AS Path attribute, which prevents the "count to infinity" problem of conventional distance vector routing protocols.

However, these measures should not prevent any growth in the number of BGP updates. At best, they might mitigate such growth, but one would expect that, over time as the Internet grows, the amount of bandwidth and processor capacity devoted to routing should increase as the size of the Internet increases. Over time routers should need faster processors and higher bandwidth to support the operation of BGP. At the same time a larger network with fixed protocol-defined timers should take more time to converge to a stable state. So we should expect to see an increase in the update message counts of the protocol for each BGP speaker and extended convergence times as the Internet grows.

What do we see?

Nothing visible in the observed data supports these expectations. Over the past 4 years the number of entries in the IPv4 routing table has risen from 330,000 to 520,000 entries, yet the number of prefix updates in BGP has remained constant at some 40,000 prefix updates per day (Figure 7). The number of prefix withdrawals was relatively constant, averaging some 40,000 prefix updates per day. In terms of protocol performance, the average time to converge has remained relatively constant at some 70 seconds across this period.

Figure 7: BGP Daily Update Activity for IPv4



The major reason for the observed behavior varying so greatly from orthodox expectations of distance vector routing protocol behavior lies in the overall profile of the inter-AS topology of the Internet. As the number of component networks increases, the new networks all try to cluster towards the "core" of the Internet, and try to avoid attaching to the periphery. The result is an Internet that, as it grows, becomes denser rather than larger, and this increasing density assists BGP to scale.

The efforts with local peering, local exchange points, and large-scale multinational transit providers all assist in absorbing growth without increasing the "diameter" of the Internet, and these efforts offer a direct benefit in preserving the performance of the routing protocol itself.

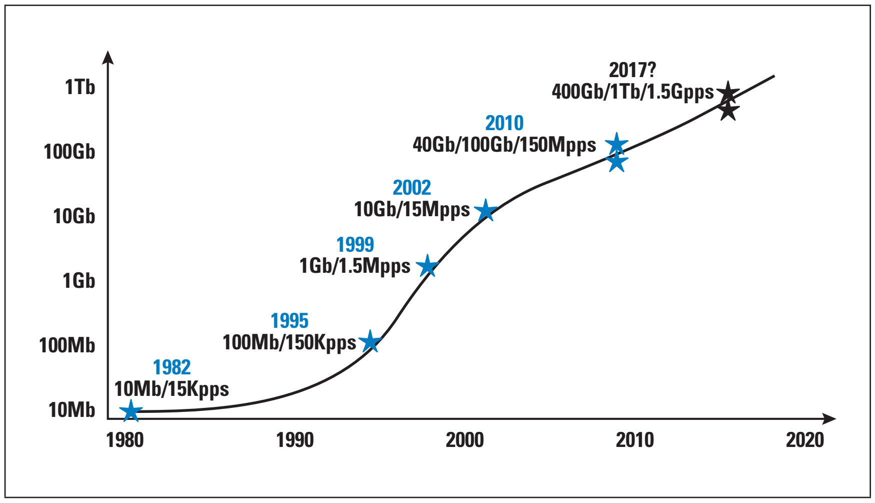
The major conclusion here is that the dynamic growth of updates is also not a cause for significant concern at this time. As long as further growth of the Internet is expressed in terms of increasing the density of the network, and as long as prefix announcements are relatively stable, the operation of the BGP routing protocol will not place extraordinary processor demands on routing equipment.

Predicting Speed

The other important parameter in terms of routing hardware is speed. The router should be capable of processing each packet, implying that in the worst case the amount of processing time available is equal to the time taken for the shortest possible packet to arrive.

In the original 10-Mbps Ethernet specification, the minimum packet size is 64 bytes, and the interpacket gap and frame preamble accounted for a further 20 bytes, implying that for the 10-Mbps Ethernet carrier, the maximum packet rate is 14,880 packets per second (pps), or one packet every 67 microseconds. Since the original 10-Mbps Ethernet specification was standardized in the early 1980s, the speed of transmission systems has increased dramatically. The pace of change in Ethernet speeds is shown in Figure 8.

Figure 8: Ethernet Speeds



Across this evolution of carrier speed, the basic unit of the minimum packet size has remained constant at 64 bytes, implying that for today's 100-Gbps systems the maximal packet rate is some 150 million packets per second, and the interpacket arrival interval is now 6.7 nanoseconds (ns). Taking this thought further with the anticipated 1-Tbps Ethernet specification, the interpacket arrival interval would be cut by a further factor of 10, to 0.67 ns.

Beyond the Tbps threshold, predictions of carriage speeds are difficult to make. Between 1995 and 2002 we saw the carriage speed rise from 10 Mbps to 10 Gbps, a thousandfold increase in 7 years. But a further 8 years elapsed until the standardization of the 100-Gbps system in 2010. To some degree we expect to see a 1-Tbps standard in 2017, but beyond that there is no clear consensus on where and how any further speed increases may be realized.

Router Limitations

What if you wanted to purchase a router today and wanted it to have a production lifetime of many years? What are the basic specifications that such a unit would need to meet in order to address the anticipated demands of, say, a 5-year service life routing the public Internet using BGP?

The processor speeds are not a major issue in terms of processing BGP routing updates. It appears that the push from network operators to maximize connectivity has a positive feedback in terms of limiting the growth of network updates, and the processing capability required to keep pace with today's BGP would not be significantly different from that required in 10 years. The processing capability required in today's routers is not going to vary significantly in the coming years.

However, it's not the same story in terms of the forwarding-table size of the line cards. At the start of 2014 a TCAM with capacity for 512,000 IPv4 entries and 25,000 IPv6 entries would have still been adequate, but now at the end of the year these numbers are inadequate. Ten years is possibly an adequate amount of time to see the transition to IPv6 through to completion in an optimistic scenario, in which case it may no longer be necessary to provide any residual IPv4 support. But this transition has so far taken longer than anyone predicted even 10 years ago, so perhaps in terms of estimating future needs for routing equipment, we should take a more conservative outlook. That conservative outlook would see further fragmentation of the IPv4 address space, and that pessimistic scenario would see the IPv4 routing table approach 1 million entries in late 2019. In addition, we need to include consideration of the IPv6 forwarding table. Assuming some form of momentum behind continued uptake of IPv6 in the coming years, we can anticipate that the IPv6 routing table will grow to some 125,000 entries by 2019.

Beyond that it's more challenging to predict. If we predicted that we would continue to use fine-grained routing control to perform traffic engineering, and use prefix blocks for network policy discrimination, then we could anticipate that the level of routing fragmentation in IPv6 would rise to the same levels we see in IPv4 today. If that's the case, then at the 10-year point we can anticipate an IPv6 routing table of some 512,000 entries.

While Moore's Law talks about the number of gates in an integrated circuit, it does not make the same prodigious predictions over the speed of the chip clock, and clock speeds certainly have not doubled every 1 or 2 years. The fastest available commodity DRAM uses a clock cycle time of between 40 and 50 ns, which is far too slow for 100 Gbps, let alone 1 Tbps. Router memory uses specialist high-speed memory, such as *Double Data Rate Type Three Synchronous Dynamic Random-Access Memory* (DDR3DRAM) and *Reduced-Latency Dynamic Random Access Memory* (RLDRAM), which have clock speeds of up to 9 and 1.9 ns, respectively. This speed is comparable to a 100-Gbps transmission system, which is the form of memory used in today's routers.

If we want this router to survive a production lifetime of 5 years, then the line speeds present a challenge. If the network sits on 100-Gbps transmission systems over this period, then current state-of-the-art high-speed memory would be adequate, but that's a rather unrealistic expectation. Within this 5-year span we will most likely see the emergence of 1-Tbps transmission systems, and if that happens then we will have to improve the clock speeds both in memory and in the line-card packet-processing engines to operate at sub-nanosecond clock speeds. I suspect that this clock speed issue may be the harder challenge and may call for the more imaginative solutions in router design in the continuing effort to meet the demands of an ever-growing Internet.

For production processors the clock rate has remained relatively constant for the past decade. The state-of-the-art in 2002 was a 3-GHz processor, and it has increased only to a 5-Ghz processor today. In the computing world the quest for ever-faster computers quickly turned from a quest for faster clock speeds across a giant monolithic system into a quest for ever-larger amounts of parallelism. That way the computer industry was able to meet escalating demands for processing capability and throughput without resorting to exotic technologies in order to support extremely high clock speeds. If this story is less than reassuring, the picture with memory speeds is no better. High memory speeds are achieved through pipelining of memory access requests, rather than in a basic increase in the clock rate.

The Internet may be on the verge of a similar threshold in the design of transmission and switching systems. To date the effort has been largely one of increasing clock speeds in what is essentially a serial paradigm.

BGP is a single-best-path selection routing protocol, and efforts to introduce serialism, such as in equal-cost multipath selection or other forms of dispersed traffic across multiple paths in parallel, have not proved to be all that robust in an inter-AS routing environment. But we can't rely on turning up the clock speed indefinitely.

At some point we may need to take some of the intra-AS approaches to traffic management across parallel paths, using various forms of path pinning, segment routing, and multipath routing, and apply it to the inter-AS routing space, so that we would be looking at further speed increases through the explicit approach of parallelism.

Of course there is also "Plan B." It we really want to reduce the maximal packet rate on high-speed transmission systems, we can always contemplate lifting the minimum packet size. If the minimum packet size had kept itself in proportion to carriage speed, a 64-byte minimum packet on a 10-Mbps system would be a 64-kilobyte minimum packet on a 10-Gbps system, and a 1.2-megabyte packet on a 1-Tbps system. Lifting the minimum packet size to 1.2 megabytes on very-high-speed systems is perhaps heading too far, but when we contemplate these 1-Tbps systems, then perhaps we should reserve some time to think about speed and capability and whether it's time to revise the minimum packet sizes on these ultra-high-speed systems.

Either way, while the next 5 years of Internet growth can be predicted within some acceptable levels of uncertainty, trying to push this range of visibility out to 10 years is a tough task. The continual pressures of scale and speed don't look as if they are stopping anytime soon, so no doubt sometime in the future we will encounter more Internet "bad hair days," as deployed equipment trips over further basic limitations in their size and speed in the face of the inexorable continuing growth of the Internet.

For Further Reading

[0] Yakov Rekhter, Susan Hares, and Tony Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, January 2006.

[1] Geoff Huston, "The BGP Routing Table," *The Internet Protocol Journal,* Volume 4, No. 1, March 2001.

[2] Kris Foster, "BGP Communities," *The Internet Protocol Journal,* Volume 6, No. 2, June 2003.

[3] "BGP: the Border Gateway Protocol Advanced Internet Routing Resources," **`www.bgp4.as`**

GEOFF HUSTON, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of numerous Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005. He served on the Board of Trustees of the Internet Society from 1992 until 2001. E-mail: **`gih@apnic.net`**

The ZigBee IP Protocol Stack

by Douglas Comer, Purdue University

A major area of networking is emerging: communication among intelligent embedded systems. The idea is that computing systems can be embedded in many devices, and the systems will be able to use the Internet to communicate with other devices. Of course, some embedded devices will provide information that humans view, and humans may run applications that control embedded devices. Consequently, the new networking paradigm does include some human interaction. However, the major emphasis is on systems that can interact with their environment and with each other, rather than on conventional computers that store data and run applications. Researchers and professionals have coined the terms *Internet of Things* (IoT, or iThings) and *Machine-to-Machine* (M2M) *Applications* to capture the idea. Despite sounding awkward, the phrase "Internet of Things" appears to have become widely accepted.

This article begins by presenting examples of intelligent embedded systems and the way they communicate. It then discusses one technology in detail: a protocol stack for wireless mesh networks that is designed for smart-grid applications. The article assesses the protocol stack, the use of IPv6, and some of the consequences of the design.

Sensing, Monitoring, and Control Applications

We use the term *embedded system* to refer to a computational system that is an integral part of another mechanism or device. The chief difference between an embedded system and a conventional computer system arises from their external connections. A conventional computer system deals with information: the computer can store, access, and manipulate data. In contrast, an embedded system can *sense* and *control* the physical world around it.

As an example, consider a thermostat used to control a heating and air conditioning system. A modern thermostat (called a *smart thermostat*) constitutes an embedded system: the thermostat contains an embedded processor that runs software to perform all functions. A user can configure the thermostat to change settings according to the time of day. The thermostat has connections to a variety of sensors that might include an indoor temperature sensor, an outdoor temperature sensor, a sensor that detects airflow (that is, whether the fan is operating), and a sensor connected to push buttons that allow a user to set the desired temperature. More advanced systems have sensors for the relative humidity of the air. In addition, the thermostat has connections to controls that allow the processor to turn the heater or air conditioner on or off, regulate the speed of the fan, and control humidifier and dehumidifier functions.

Most computer users are already familiar with embedded systems. For example, a printer connected to a computer incorporates an embedded system. When a computer sends a document to the printer, the embedded system of the printer controls the motors and mechanisms in the printer, causing them to feed sheets of paper through the printer, move the ink jet mechanism, and spray drops of ink. The printer also contains sensors that can detect a paper jam or low ink supplies.

General Electric (GE), the largest industrial company in the United States, is going beyond items for the consumer market. GE produces aircraft engines, power-plant turbines, locomotives for railroads, medical imaging equipment, and heavy-duty machinery that transports people, heats homes, and powers factories. Using the phrase *Industrial Internet,* GE has launched a major initiative to incorporate communicating embedded systems in both its factories and its products.

Power Conservation and Energy Harvesting

Some embedded systems, such as the embedded control system in a printer, attach to a reliable source of continual power. However, many embedded systems rely on temporary power, and are designed to conserve energy. For example, cell phones run on batteries and environmental sensors located in remote locations (for example, a desert) may use photocells.

As a special case, some embedded systems are designed to *harvest energy* from the environment around them. For example, a sensor in the ocean might use the motion of waves to generate power, and a sensor near a hot spring might use thermal energy. Energy harvesting even includes the kinetic energy that humans generate merely by opening a door or flipping a light switch. An embedded system that uses harvested energy may need to operate periodically—the system might need to accumulate energy until a sufficient charge is available (for example, to run a radio transmitter).

A World of Intelligent Embedded Devices

To understand the vision for the Internet of Things, we have to imagine that powerful embedded systems will be everywhere: houses, office buildings, vehicles, shopping malls, and street corners. For example, consider vehicles. In addition to systems that provide entertainment and navigation, designers envision systems that allow a vehicle to calculate the distances to surrounding vehicles, sense objects in the roadway, warn of changes in pavement (for example, from construction), and sense lanes and warn the driver when a car drifts. A vehicle with intelligent embedded systems can communicate with nearby vehicles, and can coordinate braking. An intelligent embedded system can use facial recognition to identify drivers when they enter a vehicle, adjust settings to their preferences, monitor an individual driver's driving habits and watch for unusual driving, and adjust warnings to accommodate a specific driver's reaction times.

In office buildings, embedded systems already sense the presence of individuals and adjust lights and heating or cooling accordingly. A system can use sensors to change the heating and cooling systems when windows are open. More important, an intelligent embedded system will be able to use learning algorithms to accumulate patterns. For example, if given employees move in and out of their office frequently during the work day, the system can learn not to turn off the heat until they are absent for a longer time. Similarly, if an employee tends to work late, the system can learn the pattern and control the office environment accordingly. Thus, if the employee usually arrives early and goes home at 3 p.m. each day, an intelligent system can learn the pattern and anticipate the employee's arrival and departure.

The Importance of Communication

Why is emphasis shifting to intelligent embedded systems that can communicate? There are many advantages. For example, in addition to local coordination, communication allows systems to exploit *Cloud Computing*[10, 11] to analyze data from a set of embedded devices. Communication means individual embedded systems can have smaller memories and less-powerful processors, meaning they will use less energy. In short, small embedded systems can achieve complex functions by working together with nearby embedded systems or by accessing remote information.

As an example, consider a set of sensors used to assess the stress on civil infrastructure, such as bridges. Measuring stress is important in understanding whether a bridge can tolerate the load, whether reinforcement is warranted, or when a bridge should be replaced. To measure stress, engineers place small battery-powered sensors at various points along a bridge. Without communication, each sensor must have a local store to keep measurements along with a timestamp for each. If each sensor includes a radio, the set of sensors can form a wireless network in which measurement data is passed along to a collection point, which may be located across the Internet, far from the bridge. In terms of measurement, the important difference arises from coordination and rapid assessment. Communication allows sensor nodes to run a protocol that takes readings simultaneously. Uploading data in real time makes it possible to detect dangerous situations quickly and take action before a disaster occurs.

Communication can also lower costs. For example, consider *smart meters* used by utility companies. The traditional approach used to assess usage consists of placing a meter outside each customer's location and sending a person to record the meter reading each month. A smart meter incorporates wireless communication, meaning the utility company can read the meter from a remote location. Even if a smart meter uses a wireless transmitter that reaches only to the street, the meter can be read from a passing vehicle rather than an individual on foot, lowering the cost of reading meters dramatically.

Embedded Systems in Shopping Malls

In addition to the traditional sensor systems described previously, the Internet of Things includes unexpected applications. For example, many shopping malls now have large video display panels that show ads. Stores use the displays to advertise products and services as well as discounts and special promotions. Communication is needed to download ads dynamically because the content and schedules can change at any time—management needs the ability to control which ad is displayed on a given screen and how long the ad remains visible.

Where is the control system for the video displays located, and what networking technology is needed to connect the control system to the individual displays? The answer is interesting and a bit surprising: in current implementations, each display contains a TCP/IP protocol stack and a connection to the global Internet. An Internet connection allows the controller to be located anywhere. In particular, the mall can outsource IT functions by placing the controller "in the cloud." More important, providing each display with an Internet connection and protocol software means that the content does not need to reside on the same physical host as the controller.

Allowing content to be separate from the controller is important because it differentiates information owned by a retailer from the control system for a given mall, and allows a retailer to serve content to multiple malls. For example, a retailer such as Apple can place video content for ads on a cloud server, and then issue commands to the controllers for each mall to specify a schedule of items to display along with the URL of each item. Because they have Internet access, individual display systems can download a copy of an item they have been assigned to show (possibly through a local cache to improve performance).

Uploading Data from the Internet of Things

Displays in shopping malls illustrate another important capability of networked systems: the ability to upload data. Although it is not obvious to customers, some of the displays in shopping malls are equipped with a camera. When people approach the display, the system uses the camera to detect their presence. The system identifies human faces and applies analysis algorithms that use features, such as the distance between the eyes, to characterize the individual. With high accuracy, software in a mall display is able to tell whether the individual in front of the camera is male or female as well as the person's approximate age group. Thus, instead of merely following a predetermined schedule of ads to display, the system can use the characteristics of the individual to choose an appropriate ad. For example, a middle-aged male might be shown an ad for a sports car instead of an ad for women's apparel.

In addition to using video information to select ads, mall systems also gather and report data about the interaction. For example, a system uploads statistics about how many people watched a given ad, their sex and age, and how long each person or group remained in front of the display during a particular set of ads. Grocery stores are using the same approach: they are installing cameras with embedded processing capability over freezers and at other locations in the store. The cameras record whether customers stop at a given product display, how long each person looks, and how many customers finally select a product or merely move on. When it is uploaded to a server, the information from a given location can be combined with information from other locations. The key idea is that combining data from multiple sites increases the accuracy of the analysis.

Wireless Networking and IEEE 802.15.4

How should devices be connected to the Internet of Things? Wireless networking technologies are popular, even in the case of semipermanent deployments across a small area. For example, consider the electronic displays in malls. Although they are semipermanent (that is, a display usually remains stationary for weeks), wireless networking means a display can be moved without installing a new network connection.

What wireless networking technologies should be used to connect an intelligent embedded system? The answer depends on several factors, including the geographic distance between nodes of the network, the desired data rates, and the power requirements. Power is important in two ways. In the case of embedded sensor systems that run on battery power, overall power consumption must be minimized to maximize battery life. Although the power used by a radio transmitter can dominate the overall battery drain, using a smaller memory or reducing processor speed can also lower overall power requirements. In the case of embedded systems that have a continuous power source (for example, connect to a wall outlet), it may be necessary to limit radio transmissions to avoid interfering with other devices or other transmissions.

Several low-power wireless networking technologies have been standardized. This article considers a network technology defined by the IEEE standard 802.15.4[1]. Various versions of 802.15.4 have been produced; they differ in the frequency bands and modulation techniques used as well as the *Maximum Transmission Unit* (MTU). The IEEE standard specifies the physical and *Media Access Control* (MAC) layers of the network, and other groups have defined upper-layer protocols for use on low-power wireless networks. For our purposes, it is necessary to understand only the general characteristics of 802.15.4 technology:

- The data rate is relatively low (a maximum of 250 kbps).
- The MTU is extremely small (127 octets).
- The distance is limited (a maximum of 10 meters with a conventional antenna and power from a battery).

A Mesh Network for Smart-Grid Sensors

One application of 802.15.4 wireless technology arises from an effort to add intelligent computer management to the electrical power-distribution system. Known as *Smart Grid,* the overall plan includes placing sensors in all devices that use electricity. In addition to large systems, such as those used for heating and cooling, the designers envision sensors in kitchen appliances (for example, ovens, refrigerators, dishwashers, and even toasters), computer systems, and entertainment systems (for example, televisions and stereos), and small handheld appliances. The utility companies want to charge more during peak hours, and the sensors will communicate with the utility company to determine pricing and warn users when prices are high. Alternatively, the sensors will be capable of disabling certain uses during peak hours.

The most obvious design for a system of sensors in a residence consists of placing a base station in the residence and using a wireless technology that allows the base station to communicate with each sensor. The approach is known as a *hub-and-spoke* topology. For example Wi-Fi (802.11) systems use a hub-and-spoke approach. However, such a design does not work well for all situations. In particular, metal pipes and other obstructions in a building can interfere with wireless signals and make it impossible to reach all locations from a single point, especially in the case of portable appliances that can be moved from one room to another. Therefore, the smart-grid designers envision an adaptive system in which the set of sensor nodes automatically forms a *self-organized mesh network,* simply called a *mesh*. Each node in the mesh performs two tasks: communication for the device to which it attaches and forwarding for other nodes.

A residence will contain a *border router* that connects the mesh network to the outside world. When a node boots, it joins the mesh and tries to establish a connection to the border router. If it can reach the border router directly, the node communicates directly with the border router. If it cannot reach the border router directly, the node searches for a nearby neighbor node that has a path to the border router. In essence, the neighbor agrees to act like a router and forward packets.

If multiple neighbors have a path to the border router, a node applies a selection algorithm to choose one. The selection algorithm can account for several metrics, including quality of the radio signal (that is, the level of interference), latency, link capacity, and capacity of intermediate forwarding nodes. When a node selects a path to the border router, the node informs its neighbors and agrees to forward their packets. Thus, if it is possible to create a network that gives each node connectivity, the nodes will form a mesh network automatically. We will return to the question of how a neighbor makes a selection later in the article.

A Forwarding Tree for a Mesh Network

Our description makes the selection of paths in a mesh seem trivial. In fact, a mesh may offer many possible paths. The distinction between routing (choosing paths among networks) and forwarding (choosing paths within a network) is blurred because we can think of each radio link as a network. More important, a path with fewest hops may not be optimal because connectivity is pairwise among nodes, and a low signal strength between a given pair may make the path unreliable. Furthermore, if mesh nodes are mobile or changes in the environment interfere with signals (for example, people moving through a room), forwarding must change dynamically. Despite the potential complexity of mesh routing, solutions have been created to handle basic cases. In particular, we will see that protocols have been created to handle forwarding between the border router and individual nodes in a semistatic mesh topology.

The key idea that simplifies mesh forwarding is an observation: if the primary goal is to enable sensors to communicate with a remote server, each sensor node needs to choose only one way to reach the border router. Traffic from a remote server to a mesh node can be directed along the path in the reverse direction. That is, the forwarding paths in the network form a tree in the graph-theoretic sense (that is, a graph with no cycles), with the border router as the root. Figure 1 illustrates a set of sensor nodes and one possible forwarding tree.

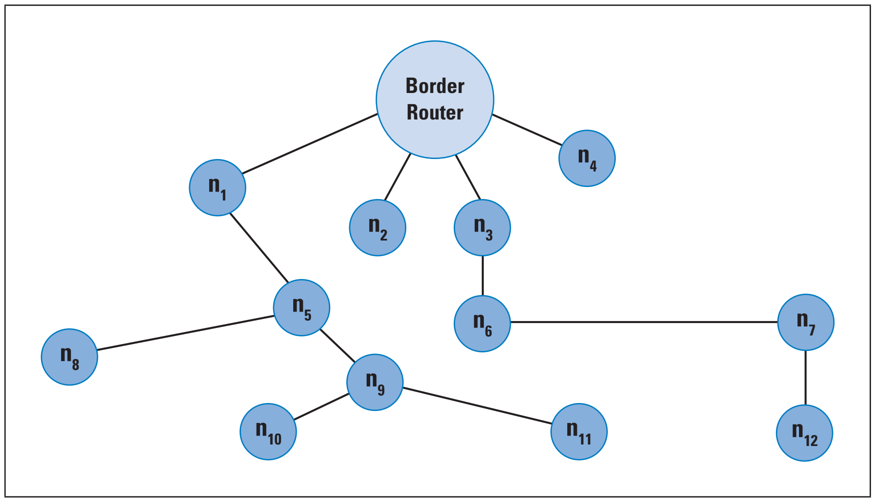


Figure 1: Example Forwarding Tree Imposed on a Set of Sensor Nodes. A Tree Results if Each Node Chooses One Path to the Border Router.

As the figure indicates, a border router is usually larger than other nodes in the network (that is, has more processing power and memory). We will see that a border router is expected to run routing and server software that provides forwarding service for the entire mesh.

Using Internet Protocols in a Mesh

In terms of Internet protocols, the question is whether to use the traditional paradigm of assigning an IP prefix to the entire mesh or to use the paradigm of treating each radio link as a separate point-to-point network.

The two approaches are known as:

- *Mesh-under:* The mesh acts like a single network, and Layer 2 protocols handle broadcast and multicast.
- *Route-over:* The mesh acts like a set of point-to-point links, and Layer 3 protocols perform all forwarding.

With the *mesh-under* approach, which is favored by IEEE, a node uses Layer 2 protocols to form a forwarding tree. The idea is similar to the bridge and spanning-tree protocols used for Ethernet. For example, a node uses a Layer 2 broadcast to discover which neighbors are within radio range. Each neighbor responds, allowing the pair to learn that they can reach each other and the signal quality. The border router also uses Layer 2 broadcast to advertise itself. Nodes in the mesh pass along information about which nodes they can reach. Eventually, each node in the mesh is aware of other nodes and how to reach them, as well as how to forward broadcast packets. Thus, given the Layer 2 address of the border router, nodes in the mesh will know how to forward packets (that is, they will have formed a forwarding tree).

With the *route-over* approach, which is favored by the *Internet Engineering Task Force* (IETF), a node uses Layer 3 protocols to identify neighbors and form a forwarding tree. Of course the underlying hardware does not understand IP addresses or the IP datagram format. Thus, Layer 3 packets are carried in Layer 2 frames. For example, to find neighbors, Layer 3 software generates an IPv4 datagram with a local broadcast address or an IPv6 datagram with a link-local multicast address. The datagram is sent via hardware broadcast.

In either approach, when a new node enters the mesh, the node must choose how to link itself into the tree. There are two steps. In the first step, a node must find the set of neighboring nodes that can be reached directly and assess the quality of the radio link to each neighbor. In the second step, the node must choose either to communicate with the border router directly or to use one of the neighbors when forwarding packets. Note that the quality of a signal is of key importance—even if a node can reach the border router directly, the node may choose an indirect path if the signal quality of the direct connection is sufficiently low.

In terms of an IP protocol stack, a major difference between mesh-under and route-over arises in IP forwarding. The mesh-under approach follows the traditional paradigm of treating the entire mesh as a single network with familiar characteristics of a single broadcast domain and the ability for an arbitrary pair of nodes to communicate.

IP assigns a single prefix to the mesh network, and IP forwarding causes any datagram destined for a node on the network to be passed to the underlying hardware interface for delivery. When it accepts an outgoing datagram from IP, the network interface uses the information that has been gathered by Layer 2 routing protocols to choose a next hop to which the datagram will be forwarded. As the packet travels across the mesh, the packet is processed only by Layer 2 on each intermediate node. When it arrives at the destination, the datagram is passed up to Layer 3. Thus, all mesh details are hidden from Layer 3.

The route-over approach makes IP aware of the mesh topology. That is, IP becomes aware that although some nodes on the network are reachable directly, others are not. In particular, using route-over breaks a standard assumption in IP protocols that if two hosts share an IP prefix, the two attach to a network that allows them to exchange packets directly. In a route-over mesh, all nodes in the mesh share a prefix, even though a given node can communicate directly only with nearby neighbors. Using IPv6 terminology, we say that a node is either *on link* or *off link*. To handle nodes that are not directly reachable, IP uses *source routing*. IP must understand the topology of the mesh and be able to specify a path through the mesh to the destination (for example, go to node 9, then to node 5, then to node 1, and finally to the border router). The next sections describe the *ZigBee* protocol stack that uses the route-over approach, and later sections assess some of the problems that arise.

The ZigBee IPv6 Protocol Stack

The ZigBee Alliance[2] and the IETF[3] are cooperating to define the use of IPv6 in a mesh design that follows the route-over approach. The ZigBee Alliance has defined an open standard known as *ZigBee IP*[4]. Table 1 lists three key IETF working groups related to the ZigBee effort. The next sections describe the protocols that are being developed.

Table 1: IETF Working Groups Related to the ZigBee Effort

Name	Main Contribution
6LoWPAN	IP-over-802.15.4 Shim Layer
ROLL	RPL – A Routing Protocol for Mesh Networks
CoRE	CoAP – Constrained Application Protocol

The basic idea of the ZigBee route-over design is to use IPv6 when possible and introduce modifications as needed. A protocol has been created to compress IPv6 datagrams and send them over an 802.15.4 radio link. A modified form of IPv6 *Neighbor Discovery* is used to find the IP addresses of directly reachable neighbors, and a protocol has been invented to allow neighbors to exchange characteristics.

In addition, a new routing protocol is used to gather information about connectivity throughout the mesh and compute forwarding information. Finally, an IPv6 source route header is used to forward each datagram hop-by-hop across the mesh. The next sections describe some of the basic protocols.

IPv6 over Low-Power Wireless Networks

The *IPv6 over Low power Wireless Personal Area Networks* (6LoWPAN) effort defines the transmission of IPv6 over a 802.15.4 radio link. The primary problem is a conflict between the IPv6 requirement for an MTU of at least 1280 octets[5] and the 802.15.4 protocol that species a maximum of 127 octets. In fact, if AES-CCM-128 encryption is used, the available payload size is reduced to 81 octets. To send an IPv6 datagram over such a link, 6LoWPAN introduces an extra shim layer that performs compression and transmission. The shim layer accepts an outgoing IPv6 datagram, compresses the header, divides the datagram into a series of pieces we will call *fraglets,* and sends each fraglet in a separate packet. On the receiving side, the 6LoWPAN shim layer accepts incoming fraglets, recombines them into a single datagram, decompresses the header, and passes the result to the IP layer. Thus, IPv6 is configured to send and receive complete datagrams without knowing that the shim layer breaks a datagram into fraglets for transmission.

There are two reasons 6LoWPAN does not use conventional IPv6 fragmentation. First, 6LoWPAN needs to operate over only a single link. Thus, the protocol is much simpler because all the fraglets of a datagram must arrive in order. Second, IPv6 fragmentation cannot handle an MTU of 127 octets.

6LoWPAN Neighbor Discovery

Traditional *IPv6 Neighbor Discovery* (IPv6-ND) provides an address-resolution mechanism that can be used for, among other things, *Duplicate Address Detection* (DAD). Unfortunately, IPv6-ND makes a fundamental assumption that an IPv6 prefix maps to a broadcast domain. Therefore, a node can use IPv6 multicast, which maps to hardware broadcast, to reach all other nodes that share the prefix. In a mesh network, however, a broadcast transmission may reach only some of the nodes in the mesh, making some of the nodes off link. As a result, conventional duplicate address detection will not work correctly.

6LoWPAN Neighbor Discovery (6LoWPAN-ND) defines several changes and optimizations of IPv6-ND that are intended specifically for lossy, low-power wireless networks that have limited range. In general, 6LoWPAN-ND avoids all mechanisms that flood packets across the mesh. Instead of requiring individual nodes to engage in duplicate address detection, 6LoWPAN-ND uses a *registration* approach in which each node in the mesh registers its address with software that runs on the border router.

As nodes register their addresses, software on the border router flags any duplicates (recall that a border router has the processing power and memory needed to handle networkwide services, such as address registration). Finally, 6LoWPAN-ND allows nodes to *sleep* (that is, go into a stasis state to conserve power). When a node reawakens, it must renew its address registration in case some other node registered a duplicate address during the sleep period.

Mesh Link Establishment

IPv6 is designed with the assumption that underlying hardware links have been configured before IP software runs. In particular, IPv6 expects exchanges to be authenticated, meaning that links must already be in place. For an 802.15.4 mesh, a node must choose how to link into the forwarding tree. Link configuration is complex because radio transmissions can be asymmetric. A node cannot merely listen for transmissions from neighbors and choose the neighbor with the strongest signal, because the question is how well a neighbor can receive transmission of a node. Consider the case of a border router with a powerful transmitter and large antenna. It may be possible for a node to receive a strong signal from the border router, even if the node transmitter is too weak to reach the border router. Thus, before IPv6 can be used in a route-over mesh, a lower-level protocol is needed that allows a node to learn the level of signals that neighbors observe when receiving the transmissions of the node.

ZigBee uses the *Mesh Link Establishment* (MLE) protocol for link configuration. MLE employs a two-way packet exchange: one node transmits a message and a receiving node sends a reply. The reply reports the quality of the signal that was observed. When a node receives an MLE reply from each neighbor, the node will know how well each neighbor can receive its transmissions. Of course, signal strength can change over time if nodes move or electrical interference is introduced (for example, a large electrical motor starts to run). Therefore, the measurements must be repeated periodically.

MLE includes facilities for more than signal-strength assessment. During packet transmission, MLE allows nodes to exchange configuration information. The two nodes exchange address information, and choose a type of security for the link. Most important, MLE allows a node to inform a neighbor that it can reach a border router. Thus, when it joins a mesh, a new node runs MLE, gathers information about which neighbors have a path to the border router, and uses signal strength to choose one of the neighbors as a parent node in the forwarding tree.

Interestingly, not all the ZigBee protocols account for asymmetric signal strength. In particular, when it builds a forwarding tree, the *Routing Protocol for Low-Power and Lossy Networks* (RPL) selects links that are bidirectional; if communication can proceed only from node *A* to node *B* and not from node *B* to node *A*, RPL does not include a link between *A* and *B*.

However, the ability to communicate does not imply that the quality is the same in both directions. In the case of a border router communicating with another node, it seems reasonable to assume that the signal sent by the border router could be stronger than the signal sent by the other node. Even in the case of two nodes that are not border routers, it may be that the signal received in one direction is much stronger than the signal received in the reverse direction. Nevertheless, once it chooses a path, RPL sends traffic in both directions over the path.

Forwarding in a ZigBee Route-Over Mesh

Conventional IP forwarding uses the IP prefix when choosing a next hop. As pointed out previously, however, some nodes of the network will be on link (that is, directly reachable) and others will be off link (that is, reachable only indirectly). Therefore, when deciding how to forward a datagram, IP must use more information than the network prefix. The problem can be handled in two ways, and ZigBee IP uses the terms *storing mode* and *non-storing mode* to characterize the two approaches.

As the term *storing* implies, each node in the network stores a significant amount of information. In addition to storing the address of a parent (that is, the next hop to the border router), each node learns and stores a next-hop address for each node that is downstream. In a graph-theoretic sense, a node in the tree stores a next hop to each node in its subtree. The worst case occurs in networks where a single node, *N*, connects all other nodes to the border router. For such a case, node *N* stores a next hop to all other nodes in the network.

When it needs to forward a datagram, the IP software on a node consults the forwarding information that has been stored locally. If the destination is downstream, the information specifies the next hop to use to reach the destination. If the destination is not downstream, the node forwards the datagram along the path toward the border router.

The memory requirements for storing mode are more extensive than the previous description implies. Once RPL has computed routes, a node needs to store only a next hop for destinations in its subtree (*N – 1* destinations in the worst case). However, additional memory is needed during route computation because RPL uses a link-state algorithm. Therefore, a node must collect pairwise link advertisements for all links in the subtree and then run the shortest-path algorithm to compute next hops. The memory required is still proportional to the number of nodes in the subtree, but the computation can require twice the memory used to store next-hop information.

Although it handles transfer along the edges of a forwarding tree, storing mode does not provide optimal routing in all cases. For example, consider two nodes that lie in close proximity but are not in the same forwarding subtree.

When one sends to the other, the packet is forwarded up the tree toward the border router. If the packet reaches a node that is common to both forwarding subtrees, the packet is sent down the other tree to its destination. The worst case occurs when the border router is the only node in common with both subtrees: the packet must travel all the way up one subtree to the border router before being sent down the other subtree to the destination. The IETF is developing a protocol that will find and use routes that lie outside the forwarding tree.

The concept of a forwarding tree and a routing protocol, such as RPL, that computes and maintains the tree arises from three assumptions: the topology will remain relatively static, nodes will communicate frequently, and latency should be minimized when communication occurs. By precomputing a forwarding tree, the mesh nodes remain ready to forward any packet at any time. In situations where the topology changes or traffic is infrequent (for example, a sensor mesh where sensor values are collected once per week), the overhead incurred in maintaining routes may not be warranted. Instead, it may be more efficient to use an on-demand approach in which the mesh nodes wait for a packet, find a route, send the packet, and then delete the route.

Non-storing mode is designed for networks in which nodes have limited memory and CPU resources. To minimize local storage and processing, each node learns only two things: the set of directly reachable neighbors and the identity of one neighbor that leads to the border router. The border router is assumed to have substantial amounts of memory and processor capability, and can perform all the necessary path computation. The border router learns the complete topology of the mesh, and handles all source routing. When a node has a datagram to send, the node forwards the datagram to the border router. If the datagram is destined for a site on the Internet, the border router forwards the datagram. If the datagram is destined for another node in the mesh, the border router encapsulates the datagram in an outer datagram, uses its copy of the topology information to insert a source-route header in the outer datagram, and forwards the encapsulated datagram across the mesh. At each step, a node in the mesh finds the address of a neighbor in the source-route header, and uses the address to send the datagram to the specified neighbor. Perhaps the most serious consequence of using IPv6 to implement a route-over mesh arises from the requirement for source routing in non-storing mode and the size of an IPv6 source-route header.

Non-storing mode may seem to waste network resources, because a datagram sent from one node to another first goes to the border router and then to its destination. The worst case occurs when a packet is sent between a pair of nodes that are two hops apart but whose forwarding trees intersect only at the border router—instead of two hops, the packet may traverse *N* hops, where *N* is the number of nodes.

Nevertheless, the ZigBee Alliance selected non-storing mode to permit individual nodes to have extremely small memories and slow CPUs, minimizing both energy consumption and cost.

An important point about storing mode arises from limitations on scaling: in the worst case, information about mesh connectivity requires space proportional to *N*, the number of nodes in the network. Although most ZigBee mesh networks are expected to have fewer than 24 nodes, some mesh networks contain thousands of nodes. Consequently, for a large mesh, storing mode implies that each node must store tables that are large relative to the memories available on small devices (for example, 64 or 128 kilobytes of RAM). Using non-storing mode allows small battery-operated nodes to store only the following items in memory:

- A list of directly reachable neighbors and the MAC address of each
- The identity of the neighbor that is currently serving as the path to the border router

Note that the memory required when using non-storing mode is proportional to the number of reachable neighbors, which may be substantially less than the number of nodes in the mesh. In fact, even in a dense mesh, a node can restrict the list to the top *K* neighbors (that is, the *K* neighbors that report the highest signal strength).

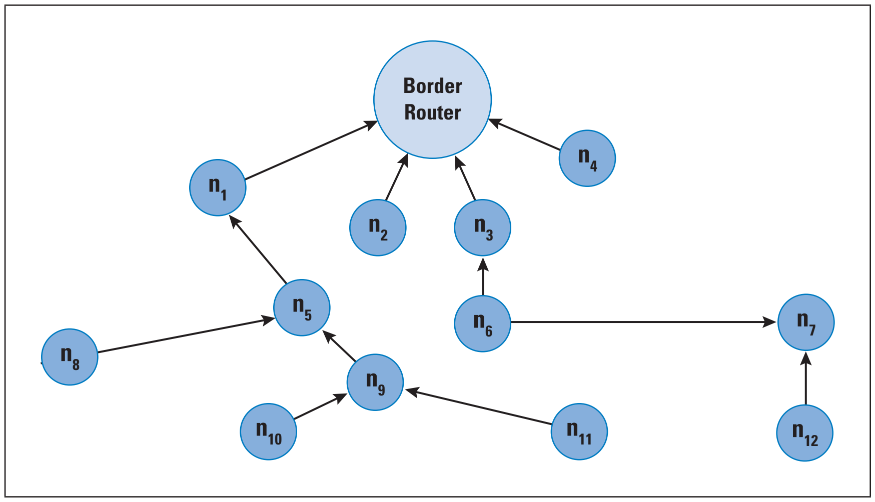
Routing Protocol for Low-Power and Lossy Networks

The IETF has defined a routing protocol that can be used with IPv6 in a route-over mesh network. Known as *Routing Protocol for Low-Power and Lossy Network*s (RPL), the protocol allows nodes to advertise direct connections and to learn about other connections in the mesh. RPL defines an IPv6 header so that datagrams can carry RPL information in addition to a payload. The ZigBee IP standard specifies the use of non-storing mode. In non-storing mode, RPL propagates connection information *upward* to the border router. The border router runs a special version of RPL software that gathers the information, bmeaning that it learns the topology of the entire mesh.

When it learns the topology, the border router computes a forwarding tree. Instead of imposing a tree on an undirected graph, RPL makes each link directed, with the direction toward the root (that is, toward the border router). Thus, RPL calls the graph of the mesh topology a *Destination-Oriented Directed Acyclic Graph* (DODAG). Figure 2 illustrates the DODAG form of the tree in Figure 1.

Although links in the DODAG point toward the root, the representation is merely a detail of the protocol, and does not dictate packet flow. In particular, when a border router needs to send a datagram to one of the nodes, the border router uses the DODAG in the reverse direction by composing a source-route header that lists nodes down the tree (that is, in the reverse of the arrows in the figure).

Figure 2: The DODAG RPL Defined for the Tree in Figure 1.



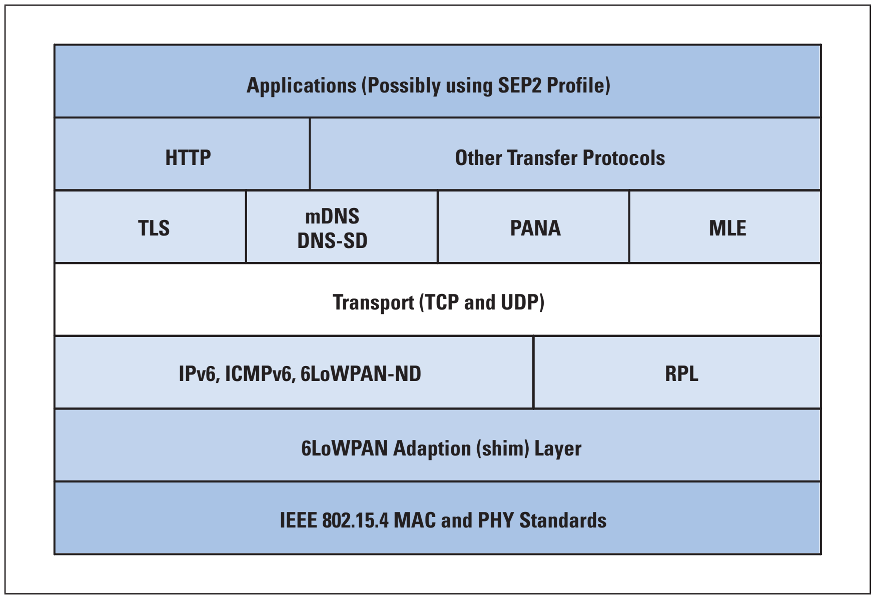
RPL separates nodes in the tree into three types: *root* (the border router acts as the root of the DODAG), *leaf* (that is, a node that has only one connection), and *intermediate* (a node that has at least two connections and forwards datagrams). Because intermediate nodes forward packets, the ZigBee IP standard uses the term *ZigBee IP router,* abbreviated *ZIP router.* Leaf nodes do not run RPL. Instead, each leaf node connects to a ZIP router, which handles all forwarding responsibilities for the leaf node. That is, a ZIP router informs the border router about each leaf node to which it connects.

RPL is much more complex than indicated here. For example, RPL can be used with storing mode; messages must specify the mode being used. In addition, RPL can distribute information down the tree. For example, it is possible to use RPL to inform nodes about the IP prefixes being used.

Other Protocols in the ZigBee IP Specification

Besides the major protocols defined previously, the ZigBee IP specification includes many other protocols. ZigBee IP uses IPv6, *Internet Control Message Protocol Version 6* (ICMPv6), TCP, *User Datagram Protocol* (UDP), *Protocol for carrying Authentication for Network Access* (PANA), *multicast DNS* (mDNS), *DNS Service Discovery* (DNS-SD), and *Transport Layer Security* (TLS), which is used in conjunction with PANA, *Extensible Authentication Protocol* (EAP), and *Extensible Authentication Protocol Transport Layer Security* (EAP-TLS) for authentication. Applications that conform to the *Smart Energy Profile 2.0*[6] use traditional HTTP as a transfer protocol. As an alternative, the IETF is defining a new protocol known as the *Constrained Application Protocol* (CoAP)[7] that is designed for constrained networks, including an 802.15.4 mesh. Figure 3 summarizes the major protocols.

Figure 3: The Layering of Major Protocols Used in a ZigBee Mesh Network


Applications (Possibly using SEP2 Profile)
HTTP
Other Transfer Protocols
TLS
mDNS DNS-SD
PANA
MLE
Transport (TCP and UDP)
IPv6, ICMPv6, 6LoWPAN-ND
RPL
6LoWPAN Adaption (shim) Layer
IEEE 802.15.4 MAC and PHY Standards

Assessment of Using IPv6 Route-Over for a Mesh

Many questions arise about the use of IPv6 and a route-over approach in a mesh of low-power wireless sensor nodes. Is a route-over design better than a mesh-under design? How much additional protocol overhead is needed for route-over? Will a route-over implementation require more memory than a mesh-under implementation? If so, how much more? Does it make sense to use IPv6 in a mesh network that has an extremely small MTU and extremely slow links? If RPL uses non-storing mode, what is the overhead in terms of additional packet forwarding? How much of IPv6 and the IPv6 support protocols must change to make them operational in the mesh environment?

As mentioned previously, the IPv6 standard specifies that IPv6 cannot be used over a network with an MTU smaller than 1280. Thus, 6LoWPAN adds a shim layer to divide a datagram into fraglets for transmission. Fraglet transmission has an advantage over conventional fragmentation: all fraglets must arrive sequentially and in order. That is, after the first fraglet reaches a receiver, subsequent frames from the sender must contain the rest of the fraglets, with no fraglets from other datagrams. Thus, if an incoming frame does not contain the expected fraglet, a receiver discards the entire datagram. The advantage of fraglets compared to traditional fragmentation lies in reduced memory usage: a receiver does not need to store buffers for a set of partial datagrams.

The chief disadvantage of the fraglet approach arises from a combination of three factors: large datagram size, an extremely small MTU, and higher probability of loss. The datagram size is especially large for datagrams sent from the border router to an individual mesh node because the extra level of encapsulation adds an IPv6 base header and a source-route header.

The result is that a minimum-size datagram (1280 octets) will be divided into 11 fraglets. If the probability of losing a given packet is p $(0 < p \leq 1)$, the probability of losing the entire datagram is much higher than p. Although the 6LoWPAN specification recognizes lossy behavior, relying on fraglet transmission can increase retransmissions (and latency).

Another issue related to MTU arises because a border router must choose an MTU for datagrams that arrive from outside the mesh. The IPv6 standard specifies a minimum link MTU of 1280. If the border router enforces an MTU of 1280 on external links, when a datagram arrives from the outside the datagram must be further encapsulated before transmission across the mesh. Unfortunately, the additional header increases the datagram size to $1280 + \delta$, making it larger than the 6LoWPAN MTU of 1280. One solution defines the 6LoWPAN MTU to be $1280 + \delta$, but requires the border router to enforce an MTU of 1280 for external sources. Unfortunately, embedding such special restrictions in IPv6 code reduces the generality—employing nonstandard techniques to handle a mesh makes the protocol stack brittle. For example, if someone invents a new path MTU discovery mechanism, the new mechanism cannot be integrated into the border router until it has been modified to honor the special MTU rules.

Designers realized that conventional IPv6 protocols cannot be used to configure a radio link or to perform a two-way signal assessment, so they created MLE. They also had to replace IPv6 Neighbor Discovery because nodes in the mesh share a single IP prefix even though they do not share a single broadcast domain. On the surface, it might appear that MLE and IPv6-ND could be modified to work together: MLE finds neighbors and IPv6-ND uses the information from MLE to maintain a list of MAC addresses for the neighbors. However, IPv6-ND also handles other tasks. For example, IPv6-ND uses ICMPv6 messages to propagate network information, including network prefixes. Unfortunately, RPL also provides a way to propagate a network prefix downward from a border router to mesh nodes. Should IPv6-ND or RPL be used? Whatever one decides, one of the two protocols must be modified to avoid having a race in which both protocols attempt to propagate address prefixes at the same time. ZigBee IP solves the problem by specifying that only 6LoWPAN-ND is to be used for propagating network configuration information.

We said that to conserve power, nodes in a ZigBee mesh can choose to sleep. Interestingly, the address-detection mechanism can require a node to spend extra power. To see why, we must know two facts. First, an address registration is assigned a lifetime interval during which the registration remains valid. Second, to conserve power, sleep mode shuts down as much hardware as possible. Thus, on a low-power node, the clock may not continue running during a sleep cycle.

Now consider what happens when a node awakens. If the sleep cycle is sufficiently long or its clock was not running, a node cannot know whether another node arrived and registered a duplicate IP address with the border router. Thus, after awakening, the node must send a message to re-register, and must receive a reply before using its address. In a conventional network, the IPv6 use of a /64 address prefix and embedded MAC addresses makes duplicate addressing unlikely. However, 802.15.4 includes a 16-bit MAC address, meaning nodes must follow the protocol to avoid duplicate addresses.

Another unexpected complication arises from mesh routing. Routing protocols used in conventional networks choose shortest paths (unless policy dictates an alternative). Mesh routing protocols must contend with multiple free variables: the signal strength of radios along the path, the length of the path, and the probability of interference. Thus, the shortest path through the mesh may not be optimal. More important, there is no easy way to estimate the probability of interference or to quantify the tradeoff between path length and signal quality. In fact, it may even be difficult to calculate the relationship between signal quality and effective throughput. Power sources can further complicate routing. For example, we can imagine a routing system that prefers nodes that obtain power from a continuous power source over nodes that obtain power from a battery. As a consequence of multiple items to optimize and no obvious objective function, mesh routing protocols can be more complex and more difficult to tune than conventional routing protocols.

Part of the inefficiency in a ZigBee mesh arises from a fundamental IPv6 design decision: an IPv6 datagram header cannot be modified after the datagram is in transit. To transit a mesh, a datagram must include a source-route header and an RPL header. However, the extra headers are relevant only within the mesh, and must be removed if a datagram leaves the mesh. If extra headers are allowed to remain, the datagram might pass across another ZigBee mesh, where the information could be misinterpreted, causing datagrams to be misrouted. Similarly, if a datagram passes from the outside into the mesh, the appropriate headers must be added. As a consequence of the IPv6 design, headers cannot be added or removed. Therefore, the only viable option is encapsulation: each IPv6 datagram sent across the mesh must be encapsulated in an IPv6 datagram that has the appropriate headers. In a network with a large MTU, IP-in-IP encapsulation adds a small overhead. With an 802.15.4 radio, however, the hardware MTU is only 127 octets. Thus, one pair of source and destination IPv6 addresses occupies more than 25% of the MTU. By contrast, a pair of IPv4 addresses accounts for just over 6% of the MTU. Unlike IPv4, which allows options to be inserted in an existing header, IPv6 requires an encapsulating header to hold a source route option. The approach requires adding multiple IPv6 addresses, which can easily generate one or more extra fraglets. As a result, the choice of IPv6 instead of IPv4 increases the overhead significantly, and makes the resulting network less efficient.

From the observations discussed previously, we conclude:

Although it is possible to use a route-over paradigm and IPv6 for an 802.15.4 ZigBee mesh, doing so means inventing alternative protocols, creating special exceptions, and incurring significant overhead.

Summary

An emerging trend focuses on an Internet of Things, in which intelligent embedded systems that sense and control their environment use Internet technology to communicate. Examples include sensors in vehicles, residences, office buildings, shopping malls, and civil infrastructure.

A consortium of vendors known as the ZigBee Alliance is specifying standards for a networking technology that uses IEEE 802.15.4 wireless network hardware to form a mesh of Smart Grid sensors. A ZigBee network has a border router that connects to the outside; other nodes in the mesh self-organize to connect and establish forwarding.

In terms of protocols, the ZigBee Alliance is working with the IETF on a route-over approach that uses IPv6. In principle, a route-over system uses IP for all forwarding. In practice, IPv6 and standard IPv6 support protocols are insufficient for a low-power, lossy wireless mesh technology that has a small MTU and a low data rate. Consequently, work has focused on replacing many parts of IPv6, adding a shim layer to accommodate small MTU, inventing a new protocol that tests signal strength and establishes links, and building a new routing protocol that constructs a forwarding tree. Even with the changes, the design of IPv6 does not match IEEE 802.15.4 radio technology well.

Acknowledgment

Material in this article has been taken with permission from [8].

References

[1] The IEEE 802.15.4 standard can be purchased from: **http://webstore.ansi.org/RecordDetail.aspx?sku=IEEE%20802.15.4-2011**

[2] Information about the ZigBee Alliance can be found at: **http://www.zigbee.org/**

[3] Information about the Internet Engineering Task force can be found at: **http://www.ietf.org/**

[4] The ZigBee IP specification can be found at: **http://zigbee.org/zigbee-for-developers/network-specifications/zigbeeip/**

[5] Stephen E. Deering and Robert M. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, December 1998.

[6] A complete list of ZigBee specifications can be found at:
`http://zigbee.org/zigbee-for-developers/applicationstandards/`
and:
`http://zigbee.org/zigbee-for-developers/network-specifications/`

[7] The specification for the Constrained Application Protocol (CoAP) can be found at:
`http://datatracker.ietf.org/doc/draft-ietf-core-coap/`

[8] Douglas E. Comer, *Internetworking with TCP/IP Volume 1: Principles, Protocols, and Architecture,* sixth edition, Prentice Hall, ISBN 0-13-608530-X.

[9] David Lake, Ammar Rayes, and Monique Morrow, "The Internet of Things," *The Internet Protocol Journal,* Volume 15, No. 3, September 2012.

[10] T. Sridhar, "Cloud Computing—A Primer: Part One," *The Internet Protocol Journal,* Volume 12, No. 3, September 2009.

[11] T. Sridhar, "Cloud Computing—A Primer: Part Two," *The Internet Protocol Journal,* Volume 12, No. 4, December 2009.

DOUGLAS E. COMER is a Distinguished Professor of Computer Science at Purdue University. Formerly, he served as VP of Research and Research Collaboration at Cisco Systems. As a member of the original IAB, he participated in early work on the Internet, and is internationally recognized as an authority on TCP/IP protocols and Internet technologies. He has written a series of best-selling technical books, and his three-volume Internetworking series is cited as an authoritative work on Internet technologies. His books, which have been translated into 16 languages, are used in industry and academia in many countries. Comer consults for industry, and has lectured to thousands of professional engineers and students around the world. For 20 years he was editor-in-chief of the journal *Software-Practice and Experience.* He is a Fellow of the ACM and the recipient of numerous teaching awards. E-mail: **`comer@cs.purdue.edu`**

Letters to the Editor

Hi Geoff,

I found a copy of *The Internet Protocol Journal,* Volume 17, No. 1, on my desk this morning and really enjoyed your article "A Question of DNS Protocols." If you have a couple of spare minutes, some questions occurred to me:

Could the number of open resolvers be due to the implementer's lack of experience and/or "open" out-of-the-box resolver configuration? If the default configuration provided by vendors was restricted (for example, for Internet Systems Consortium's *BIND* specifying **`edns-udp-size`**, **`max-udp-size`**, **`rate-limit`** and so on), do you think this restriction might slowly reduce the number of open resolvers? I'm happy to push this idea within Red Hat if you think it could be worthwhile.

To your knowledge was there any follow-up research into which clients were failing to transition to TCP queries? If it's one or two resolvers, maybe the maintainers could be engaged directly to push TCP support. Theoretically, if people start restricting their *Extension Mechanisms for DNS* (EDNS) size to 512, these clients will break anyway, correct?

Thanks again for your time,

Cheers,

—*Morgan Weetman, Red Hat Consulting*
`mweetman@redhat.com`

The author responds:

Thanks for your questions.

The basic problem we observe with these open resolvers is that there is a large-scale use of *Customer Premises Equipment* (CPE) network interface devices that include DNS resolution. Evidently, the intent was to take DNS queries from the "inside" and pass them on to the *Dynamic Host Configuration Protoco*l (DHCP)-provided "outside" DNS resolver, and cache the results that come back. The local cache provides a small increment in perceived performance on the inside, and the DNS forwarder removes the pressure of the *Network Address Translation* (NAT) function of the CPE for these *User Datagram Protocol* (UDP) transactions.

All good, but there are an annoyingly large number of units, evidently numbering in the tens of millions, where the DNS resolver function on the units has no idea what is the "inside" and what is the "outside." It will happily treat queries coming on in the "outside" and resolve these names in the same fashion as if the query was received on the "inside," and then send the response back to the "outside" query agent.

The critical configuration element that appears to be missing on these units is a filter to drop incoming packets with destination port 53 that are addressed to the exterior network interface on the unit.

With respect to your question relating to follow-up research, I should reiterate that we found that some 17% of "visible" resolvers appear not to fail over to use TCP when they receive a UDP DNS response with the truncated bit set. Now the problem with the DNS is that there is very little in the way of fingerprinting of resolvers, so apart from their IP address it's challenging to understand what is going on. A common assumption is that these units live behind a firewall that prevents TCP port 53 from traversing in either direction, but it's an assumption I've not tested for explicitly. This assumption leads to the strong suspicion that it's not DNS resolvers per se, but the environment into which they are deployed that is the problem here.

And with respect to using a small EDNS0 size field, then yes—if the response is larger than the offered size, then the responder will cram as much as it can into the offered size and set the truncate bit. The querier is meant to interpret this response as a signal to re-query over TCP, and if it fails for whatever reason, then they are unable to complete name resolution.

Thanks for your questions. I trust I've been able to answer them here.

—Geoff Huston, APNIC
`gih@apnic.net`

Geoff,

Regarding your article on DNS Protocols in IPJ Volume 17, No. 1, I have a few observations that may be of interest:

1. Blocking of TCP/Port 53 throughout the Internet, especially on endpoint networks, is a real issue. The security myth that blocking TCP/53 somehow makes DNS "more secure" by disallowing zone transfers originated sometime in the mid-1990s, and persists to this day (obviously, all that's required to disallow unauthorized zone transfers is to configure one's authoritative DNS servers properly). More than a few large-scale authoritative DNS hosters and DNS registrars who offer authoritative DNS hosting services incorrectly block TCP/53 queries to their authoritative DNS server arms. I run into all these issues with some regularity.

2. Many, many authoritative and recursive DNS servers are not scaled and tuned to support large numbers of simultaneous TCP connections, and will experience availability problems because they're overwhelmed by a comparatively small number of TCP DNS queries versus an equivalent number of UDP DNS queries. This problem also holds true of load-balancing devices that are often placed in front of both recursive and authoritative DNS server farms. I run into this problem with some regularity, as well.

3. The relative latency of TCP connection setup times combined with the practices of dynamically assembling Web pages/app views from multiple named/numbered Web servers (whether they're actually separate servers or merely additional DNS records for the same actual server) in an attempt to speed page-load times via parallelism is also a significant challenge.

The second problem is potentially resolvable (pardon the pun), but would require a high degree of capital and operational expenditure committed across many organizations to make it practical.

The third problem is a real challenge, because the designers of Web servers and apps would have to be re-educated—and they are often completely siloed from any individuals or organizations with operational experience.

The first problem is well-nigh intractable, because after filters are put into place (in this case, out of misinformed ignorance), all too often they are never removed. It's a pretty safe bet that the networks that incorrectly filter TCP/53 at this late date are never going to "see the light," so to speak.

So, while I agree that DNS over TCP would have many desirable characteristics, chief among them reducing the DNS reflection/amplification *Distributed Denial of Service* (DDoS) vector, I consider it unlikely to be practical. I first looked at this issue in 2005 when I was at Cisco, and all the problems mentioned before that applied then also apply now—in many cases with a much higher degree of prevalence than a decade ago.

Ultimately, the best solution from a number of standpoints may well be to move away from the DNS entirely towards something similar to the *Peer Name Resolution Protocol* (PNRP). I believe that more and more applications and services are going to end up being hyper-distributed among nodes we tend to think of today as "clients" (for example, mobile devices, CPE, all the various types of embedded systems)—and that because of its universal necessity and applicability, the migration of name resolution/directory services to such a model should be actively pursued.

—Roland Dobbins, Arbor Networks
rdobbins@arbor.net

The author responds:

Hi Roland,

Thanks for your comments. The speculation as to where next is certainly interesting, and the overriding consideration as to whether and how we can stay within a uniform and consistent name space while at the same time moving away from the existing DNS structure and the related resolution protocol is an aspect that greatly interests me. Again thanks for taking the time to note down your comments.

Kind regards,

—Geoff Huston, APNIC
gih@apnic.net

Book Review

Internet Peering Playbook

The Internet Peering Playbook—Connecting to the Core of the Internet, 2014 Edition, by William B. Norton, DrPeering Press, ISBN-13: 978-1937451110.

When I realized I needed to understand how Internet peering worked, it was timely that Bill Norton shared his book with me at the *North American Network Operators' Group* (NANOG) conference last summer. I read it on my 6-hour flight home and finished it the next night. Not only is it an easy read for a non-engineer in the telecom space, it is clear and concise on how peering actually works. Norton starts off with the basics on how the peering ecosystem works, who peers privately and who peers publicly, and why. He details the recent hoopla over Comcast, Netflix, and Level 3 with their public versus paid peering and simply delivers the facts and none of the emotion iterated on some of the players' blog sites.

Norton then reaches well beyond the basics of peering to include examples of "tricks of the trade." He graphically lays out samples, and explains them so well the book makes for a great read. The topic of discussion today is that some ISPs are using their access network as a monopoly, and want to charge content providers for the traffic that runs on their network. Tricks of the trade start with simple bundling options that hide additional traffic, but quickly can end up "playing chicken," where the network traffic becomes significantly asymmetric, usually resulting in one end dumping traffic on another peer's network. The network infrastructure is usually also asymmetric, resulting in a request to re-negotiate from what was a free peering situation to a pay-to-play requirement. Both sides believe the other side needs them more. Thus playing chicken is initiated.

Sometimes a new peering negotiation is made, sometimes not. If not, the result can be de-peering and possibly a severe traffic disruption. As Norton says "it really tests the assertion that both sides are receiving *equal value* from the relationship." In most cases, additional connectivity is deployed and traffic is spread across more sites to even the load.

In order to peer, traffic volumes must meet a minimum to be worth the allocation of ports. Yet "bluffing" or claiming the traffic load is adequate when it's not is one way to get a peering transaction initiated. Another way is to claim performance problems that can be easily solved by simply peering when no peering had been in place previously. Because coordinators rarely have time for in-depth traffic analysis, this type of peering becomes another trick of the trade to gain free peering, at least in the short term.

Be open, be loud, be a friend, and be sweet are all positive routes for proper peering techniques. Negative approaches such as *Make It Long and Difficult* (MILD) are used, where discussions are prolonged and appear open but nothing really happens. Norton even brings up peering tactics that don't work, such as trying to dominate in a single foreign market, public badgering (I assume at NANOG events), holding content hostage, sending blind requests, or simply lacking knowledge in the peering backbone space is enough to be shunted in this tight community.

A recent report from *Measurement* Lab (M-Lab) shows "sustained performance degradation for access customers when traversing interconnections" and displays proof in numbers that peering degradation occurs. The report explains very clearly that the traffic degradation is due to the business relationships of the interconnections, and is not at all the fault of technical problems[1].

What the Internet peering community does not want is for the *Federal Communications Commission* (FCC) to try to regulate this market. However, the FCC is beginning to take steps to criticize ISPs for "throttling" traffic. As our world revolves around increasing access to bandwidth, not everyone needs to understand the importance of Internet peering, but it surely is interesting!

—Eve Griliches
egriliches@btisystems.com

[1] **http://www.measurementlab.net/static/observatory/M-Lab_Interconnection_Study_US.pdf**

Read Any Good Books Lately?
Then why not share your thoughts with the readers of IPJ? We accept reviews of new titles, as well as some of the "networking classics." In some cases, we may be able to get a publisher to send you a book for review if you don't have access to it. Contact us at **ipj@protocoljournal.org** for more information.

MANRS: Improving Global Routing Security & Resilience

Most end users don't give much—if any—thought to things like the Internet's global routing system because, for the most part, the Internet has *just worked* for years. However, in several instances vulnerabilities in the security and resilience of that routing system have manifested themselves: a 2008 incident that made *YouTube* temporarily unreachable around the globe, multiple cases of Internet traffic deflection by some Chinese Internet Service Providers, and an April 2014 incident in which an Indonesian network operator mistakenly claimed that it "owned" many of the world's networks, just to name a few.

Internet security, in general, is a difficult area when it comes to incentivizing network operators to act with the good of the whole Internet in mind, and security of the global Internet infrastructure, be it the *Domain Name System* (DNS) or routing, brings additional challenges because the utility of security measures depends on coordinated actions of many other parties. Thus, while technology is an essential element, technology alone is not sufficient. To stimulate visible improvements across the entire Internet, we need a culture of collective responsibility and action.

The good news is that throughout the history of the Internet, we've seen amazing feats of collaboration among participants and shared responsibility for its smooth operation. Collaboration and shared responsibility are two of the pillars supporting the Internet's tremendous growth and success, as well as its overall security and stability.

So, how can we collectively help improve the security and resilience of the global Internet routing system and prevent the kinds of vulnerabilities we mentioned earlier? One of the approaches is the *Routing Resilience Manifesto* initiative, which features the *Mutually Agreed Norms for Routing Security* (MANRS) document. This initiative of several leading network operators, supported and coordinated by the Internet Society, was publicly launched on November 6, 2014.

MANRS captures the collaborative spirit that has been a hallmark of the Internet's growth, and provides motivation and guidance to network operators in addressing issues of security and resilience of the global Internet routing system.

MANRS defines a compact and clear set of actions that network operators should implement to improve routing security on their own networks and across the Internet as a whole; specifically:

- Prevent propagation of incorrect routing information.
- Prevent traffic with spoofed source IP addresses.
- Facilitate global operational communication and coordination between network operators.
- Facilitate validation of routing information on a global scale.

The Routing Resilience Manifesto is more than just the MANRS document; it is a commitment to improve the global Internet. In order to become a participant of this initiative, a network operator has to implement one or more of the identified actions.

More than a dozen network operators around the world have already signed up, and we expect more to join. You can learn more about the effort and sign up to be part of this exciting movement to make the Internet a safer place for everyone at `www.routingmanifesto.org`

IAB Statement on Internet Confidentiality

The *Internet Architecture Board* (IAB) issued the following statement on November 14, 2014:

In 1996, the IAB and *Internet Engineering Steering Group* (IESG) recognized that the growth of the Internet depended on users having confidence that the network would protect their private information. RFC 1984[1] documented this need. Since that time, we have seen evidence that the capabilities and activities of attackers are greater and more pervasive than previously known. The IAB now believes it is important for protocol designers, developers, and operators to make encryption the norm for Internet traffic. Encryption should be authenticated where possible, but even protocols providing confidentiality without authentication are useful in the face of pervasive surveillance as described in RFC 7258[2].

Newly designed protocols should prefer encryption to cleartext operation. There may be exceptions to this default, but it is important to recognize that protocols do not operate in isolation. Information leaked by one protocol can be made part of a more substantial body of information by cross-correlation of traffic observation. There are protocols which may as a result require encryption on the Internet even when it would not be a requirement for that protocol operating in isolation.

We recommend that encryption be deployed throughout the protocol stack since there is not a single place within the stack where all kinds of communication can be protected.

The IAB urges protocol designers to design for confidential operation by default. We strongly encourage developers to include encryption in their implementations, and to make them encrypted by default.

We similarly encourage network and service operators to deploy encryption where it is not yet deployed, and we urge firewall policy administrators to permit encrypted traffic.

We believe that each of these changes will help restore the trust users must have in the Internet. We acknowledge that this will take time and trouble, though we believe recent successes in content delivery networks, messaging, and Internet application deployments demonstrate the feasibility of this migration. We also acknowledge that many network operations activities today, from traffic management and intrusion detection to spam prevention and policy enforcement, assume access to cleartext payload. For many of these activities there are no solutions yet, but the IAB will work with those affected to foster development of new approaches for these activities which allow us to move to an Internet where traffic is confidential by default.

[1] IAB and IESG, "IAB and IESG Statement on Cryptographic Technology and the Internet," RFC 1984, August 1996.

[2] Stephen Farrell and Hannes Tschofenig, "Pervasive Monitoring Is an Attack," RFC 7258, May 2014.

Upcoming Events

The *North American Network Operators' Group* (NANOG) will meet in San Antonio, Texas February 2–4, 2015; and in San Francisco, California, June 1–3, 2015. See: `http://nanog.org`

The *Internet Corporation for Assigned Names and Numbers* (ICANN) will meet in Singapore, February 8–12, 2015; in Buenos Aires, Argentina, June 21–25, 2015; and in Dublin, Ireland, October 18–22, 2015. See: `http://icann.org`

The *Asia Pacific Regional Internet Conference on Operational Technologies* (APRICOT) will be held in Fukoka, Japan, February 24–March 6, 2015. See: `http://www.apricot.net`

The *Internet Engineering Task Force* (IETF) will meet in Dallas, Texas, March 22–27, 2015; in Prague, Czech Republic, July 19–24, 2015; and in Yokohama, Japan, November 1–6, 2015.
See: `http://www.ietf.org/meeting/`

The Internet Protocol Journal is published under the "CC BY-NC-ND" Creative Commons Licence. Quotation with attribution encouraged.

This publication is distributed on an "as-is" basis, without warranty of any kind either express or implied, including but not limited to the implied warranties of merchantability, fitness for a particular purpose, or non-infringement. This publication could contain technical inaccuracies or typographical errors. Later issues may modify or update information provided in this issue. Neither the publisher nor any contributor shall have any liability to any person for any loss or damage caused directly or indirectly by the information contained herein.

Supporters and Sponsors

The Internet Protocol Journal (IPJ) is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol. Publication of IPJ is made possible by:

Supporters

Internet Society™ CISCO™

Diamond Sponsors

Google™ ICANN

Ruby Sponsor

RIPE NCC

Sapphire Sponsors

TEAM CYMRU INSIGHT THAT IMPROVES LIVES

VERISIGN™

Emerald Sponsors

Afilias℠ apia APNIC .auDA .au DOMAIN ADMINISTRATION LTD

CISCO™ COMCAST DNS-OARC Dyn℠

JUNIPer NETWORKS NSRC Network Startup Resource Center WIDE PROJECT 世纪互联® www.21vianet.com

Corporate Subscriptions

amsix amsterdam internet exchange

ISC Internet Systems Consortium

Limelight NETWORKS

netnod

SIDN

Individual Sponsors

Lyman Chapin, Steve Corbató, Dave Crocker, Jay Etchings, Hagen Hultzsch, Dennis Jennings, Jim Johnston, Merike Kaeo, Bobby Krupczak, Richard Lamb, Tracy LaQuey Parker, Bill Manning, Mike O’Connor, Tim Pozar, George Sadowsky, Helge Skrivervik, Rob Thomas, Tom Vest, Rick Wesson.

For more information about sponsorship, please contact `sponsor@protocoljournal.org`

The Internet Protocol Journal
NMS
535 Brennan Street
San Jose, CA 95131

ADDRESS SERVICE REQUESTED

The Internet Protocol Journal
Ole J. Jacobsen, Editor and Publisher

Editorial Advisory Board

Fred Baker, Cisco Fellow
Cisco Systems, Inc.

Dr. Vint Cerf, VP and Chief Internet Evangelist
Google Inc, USA

Dr. Steve Crocker, Chairman
Internet Corporation for Assigned Names and Numbers

Dr. Jon Crowcroft, Marconi Professor of Communications Systems
University of Cambridge, England

Geoff Huston, Chief Scientist
Asia Pacific Network Information Centre, Australia

Olaf Kolkman, Chief Internet Technology Officer
The Internet Society

Dr. Jun Murai, Founder, WIDE Project, Dean and Professor
Faculty of Environmental and Information Studies,
Keio University, Japan

Pindar Wong, Chairman and President
Verifi Limited, Hong Kong

The Internet Protocol Journal is published quarterly and supported by the Internet Society and other organizations and individuals around the world dedicated to the design, growth, evolution, and operation of the global Internet and private networks built on the Internet Protocol.

Email: ipj@protocoljournal.org
Web: www.protocoljournal.org

The title "The Internet Protocol Journal" is a trademark of Cisco Systems, Inc. and/or its affiliates ("Cisco"), used under license. All other trademarks mentioned in this document or website are the property of their respective owners.

Printed in the USA on recycled paper.

BY NC ND