The Internet Protocol Journal

March 2015 Volume 18, Number 1

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From the Editor 1

Scaling the Root..... 2

Gigabit Ethernet 20

Fragments 33

Call for Papers 34

Supporters and Sponsors 35

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

ISSN 1944-1134

From The Editor

In the summer of 1984 I joined SRI International in Menlo Park, California, working at the *Network Information Center* (NIC). The NIC provided numerous services relating to the ARPANET and MILNET, including a telephone help line, various printed materials, login credentials for dialup users, and the all-important **HOSTS.TXT** file that mapped host names to their corresponding IP addresses. The **HOSTS.TXT** file was updated once a week and made available via FTP and from the NIC's dedicated name server. The original documents that described the *Domain Name System* (DNS) had been published in late 1983, and all of us at the NIC were keenly aware that a transition from a centrally maintained file to a distributed and hierarchical name resolution system would soon be underway. The original design of the DNS has proven itself to be both robust and scalable, and the protocol has been enhanced to support IPv6, as well as security (DNSSEC). In our first article, Geoff Huston gives an overview of the DNS and discusses possible ways in which to further scale the system.

Ethernet has been a critical component of *Local-Area Networks* (LANs) for many decades. As with most networking technologies, there have been several iterations of the Ethernet standards, each providing orders of magnitude faster transmission rates. In our second article, William Stallings gives an overview of recent developments and standardization efforts for *Gigabit Ethernet*.

If you received a printed copy of this journal in the mail, you should also have received a subscription activation e-mail with information about how to update and renew your subscription. If you didn't receive such a message, it may be because we do not have your correct e-mail address on file. To update and renew your subscription, just send a message to **ipj@protocoljournal.org** and include your subscription ID. Your subscription ID is printed on the back of your journal.

Let me remind you that IPJ relies on the support of numerous individuals and organizations. If you or your company would like to sponsor IPJ, please contact us for further details. Our website at **protocoljournal.org** contains all back issues, subscription information, a list of current sponsors, and much more. Our first blog entry "Notes from NANOG 63," was recently posted on the website.

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

Scaling the Root

by Geoff Huston, APNIC

The *Domain Name System* (DNS) of the Internet is a modern-day miracle that has proved to be exceptionally prodigious. This technology effectively supported the operation of the Internet from a scale of a few hundred thousand users to today's system of some 3 billion users and an estimated 8 to 10 billion devices. Not only has it supported that level of growth, it has done so without any obvious cracks within the basic protocol. But that point should not imply that nothing has changed in the DNS protocol over the past 30 years. While the basic architecture of the DNS as a simple query/response protocol has remained consistent over this period, we have adorned the base query/response protocol with various optional "bells and whistles" that are intended to improve its robustness, and we have constantly updated the platform infrastructure of the DNS to cope with ever-increasing query loads as the Internet expands. In this article we look at one aspect of this effort: the current considerations of how to scale the root of the DNS.

An Introduction to the DNS

The DNS is a hierarchically distributed naming system. The inherent utility of a name system in a digital network lies in an efficient and consistent mapping function between *names* and *IP addresses*. With the progenitor of the DNS, this mapping function took the form of a single file (`HOSTS.TXT`) that listed all the active host names and the corresponding IP address for each address that resided on each host. The problem with this approach was the coordination of entries in this hosts file so that all hosts had a consistent view of the name space of the network. As the network grew, the administrative burden of coordinating this burgeoning name space became unworkable. The DNS replaced this replicated file with a dynamic query system that allowed hosts to query a distributed name data base and retrieve the current value of the mapped IP address.

To achieve this goal, the name system uses a hierarchical structure. A name in the DNS is a sequence of labels that scan from left to right. This sequence of labels can be viewed in a pairwise fashion, where the label to the left is the "child" of the "parent" label to the right. The apex of this hierarchical name structure is the root, which is notionally defined as the trailing "." at the rightmost part of a *Fully Qualified Domain Name* (FQDN).

As a name-resolution system, the DNS is constructed as a collection of agents that ask questions and receive answers, so called *Resolvers,* and a set of servers that can provide the authoritative answer for a certain set of questions, *Authoritative Name Servers*. A set of name servers that are configured as being authoritative for a given *zone* should provide identical answers in response to queries for names that lie within this zone.

The task of a resolver is to ask questions of servers. But if each server is able to provide answers for only specific zones, the question then becomes: which server to ask?

For example, to resolve the name **www.example.com.**, a resolver needs to find the authoritative name servers for the zone **example.com**. This information (the set of authoritative name servers for a zone) is stored as part of a zone delegation record that is loaded into the parent zone. In our example, to resolve **www.example.com.**, a resolver needs to query any of the servers for the **example.com.** zone. These servers can be found by querying any of the authoritative name servers for the **com.** zone. But to do that a resolver needs to know who are the authoritative name servers for the **com.** zone. This information is held in the Root Zone, and can be retrieved by sending a query to any of the *Root Zone Servers*.

The way this process is implemented in the DNS is that when an authoritative name server is queried for a name that lies within the scope of a delegated child of the zone of the server, the server will respond to the query not with the desired answer, but with the set of authoritative name servers for the immediate child zone that encompasses the name of the query.

Therefore, when a recursive resolver is passed a query relating to a name about which it has no knowledge, it will first send the query for this name to a root server. The response is not the desired information, but the name servers that are authoritative for the top-level domain name being queried. The recursive resolver will then query one of these name servers for the same name, and will receive in response the name servers that are authoritative for the next level of domain name, and so on. For example, a resolver with no a prioi knowledge of the DNS other than a list of servers for the root zone, when attempting to resolve the name **www.example.com.**, will first query one of the root servers for this name. The response of the root server should be the set of authoritative name servers for the **.com** zone. The resolver will next query one of these servers with the same query. The response will be the set of servers for the **example.com.** zone. When one of these servers is queried, it should respond with the desired *Resource Record* (such as the mapped IP address of the name).

To make the DNS work efficiently, resolvers typically remember these responses in a local cache, and will not re-query for the same information unless the cached entry has timed out in the local cache. For example, a subsequent query for **ftp.example.com**. would use the local cache of the resolver to select the set of authoritative name servers for **example.com** and pose this query directly to one of these servers.

So resolvers can dynamically discover and cache all aspects of the DNS, with one critical exception. The root zone of the DNS cannot be dynamically discovered in this manner, because it has no parent. To get around this problem, DNS resolvers are configured with a *root hints* file, which contains a list of IP addresses of those name servers that are authoritative for the root zone. When a resolver starts up, it sends a *root priming query* to one of the servers listed in the hints file, requesting the current set of root name servers. In this way the resolver then is primed with the current set of root name servers.

What are root servers used for?

As explained previously, one intended role of the root servers is to respond to root priming queries. Secondly, as previously explained, the root servers respond to queries relating to labels that are defined in the root zone, and also respond negatively to queries relating to labels that are not defined in the root zone. Resolvers establish the identity of name servers for those names that are at the *top level* of the DNS name hierarchy by directing a query to a root server.

Obviously, if every name-resolution attempt involved a resolver making a query to the root name servers, then the DNS root servers would have melted under the consequent load years ago! Resolvers reduce this load by caching the answers they receive, so that the profile of queries set to the root are dominated by queries for "new" names that resolvers have not previously seen (and cached). Typically, the response from the root name server is one that indicates that the name does not exist in the root zone. Given that the overwhelming role of the root servers is to reply with a "no such name" response, then it would seem that the role of the root server is somewhat inconsequential. However, resolvers do not keep a permanent copy of the responses they receive from their queries for names that exist in the root zone. They use a timed local cache, and from time to time the resolver needs to repeat the query in order to refresh the cache entry. If the root servers disappeared, these refresh queries would fail and the resolver would be unable to answer further queries about this zone. So while the predominate load on the root servers is to respond to junk queries, their continuing availability is of paramount importance to the Internet. And if the resolvers of a network were isolated from the root server constellation, then the network would, over a short period of time, cease to have a working name system.

This situation would lead us to the thought that if root servers are so critical, then a single host serving the root zone for the entire Internet would be a poor design choice. Perhaps every network should run a root server. This thought touches on numerous considerations, not the least of which includes considerations of the underlying query and response protocol that the DNS uses.

DNS Protocol Considerations

The DNS is a simple query/response protocol. In order to allow a query agent to recognize the appropriate response, the response is a copy of the original query, with additional fields included.

The two mainstream IP transport protocols are the *User Datagram Protocol* (UDP) and the *Transmission Control Protocol* (TCP). TCP is ill-suited to simple query/response transactions, given that each TCP connection requires an initial packet exchange to open a connection, and a closing exchange, and while the transaction is open the server needs to maintain TCP session context. UDP eschews this overhead, and each query can be loaded into a single UDP packet, as can the corresponding response. Although TCP is a permitted transport protocol for DNS, the overwhelming operational preference is to use UDP. It's fast, efficient, and works well. But that's not quite the entire story. We need to go down one level of the protocol stack and look at the IP protocol.

The *IPv4 Host Requirements Specification*[1] mandates that all compliant IP host systems be able to accept and process an IP packet that is at least 576 octets. Individual IP fragments may be smaller than this number, but the host must be able to reassemble the original IP packet if it is 576 bytes or less. The consequence of this requirement is that compliant hosts need not necessarily accept an IP packet larger than 576 octets, but they will all accept packets of this size or smaller.

Allowing for 20 octets of the IP header and a maximum of 40 octets of IP header options, 516 octets of payload remain. UDP headers are 8 octets, so we would expect that if we were to define a protocol that used UDP as its transport protocol, then a UDP payload of a maximum of 508 octets of payload would be assured to reach any IPv4-compliant host. However, the DNS specification is subtly mismatched, and the *DNS Specification*[2] specifies a DNS payload size of 512 octets or less.

So how many distinct root name servers can be listed in a priming response if we want to keep the response size to 512 octets or less? The adoption of 13 distinct root servers is a compromise between these two pressures. The exact number was the outcome of the number of distinct root server labels, and their IPv4 addresses, that can be loaded into an unsigned IPv4 UDP response to a root server priming query that is less than 512 octets. An example of a root priming response is shown in Figure 1 on page 6. (The DNS is a binary protocol that uses field compression where possible. The *dig* utility[3] generates specific DNS queries and produces a text representation of the response.)

This response is 503 octets, which will also just fit into the 512-byte limit as defined by the *Applications Requirements Specification*[4].

Figure 1: DNS Root Priming Response

```
$ dig +bufsize=512 +norecurse . NS @a.root-servers.net.

; <<>> DiG 9.9.6 <<>> +bufsize=512 +norecurse . NS @a.root-servers.net.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 38316
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 16

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1472
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.

;; ADDITIONAL SECTION:
a.root-servers.net.     518400  IN      A       198.41.0.4
b.root-servers.net.     518400  IN      A       192.228.79.201
c.root-servers.net.     518400  IN      A       192.33.4.12
d.root-servers.net.     518400  IN      A       199.7.91.13
e.root-servers.net.     518400  IN      A       192.203.230.10
f.root-servers.net.     518400  IN      A       192.5.5.241
g.root-servers.net.     518400  IN      A       192.112.36.4
h.root-servers.net.     518400  IN      A       128.63.2.53
i.root-servers.net.     518400  IN      A       192.36.148.17
j.root-servers.net.     518400  IN      A       192.58.128.30
k.root-servers.net.     518400  IN      A       193.0.14.129
l.root-servers.net.     518400  IN      A       199.7.83.42
m.root-servers.net.     518400  IN      A       202.12.27.33
a.root-servers.net.     518400  IN      AAAA    2001:503:ba3e::2:30
b.root-servers.net.     518400  IN      AAAA    2001:500:84::b

;; Query time: 145 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Sun Mar 01 10:45:46 UTC 2015
;; MSG SIZE  rcvd: 503
```

Evolutionary Pressures

These days it's rare to see a host impose a maximum IP datagram size of 576 octets. A more common size of IP datagrams is 1,500 octets, as defined by the payload size of Ethernet frames. In theory, an IPv4 datagram can be up to 65,535 octets, and in IPv6 the jumbo payload option allows for an IPv6 datagram of some 4 billion octets, but both of these upper bounds are very much theoretical limits.

More practical limits can be found in the unstandardized work to support large packets in 802.3 networks, where the value of 9,000 octets for *Maximum Transmission Unit* (MTU) sizes is sometimes found on vendors' IEEE 802.3 equipment for Gigabit Ethernet[11]. However, there are two problems with this 9,000 octet packet size. Not all networks support the transmission of 9,000 octet packets without resorting to packet fragmentation, and there are classes of security middleware that reject all packet fragments on the basis of their assumed security risk. So a 1,500-octet value appears to be a practical assumption for the maximum size of an unfragmented IP datagram that has a reasonable probability of being passed through IP networks. This is a useful assumption for the root priming response, as it is now larger than 508 (or even 512) octets by default.

IPv6

The transition of the Internet to use IPv6 implies a protracted period of support for both IP protocols, and the root server priming response is no exception to this implication. When we add the IPv6 addresses of the 13 root name servers to the packet, the size of the response expands to 755 octets, as shown in Figure 2 on page 8. (As is shown in the response, the E root server does not have an IPv6 address.)

DNSSEC

The next change has been in the adoption of *Domain Name System Security Extensions* (DNSSEC), used to sign the root zone[5]. The priming response now needs to contain the digital signature of the *Name Server* (NS) records, which expands the root priming response by a further 158 octets (Figure 3 on page 9).

Figure 2: DNS Root Priming Response with IPv6 Addresses

```
dig +norecurse . NS @a.root-servers.net.

; <<>> DiG 9.9.6 <<>> +norecurse . NS @a.root-servers.net.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 56567
;; flags: qr aa; QUERY: 1, ANSWER: 13, AUTHORITY: 0, ADDITIONAL: 25

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 4096
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.

;; ADDITIONAL SECTION:
e.root-servers.net.     3600000 IN      A       192.203.230.10
h.root-servers.net.     3600000 IN      A       128.63.2.53
h.root-servers.net.     3600000 IN      AAAA    2001:500:1::803f:235
b.root-servers.net.     3600000 IN      A       192.228.79.201
b.root-servers.net.     3600000 IN      AAAA    2001:500:84::b
j.root-servers.net.     3600000 IN      A       192.58.128.30
j.root-servers.net.     3600000 IN      AAAA    2001:503:c27::2:30
c.root-servers.net.     3600000 IN      A       192.33.4.12
c.root-servers.net.     3600000 IN      AAAA    2001:500:2::c
a.root-servers.net.     3600000 IN      A       198.41.0.4
a.root-servers.net.     3600000 IN      AAAA    2001:503:ba3e::2:30
g.root-servers.net.     3600000 IN      A       192.112.36.4
l.root-servers.net.     3600000 IN      A       199.7.83.42
l.root-servers.net.     3600000 IN      AAAA    2001:500:3::42
i.root-servers.net.     3600000 IN      A       192.36.148.17
i.root-servers.net.     3600000 IN      AAAA    2001:7fe::53
m.root-servers.net.     3600000 IN      A       202.12.27.33
m.root-servers.net.     3600000 IN      AAAA    2001:dc3::35
f.root-servers.net.     3600000 IN      A       192.5.5.241
f.root-servers.net.     3600000 IN      AAAA    2001:500:2f::f
d.root-servers.net.     3600000 IN      A       199.7.91.13
d.root-servers.net.     3600000 IN      AAAA    2001:500:2d::d
k.root-servers.net.     3600000 IN      A       193.0.14.129
k.root-servers.net.     3600000 IN      AAAA    2001:7fd::1

;; Query time: 144 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Sun Mar 01 10:50:01 UTC 2015
;; MSG SIZE  rcvd: 755
```

Figure 3: DNS Root Priming Response with DNSSEC Signature

```
dig +norecurse +dnssec . NS @a.root-servers.net.

; <<>> DiG 9.9.6 <<>> +norecurse +dnssec . NS @a.root-servers.net.
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 52686
;; flags: qr aa; QUERY: 1, ANSWER: 14, AUTHORITY: 0, ADDITIONAL: 25
;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;.                              IN      NS

;; ANSWER SECTION:
.                       518400  IN      NS      e.root-servers.net.
.                       518400  IN      NS      m.root-servers.net.
.                       518400  IN      NS      a.root-servers.net.
.                       518400  IN      NS      k.root-servers.net.
.                       518400  IN      NS      b.root-servers.net.
.                       518400  IN      NS      h.root-servers.net.
.                       518400  IN      NS      d.root-servers.net.
.                       518400  IN      NS      f.root-servers.net.
.                       518400  IN      NS      l.root-servers.net.
.                       518400  IN      NS      j.root-servers.net.
.                       518400  IN      NS      i.root-servers.net.
.                       518400  IN      NS      g.root-servers.net.
.                       518400  IN      NS      c.root-servers.net.
.                       518400  IN      RRSIG   NS 8 0 518400 20150311050000
20150301040000 16665 . 1QPFbaT+1QHnYWO6yyFvLT2JD7qddTFcRxFao1Gp+CysxaZSQ
LydQtPA q3PVaKCpIkYfaFgGrOyibkkMD+nFfBxFgh/0YZN9q984NUM6LBVjpfrA MVhLy6/
qDWssDn48HoO94RwdZPzdyz+T4/KIsyH5h2FL2kp9RF1tjKlE eUU=

;; ADDITIONAL SECTION:
e.root-servers.net.     3600000 IN      A       192.203.230.10
m.root-servers.net.     3600000 IN      A       202.12.27.33
m.root-servers.net.     3600000 IN      AAAA    2001:dc3::35
a.root-servers.net.     3600000 IN      A       198.41.0.4
a.root-servers.net.     3600000 IN      AAAA    2001:503:ba3e::2:30
k.root-servers.net.     3600000 IN      A       193.0.14.129
k.root-servers.net.     3600000 IN      AAAA    2001:7fd::1
b.root-servers.net.     3600000 IN      A       192.228.79.201
b.root-servers.net.     3600000 IN      AAAA    2001:500:84::b
h.root-servers.net.     3600000 IN      A       128.63.2.53
h.root-servers.net.     3600000 IN      AAAA    2001:500:1::803f:235
d.root-servers.net.     3600000 IN      A       199.7.91.13
d.root-servers.net.     3600000 IN      AAAA    2001:500:2d::d
f.root-servers.net.     3600000 IN      A       192.5.5.241
f.root-servers.net.     3600000 IN      AAAA    2001:500:2f::f
l.root-servers.net.     3600000 IN      A       199.7.83.42
l.root-servers.net.     3600000 IN      AAAA    2001:500:3::42
j.root-servers.net.     3600000 IN      A       192.58.128.30
j.root-servers.net.     3600000 IN      AAAA    2001:503:c27::2:30
i.root-servers.net.     3600000 IN      A       192.36.148.17
i.root-servers.net.     3600000 IN      AAAA    2001:7fe::53
g.root-servers.net.     3600000 IN      A       192.112.36.4
c.root-servers.net.     3600000 IN      A       192.33.4.12
c.root-servers.net.     3600000 IN      AAAA    2001:500:2::c

;; Query time: 145 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Sun Mar 01 10:51:04 UTC 2015
;; MSG SIZE  rcvd: 913
```

When the 576-octet boundary is crossed, it seems that any response up to 1,432 octets would have an equal probability to be supported by all DNS resolvers, so there is a case to be made that the 13 root name servers could be expanded to a slightly larger set of 14 or 15 servers and have the root priming response sit within 1,432 octets. But perhaps that is not quite the case, because the root priming response may yet need to grow further. The consideration here is the question of how to perform a rollover of the keys used to sign the root zone, and it would be prudent to leave some additional space in the response to allow for the use of a second digital signature. It would also be prudent to allow for a slightly larger key size, given the overall shift to longer keys over the past couple of decades of cryptography. The consequence is that it's unlikely that a further one or two root name servers would really be feasible.

Even Larger Responses?

Why not go over this limit and allow the response to be fragmented? After all, *Extension Mechanisms for DNS* (EDNS0)[6] allows for a resolver to inform the server of the maximum-size DNS payload that it can reassemble. The EDNS0 specification suggests a size of 4,096 as a "good compromise," and it appears that many resolvers have followed this advice.

However, as we've already noted, the problem is that when a datagram exceeds 1,500 octets, it usually has to be fragmented within the network. In IPv4, fragmentation is not uncommon, but at the same time many security firewalls regard the admission of fragments as a security risk, and the silent discarding of fragments has been observed. In IPv6 the handling of UDP fragmentation is quite different. The gateway has to send an *Internet Control Message Protocol Version 6* (ICMPv6) message back to the packet originator, and the host will then inscribe an entry in its own IP forwarding table with the revised MTU size. Subsequent UDP responses to this destination address will then use this revised MTU size, but the original response is irretrievably lost. Many aspects of this behavior are prone to error, including the use of IPv6 privacy addresses, the filtering of incoming ICMPv6 messages, the finite size of the IPv6 forwarding table, and the vulnerability of the server to spoofed ICMPv6 messages.

So while responses larger than 1,500 octets are feasible, operationally it would be prudent to limit the size of the root zone priming response to be less than 1,432 octets in IPv4. Playing it cautiously with response to packet fragmentation in IPv6 would further reduce this upper bound to 1,232 octets, because the IPv6 specification defines 1,280 as the minimum packet size that will assuredly be passed through an IPv6 network without fragmentation.

More Root Servers?

Operating just 13 root name servers is not enough for the Internet, and it has not been for many years. Adding a further 1 or 2 new root server instances was never going to change that situation, given that the demand is for many thousands of new root server instances, even if the information for these additional servers could be packed into an unfragmented root priming response.

However, one aspect of the DNS service architecture can be usefully exploited. While every resolver has to reach at least one root name server at all times, no resolver has to reach every instance of the set of root name servers at all times.

What this reality implies is that the demands of scaling the root service in the face of an expanding network can be addressed through the adoption of *anycast* clouds for many of the root name servers.

What Is "Anycast?"

Normally it is considered to be a configuration error for two or more distinct hosts to share a common IP address. However, there are times when this feature can be usefully exploited to support a highly robust service with potentially high performance. The essential prerequisite is that every instance of an anycast constellation will respond in precisely the same manner to a given input. This way it does not matter which instance of a set of host servers receives the query; the response will be the same. In an anycast scenario, the routing system essentially segments the network so that all end points that are "close" to one instance of a host within the anycast service set will have their packets directed to one host, while other end points will be directed by the routing system to use other hosts.

Anycast is most effective when using a stateless simple query/response protocol, such as DNS over UDP. However, anycast can be supported when using a TCP transport protocol, although care should be taken to ensure that the TCP sessions are relatively short in direction and that routing instability is minimized, because the redirection of packets in the middle of a TCP session to a different host in the anycast set will cause the TCP session to fail. Some operational considerations of anycast services are documented in *Operation of Anycast Services*[7].

The anycast structure has numerous major attributes that help the root server system. The first is that the multiple instances of the root server instance split up the query load against the IP address of that server into the localities served by each anycast instance. This scenario allows the root service instance to distribute its load, improving its service. Equally, it allows the root server instance to appear to be "close" to many disparate parts of the client base simultaneously, also contributing to an improvement in its service profile. This technique also allows the root server to cope with various forms of denial-of-service attacks. Wide-scale distributed attacks are spread across multiple server instances, implying that a greater server capacity is deployed to absorb the attack.

Point attacks from a small set of sources are pinned against a single server instance, minimizing the collateral damage from such an attack to a single instance of the anycast server set.

Although anycast has considerable capability and has enjoyed operational success in recent years, there are still some problems with its operational behavior. The major problem is that this environment is still "controlled," and each anycast instance is, in effect, operated by the anycast service root name server operator. You can't just spin up your own instance of a root server and expect that it will engender the same level of trust in the integrity of its operation as a duly controlled and managed instance of a root service.

But why not?

Why can't we arbitrarily expand the root service in the Internet to a level well beyond these 13 root service operators? Can we admit the concept of an "uncontrolled" root zone where anyone can offer a root zone resolution service?

Scaling the Root

There have been a couple of recent Internet Drafts on potential ways to further scale the service of the root zone that does not require the explicit permission of an existing root zone operator.

The first of these drafts is a proposal to operate a root slave service on the local loopback interface[8] of a resolver. This approach is not an architectural change to the DNS (or at least not intentionally). For recursive resolvers that implement this approach, this approach is a form of change in query behavior because a recursive resolver so configured will no longer query the root servers, but instead direct these queries to a local instance of a slave server that is listening on the recursive resolver loopback address. This slave server is serving a locally held instance of the root zone, and the recursive resolver would perform DNSSEC validation of responses from this local slave to ensure the integrity of responses received in this manner. For users of this recursive resolver, there is no apparent change to the DNS or to their local configurations.

The motivation behind this proposal is that a population of recursive resolvers is still too far away from all of the root servers, and this situation causes delays in the DNS resolution function. The caching properties of recursive DNS resolvers is such that the overall majority of queries directed to the root servers are for nonexistent top-level domains, so a pragmatic restatement of the problem space is that there are recursive resolvers that take too long to generate a *Non-existent Domain Name* (NXDOMAIN) response, and this approach would reduce this time delay.

However, given this particular formulation of the problem space, then the larger and more comprehensive the anycast constellations of the root servers, the less the demand for this particular approach.

Locales where there are adequately close DNS root services from the anycast root servers would find no particular advantage in operating a local slave DNS root server because the marginal speed differential may not be an adequate offset for the added complexity of configuration and operation of the local slave server.

The linking of the root zone information to the loopback is a point of fragility in the setup. Setting up a slave DNS server that is authoritative for the root zone would require using multiple root servers to ensure that it has access to a root zone from at least one of the anycast server constellations at any time. If at any time it cannot retrieve a master copy of the root zone, it should respond with a SERVFAIL (server failure) code, and the local recursive resolver should interpret this response as a signal to revert to conventional queries against the root servers.

This proposal provides integrity in the local root server through the mechanism of having the recursive resolver perform DNSSEC validation against the responses received from the local root slave. If the recursive resolver is configured as a DNSSEC-validating resolver, then it is configurable on current implementations of DNS recursive resolvers. However, if it is desired to limit DNSSEC validation to just the responses received from the local slave root server, then this configuration is not within the current capabilities of the more widely used DNS resolver implementations today.

The advantages of this approach is that the decision to set up a local slave root server is one that is entirely local to the recursive resolver, and the impacts of this decision affect only the clients of this recursive resolver. No coordination with the root server operators is required, nor is any explicit notification. The local slave server is only indirectly visible to the clients of this recursive resolver and no other.

A second proposal is slightly more conventional in that it proposes adding a new anycast root server constellation to the DNS root, but instead of adding a new entry to the existing root server set, it proposes a second root server hints file.[9]

One possible motivation behind this proposal lies in the observation that before the root was DNSSEC-signed, we placed much reliance in the concept that the root zone was served from only a small number of IP addresses, but when the root zone is DNSSEC-signed, then the integrity of the responses generated from a root zone is based on the ability of the receiver of the response to validate the signed responses using its local copy of the root keys. Who serves the zone in a DNSSEC context is largely irrelevant.

This approach uses the same root zone signing key to sign a second root zone, where the root zone is served by an exclusively assigned anycast address set. This second set of addresses would not be exclusively assigned to any root server operators, but allowed to be used by any party, in a form of uncontrolled anycast.

In some ways this proposal is similar to the existing AS112 work[10], where anyone can set up a server to respond to common queries for nonexistent top-level domains (such as `.local`) with NXDOMAIN responses. The implication is that if there is a perception that a locale is poorly served by the existing root server anycast constellations, then a local instance of this particular anycast root server can be set up. Because the address is specifically dedicated for unowned anycast, there is no need to coordinate with either the existing root server operators or with other operators of root zone servers on the same anycast address. A prospective operator of one of these root servers simply serves the unowned anycast root zone from one of the small pool (two address couplets, each linking an IPv4 and an IPv6 address) of reserved anycast addresses. Recursive resolvers would query this server by using a distinct unowned anycast root hints file.

Why would any recursive resolver trust the veracity of responses received from one of these unowned anycast root servers? We could well ask the same of recursive resolvers who query into any of the 13 anycast constellations for the root zone, and to some extent the risks of being led astray are similar. The one mitigation of the latter case is that hijacking an existing anycast constellation prefix requires the coercive corruption of the routing system to inject a false instance of an "owned" address, although there is no such concept as hijacking of the unowned anycast prefix. However, in both cases some skepticism on the part of the recursive resolver is to be encouraged, and recursive resolvers should be motivated to validate all such responses using DNSSEC, using the local copy of the DNS trust anchor material. This practice is still a part of "good housekeeping" recommended operational practice for recursive resolvers using the existing root servers, but is a more strongly worded requirement for resolvers using this unowned anycast service.

Although it could be regarded as a byproduct of a single hierarchical name space, the centralization of root zone information in the DNS is operationally problematical and does not cleanly fit within a distributed and decentralized peer model of a network architecture. The adoption of root server anycast constellations is an attempt to respond to this situation, to an extent, by overprovisioning of this critical service. A similar picture is emerging in the area of content provisioning, where cloud-based content is essentially an exercise in overprovisioning, where single points of distribution are replaced by a larger scale of multiple delivery points in order to improve the quality of the delivered service.

However, end users don't enjoy the same level of control, and are dependent on external conditions that are effectively out of their direct control. Not only does this dependence result in highly variable service experiences, but it also leaves the user highly exposed to various forms of online surveillance. The distributing computing world has created external dependencies for users where access to local service is reliant on external availabilities.

The DNS is a good example of this scenario, in so far as resolution of a DNS name that is not already contained in local caches requires priming queries to external servers. Obviously these dependencies on external services highlight fragility where local services cannot be reliably provided using only local infrastructure.

Both of these proposals are incremental in nature, and propose a form of augmentation to the existing structure of recursive resolvers and the root name server, rather than any fundamental change to the existing structure. In so doing, there is a distinct possibility that this form of uncoordinated piecemeal expansion at a local level could prove to be more effective across the Internet, and the critical role of the existing 13 root server operators would diminish over time if it were.

Neither of these proposed approaches is entirely without some form of change. All these uncoordinated root server operators would mean that push notification of root zone changes via the NOTIFY message would not be not feasible, so it would be back to periodic zone transfers and timers in the root zone headers. There is no longer a quick mistake-correction capability in the root zone if served in this way, although it could be argued that the massive level of caching of DNS information actually implied that any changes in the root zone were subject to cache flushing in any case, irrespective of the speed of zone change at the level of the root server anycast constellations.

What is perhaps more worrisome is that the unowned anycast proposal is in effect a proposal to fork the root zone, and recursive resolvers are forced to position themselves within one regime or the other. The only common glue left in this environment is the root key, because the only way that a client of either regime can detect that it is receiving genuine answers is to perform response validation using the root key. This type of validation is placing a massive amount of invested trust in a security artifact that is used today by only a very small subset of recursive resolvers.

It also needs to be noted that our experience to date with unowned anycast has been very poor. At one stage the IPv6 transition experimented with a form of unowned anycast in the form of 6to4 tunnel servers, and the results were hardly reassuring. Anycast clouds follow routing, not geography, and diagnosing operational failures that occur within an uncoordinated anycast structure can range from the merely challenging to highly cryptic and insoluble. The relationship between a recursive resolver and the actual root server it is querying is then occluded, and instances of structural failure in DNS name resolution are far harder to diagnose and correct. Considering that the name translation function is an essential foundation for the Internet, adding operational opacity to the root zone query function is not a step that should be taken lightly.

But that reality does not imply that the other proposal is free from operational concerns, either. The complexity of the local slave resolver, with two concurrent DNS resolvers operating within the same host, should also be questioned. Although there is a current convention in DNS resolver and server deployment to avoid the model of a "mixed" mode resolver that is both an authoritative (or slave) server for some domains and a recursive resolver for all other domains, this avoidance is perhaps nothing more than a convention, and it seems overkill for a resolver to phrase a root query and reach through the loopback interface to ask a co-resident DNS resolver the queries that are being posed to the root.

Why not just operate the local resolver in mixed mode and allow the root query to simply become a memory lookup within a single DNS resolver instance? Perhaps the only justification in this case is the issue of root zone integrity. In the mixed mode of a single resolver instance, the question that arises is a reasonable one: How can the local resolver validate the contents of the transferred root zone? In the loopback model, the local resolver performs DNSSEC validation of the root zone responses and therefore does not necessarily need to separately validate the contents of the transferred root zone. In theory a mixed-mode resolver could DNSSEC-validate the responses retrieved from its local instance of a slave zone server before passing them to the recursive resolver function, but it's not clear that any existing DNS resolver implementations perform this form of DNSSEC validation of internal queries in a mixed mode of operation. Alternate approaches of including a zone signature to ensure integrity are also a possibility to ensure that the recursive resolver is not placed into a position of inadvertently serving corrupt root zone data.

Why not take this thought a further step, and allow any recursive resolver to be a slave server for the root zone? If the zone transfer function included an integrity check across the entire transferred zone (such as a hash of the transferred zone, signed by the root zone signing key), then the recursive resolver could be assured that it was then serving an authentic copy of the root zone.

However, such an integrity check on the transferred zone only assists the local recursive resolver in assuring itself that it has obtained an authentic and current copy of the zone. Clients of that recursive resolver should be as skeptical as ever and ask for DNSSEC signatures, and perform DNSSEC validation over all signed responses that are received from the recursive resolver. The advantage of this approach is that it permits essentially an unlimited number of root name servers, where every recursive server that wants to can serve its own validated copy of the root zone. The disadvantage is that, like all large distributed systems, there is some introduced inertia into the system and updates take time to propagate. However, in a space that is already highly cached, the difference between what happens today and what would happen in such a scenario may well be very hard to see.

There are other ways that a recursive resolver can authoritatively serve responses from the root zone that avoids an explicit root zone transfer, yet still primes the recursive resolver with authoritative information about the contents of the root zone. The response to a query for a nonexistent domain provides NSEC responses that allow a resolver to construct a local cache of the entire root zone (Figure 4).

In the example shown in Figure 4, the response from the root server for the top-level name **nosuchdomain.** indicates in the signed NSEC record that was returned that all names that lie between **no.** and **np.** can be correctly interpreted to be nonexisting domains. As long as the resolver can successfully validate the digital signatures contained in this response, the resolver can cache this negative result and serve NX domain responses for all names in this range for the lifetime of the cache.

Figure 4: Example of an NSEC Response

```
dig +dnssec foo.bar.nosuchdomain @a.root-servers.net

; <<>> DiG 9.9.6 <<>> +dnssec foo.bar.nosuchdomain @a.root-servers.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NXDOMAIN, id: 18580
;; flags: qr aa rd; QUERY: 1, ANSWER: 0, AUTHORITY: 6, ADDITIONAL: 1
;; WARNING: recursion requested but not available

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;foo.bar.nosuchdomain.          IN      A

;; AUTHORITY SECTION:
.                       86400   IN      SOA     a.root-servers.net. nstld.verisign-grs.com.
2015022200 1800 900 604800 86400
.                       86400   IN      RRSIG   SOA 8 0 86400 20150304050000 20150222040000 16665 .
E7no0qtMyyVdVH/0t5LQOM+xV8VJB5GwWp6oaphV+63gi9Dj8LG71kb8 N00Sx0TaJAISa18NLa27/RPzoz3vvQAnIpyZxmhxzfyk
fkLhXxaJtFCV 4hKWxqf0EymCzGCsBIRSMttl7fypf3am15JF3ei0Cqmp/BHWjXjGs0mO te8=
.                       86400   IN      NSEC    abogado. NS SOA RRSIG NSEC DNSKEY
.                       86400   IN      RRSIG   NSEC 8 0 86400 20150304050000 20150222040000 16665 .
ojA/fqJKig89aw9+KtM2RswgMaxTVrogPiGeqoLUZgD9Rf3UNOn2tLtO VpDzzB45BquRpdfV+OluDWo+L81nRqjM5CAiZkaektUW
MmOcvqCnFf9w RuiqMdAq4vVExSWZU4G/aF6Y6WzoHVC5GlWS31PHfm9Ux2UeyeEMQ1o/ aac=
no.                     86400   IN      NSEC    np. NS DS RRSIG NSEC
no.                     86400   IN      RRSIG   NSEC 8 1 86400 20150304050000 20150222040000 16665 .
K7dypmjhxfDGtP5oWSvToH53qYdfcGi0MQ7xkF+/k89HFbZnFtOrhGd/ 8zJAdUtednJxvk55LjHY+uU4uiaNuNc+7jAilFEKL8BF
sgzvy98lvgkk 63YqAoRHJJ357q+hVil0UxVKcb8oCe3VWZsOtamap6ujSzZZQy4X3TKt jZ0=

;; Query time: 146 msec
;; SERVER: 198.41.0.4#53(198.41.0.4)
;; WHEN: Sun Feb 22 08:28:24 UTC 2015
;; MSG SIZE  rcvd: 654
```

Conclusions

DNSSEC is indeed an extremely valuable asset for the DNS, and we can move forward with a larger and more robust system if we can count on various efforts to subvert the operation DNS being thwarted by all forms of clients of the DNS, even to the level of applications insisting that they are exposed to the DNS responses and their signatures, and performing their own validation on the received data.

There is a more general observation about scaling in the Internet. We are finding it increasingly challenging to react to inexorable pressures of scaling the infrastructure of the Internet while still maintaining basic backward compatibility with systems that conform only to technical standards of the 1980s. We need to move on. We have moved beyond the 512-octet packet limit, and these days it's reasonable to assume that useful resolvers can support EDNS0 options in DNS queries. Resolvers should perform DNSSEC validation. Wondering why there are 13 available slots for root name server operators, and wondering about how we could alter their composition, or change the interaction with the DNS root to support one or two additional root servers, are unproductive lines of investigation at so many levels. It may well be time to contemplate a different DNS that does not involve these arbitrary constraints over the number and composition of players that serve the signed root zone. Instead we should rely on using DNSSEC validation to ensure that the responses received from queries to the root zone are authentic.

The attraction of unconstrained systems is that local actors can respond to local needs, and respond by using local resources, without having to coordinate or cross-subsidize the activities of others. Much of the momentum of the Internet is directed by this loosely constrained model of interaction. If we can use the possibilities opened up by securing the DNS payload, where the question of who passed the DNS information to you is irrelevant but the question of whether the information is locally verifiable is critically important, then and only then can we contemplate what it would take to operate an unconstrained DNS system for serving the root zone.

References

[1] R. Braden, "Requirements for Internet Hosts – Communication Layers," RFC 1122, October 1989.

[2] P. Mockapetris, "Domain names – Implementation and Specification," RFC 1035, November 1987.

[3] The DIG Command: **http://en.wikipedia.org/wiki/Dig_(command)**

[4] R. Braden, "Requirements for Internet Hosts – Application and Support," RFC 1123, October 1989.

[5] R. Arends, et.al, "Protocol Modifications for the DNS Security Extensions," RFC 4035, March 2005.

[6] P. Vixie, J. Damas, and M. Graff, "Extension Mechanisms for DNS (EDNS0)," RFC 6891, April 2013.

[7] K. Lindqvist and J. Abley, "Operation of Anycast Services," RFC 4786, December 2006.

[8] W. Kummari and P. Hoffman, "Decreasing Access Time to Root Servers by Running One on Loopback," work in progress, (Internet Draft: **draft-wkumari-dnsop-root-loopback-02.txt**), November 2014.

[9] X. Lee and P. Vixie, "How to Scale the DNS Root System?" work in progress (Internet Draft: **draft-lee-dnsop-scalingroot-00.txt**), July 2014.

[10] J. Abley and W. Maton, "AS112 Nameserver Operations," RFC 6304, July 2011.

[11] W. Stallings, "Gigabit Ethernet: From 1 to 100 Gbps and Beyond," *The Internet Protocol Journal,* Volume 18, No. 1, March 2015.

[12] G. Huston, "A Question of DNS Protocols," *The Internet Protocol Journal,* Volume 17, No. 1, September 2014.

[13] E. Feinler, "Host Tables, Top-Level Domain Names, and the Origin of Dot Com," *IEEE Annals of the History of Computing,* July-September 2011, **http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5986499**

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**

Gigabit Ethernet: From 1 to 100 Gbps and Beyond

by William Stallings

Since its first introduction in the early 1980s, Ethernet has been the dominant technology for implementing *Local-Area Networks* (LANs) in office environments. Over the years, the data-rate demands placed on LANs have grown at a rapid pace. Fortunately, Ethernet technology has adapted to provide ever-higher capacity to meet these needs. We are now in the era of the Gigabit Ethernet.

Ethernet began as an experimental bus-based 2.94-Mbps system[1] using coaxial cable. In a shared bus system, all stations attach to a common cable, with only one station able to successfully transmit at a time. A *Medium Access Control* (MAC) protocol based on collision detection arbitrates the use of the bus. In essence, each station is free to transmit MAC frames on the bus. If a station detects a collision during transmission, it backs off a certain amount of time and tries again.

The first commercially available Ethernet products were bus-based systems operating at 10 Mbps[2]. This introduction coincided with the standardization of Ethernet by the IEEE 802.3 committee. With no change to the MAC protocol or MAC frame format, Ethernet could also be configured in a star topology, with traffic going through a central hub, again with transmission limited to a single station at a time through the hub. To enable an increase in the data rate, a switch replaces the hub, allowing full-duplex operation. With the switch, the same MAC format and protocol are used, although collision detection is no longer needed. As the demand has evolved and the data rate requirement increased, some enhancements to the MAC layer have been added, such as provision for larger frame sizes.

Currently, Ethernet systems are available at speeds up to 100 Gbps. Table 1 summarizes the successive generations of IEEE 802.3 standardization.

Table 1: IEEE 802.3 Physical Layer Standards

Year Introduced	1983	1995	1998	2003	2010
Maximum data transfer speed	10 Mbps	100 Mbps	1 Gbps	10 Gbps	40 Gbps, 100 Gbps
Transmission media	Coax cable, unshielded twisted pair, optical fiber	Unshielded twisted pair, shielded twisted pair, optical fiber	Unshielded twisted pair, optical fiber, shielded twisted pair	Optical fiber	Optical fiber, backplane

Ethernet quickly achieved widespread acceptance and soon became the dominant technology for LANs. Its dominance has since spread to *Metropolitan-Area Networks* (MANs) and a wide range of applications and environments.

The huge success of Ethernet is due to its extraordinary adaptability. The same MAC protocol and frame format are used at all data rates. The differences are at the physical layer, in the definition of signaling technique and transmission medium.

In the remainder of this article, we look at characteristics of Ethernet in the gigabit range.

1-Gbps Ethernet

For many years the initial standard of Ethernet, at 10 Mbps, was adequate for most office environments. By the early 1990s, it was clear that higher data rates were needed to support the growing traffic load on the typical LAN. Key drivers in this evolution include:

- *Centralized Server Farms:* In many multimedia applications, there is a need for the client system to be able to draw huge amounts of data from multiple, centralized servers, called *Server Farms*. As the performance of the servers has increased, the network has become the bottleneck.
- *Power Workgroups:* These groups typically consist of a small number of cooperating users who need to exchange massive data files across the network. Example applications are software development and computer-aided design.
- *High-speed Local Backbone:* As processing demand grows, enterprises develop a configuration of multiple LANs interconnected with a high-speed backbone network.

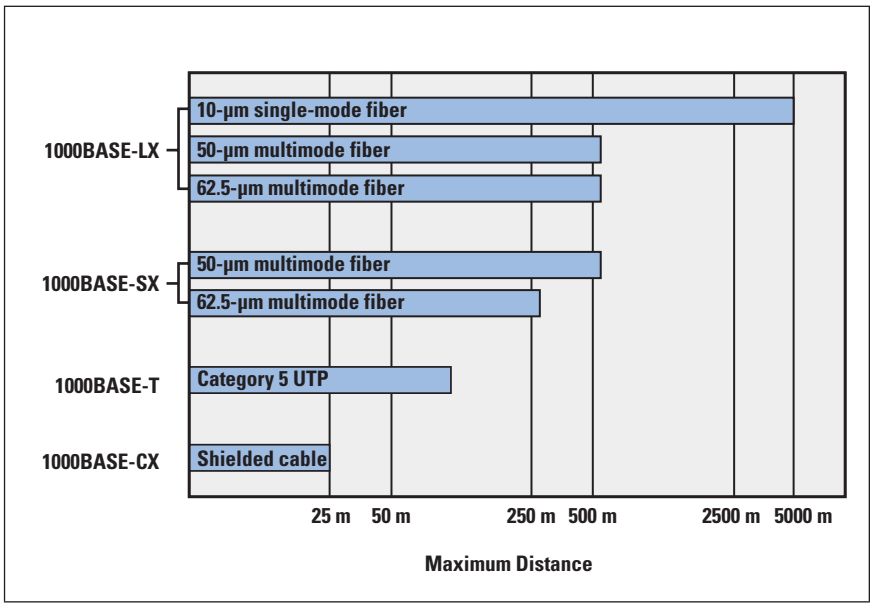
To meet such needs, the IEEE 802.3 committee developed a set of specifications for Ethernet at 100 Mbps, followed a few years later by a 1-Gbps family of standards. In each case, the new specifications defined transmission media and transmission encoding schemes built on the basic Ethernet framework, making the transition easier than if a completely new specification were issued.

The 1-Gbps standard includes a variety of transmission medium options[3, 4] (Figure 1):

- *+1000BASE-SX:* This short-wavelength option supports duplex links of up to 275 m using 62.5-µm multimode or up to 550 m using 50-µm multimode fiber. Wavelengths are in the range of 770 to 860 nm.
- *1000BASE-LX:* This long-wavelength option supports duplex links of up to 550 m of 62.5-µm or 50-µm multimode fiber or 5 km of 10-µm single-mode fiber. Wavelengths are in the range of 1270 to 1355 nm.

- *1000BASE-CX:* This option supports 1-Gbps links among devices located within a single room or equipment rack, using copper jumpers (specialized shielded twisted-pair cable that spans no more than 25 m). Each link is composed of a separate shielded twisted pair running in each direction.
- *1000BASE-T:* This option uses four pairs of Category 5 unshielded twisted pair to support devices over a range of up to 100 m, transmitting and receiving on all four pairs at the same time, with echo cancellation circuitry.

Figure 1: 1-Gbps Ethernet Medium Options (log scale)



The signal encoding scheme used for the first three Gigabit Ethernet options just listed is 8B/10B. With 8B/10B, each 8 bits of data is converted into 10 bits for transmission[3]. The extra bits serve two purposes. First, the resulting signal stream has more transitions between logical 1 and 0 than an uncoded stream; it avoids the possibility of long strings of 1s or 0s that make synchronization between transmitter and receiver more difficult. Second, the code is designed in such a way as to provide a useful error-detection capability.

The signal-encoding scheme used for 1000BASE-T is 4D-PAM5, a complex scheme whose description is beyond our scope.

In a typical application of Gigabit Ethernet, a 1-Gbps LAN switch provides backbone connectivity for central servers and high-speed workgroup Ethernet switches. Each workgroup LAN switch supports both 1-Gbps links, to connect to the backbone LAN switch and to support high-performance workgroup servers, and 100-Mbps links, to support high-performance workstations, servers, and 100-Mbps LAN switches.

10-Gbps Ethernet

Even as the ink was drying on the 1-Gbps specification, the continuing increase in local traffic made this specification inadequate for needs in the short-term future. Accordingly, the IEEE 802.3 committee soon issued a standard for 10-Gbps Ethernet. The principal requirement for 10-Gbps Ethernet was the increase in intranet (local interconnected networks) and Internet traffic. Numerous factors contribute to the explosive growth in both Internet and intranet traffic:

- An increase in the number of network connections
- An increase in the connection speed of each end station (for example, 10-Mbps users moving to 100 Mbps, and analog 56k users moving to DSL and cable modems)
- An increase in the deployment of bandwidth-intensive applications such as high-quality video
- An increase in Web hosting and application hosting traffic

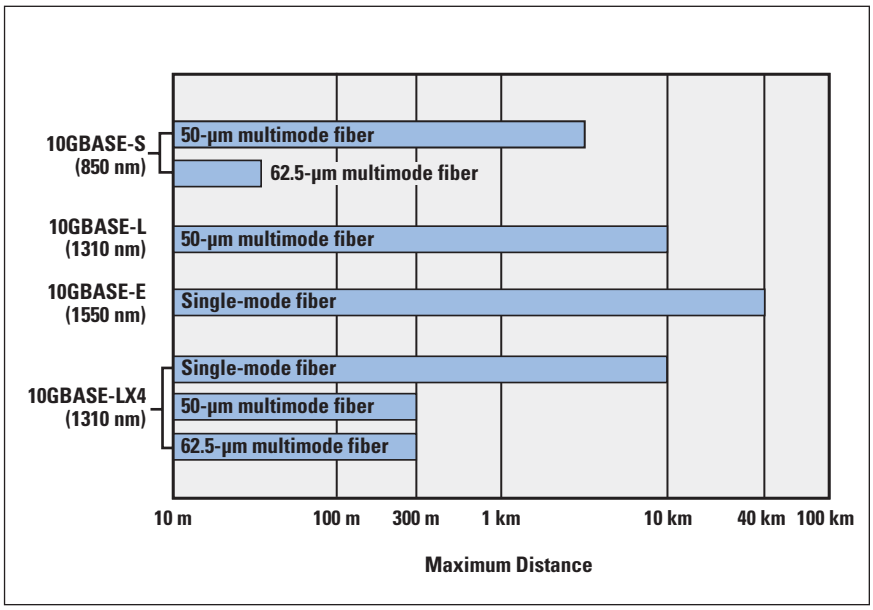
Initially, network managers used 10-Gbps Ethernet to provide high-speed, local backbone interconnection between large-capacity switches. As the demand for bandwidth increased, 10-Gbps Ethernet began to be deployed throughout the entire network, to include server farm, backbone, and campus-wide connectivity. This technology enables *Internet Service Providers* (ISPs) and *Network Service Providers* (NSPs) to create very-high-speed links at a very low cost, between co-located, carrier-class switches, and routers[5].

The technology also allows the construction of MANs and *Wide-Area Networks* (WANs) that connect geographically dispersed LANs between campuses or *Points of Presence* (PoPs). Thus, Ethernet begins to compete with *Asynchronous Transfer Mode* (ATM) and other wide-area transmission and networking technologies. In most cases where the customer requirement is data and TCP/IP transport, 10-Gbps Ethernet provides substantial value over ATM transport for both network end users and service providers:

- No expensive, bandwidth-consuming conversion between Ethernet packets and ATM cells is required; the network is Ethernet, end to end.
- The combination of IP and Ethernet offers *Quality of Service* (QoS) and traffic policing capabilities that approach those provided by ATM, so that advanced traffic-engineering technologies are available to users and providers.
- A wide variety of standard optical interfaces (wavelengths and link distances) have been specified for 10 Gigabit Ethernet, optimizing its operation and cost for LAN, MAN, or WAN applications.

The goal for maximum link distances cover a range of applications is from 300 m to 40 km. The links operate in full-duplex mode only, using a variety of optical fiber physical media.

Figure 2: 10-Gbps Ethernet Distance Options (log scale)



Four physical layer options are defined for 10-Gbps Ethernet (Figure 2). The first three have two suboptions: an "R" suboption and a "W" suboption. The R designation refers to a family of physical layer implementations that use a signal encoding technique known as 64B/66B. With 64B/66B, each 64 bits of data is converted into 66 bits for transmission, resulting in substantially less overhead then 8B/10B but providing the same types of benefits. The R implementations are designed for use over *dark fiber* meaning a fiber-optic cable that is not in use and that is not connected to any other equipment. The W designation refers to a family of physical layer implementations that also use 64B/66B signaling but are then encapsulated to connect to SONET equipment.

The four physical layer options are as follows:

- *10GBASE-S (short):* Designed for 850-nm transmission on multimode fiber. This medium can achieve distances up to 300 m. There are 10GBASE-SR and 10GBASE-SW versions.
- *10GBASE-L (long):* Designed for 1310-nm transmission on single-mode fiber. This medium can achieve distances up to 10 km. There are 10GBASE-LR and 10GBASE-LW versions.
- *10GBASE-E (extended):* Designed for 1550-nm transmission on single-mode fiber. This medium can achieve distances up to 40 km. There are 10GBASE-ER and 10GBASE-EW versions.
- *10GBASE-LX4:* Designed for 1310-nm transmission on single-mode or multimode fiber. This medium can achieve distances up to 10 km. This medium uses *Wavelength-Division Multiplexing* (WDM) to multiplex the bit stream across four light waves.

40-/100-Gbps Ethernet

Ethernet is widely deployed and is the preferred technology for wired local-area networking. Ethernet dominates enterprise LANs, broadband access, and data center networking, and it has also become popular for communication across MANs and even WANs. Further, it is now the preferred carrier wire-line vehicle for bridging wireless technologies, such as Wi-Fi and WiMAX, into local Ethernet networks.

This popularity of Ethernet technology is due to the availability of cost-effective, reliable, and interoperable networking products from a variety of vendors. The development of converged and unified communications, the evolution of massive server farms, and the continuing expansion of *Voice over IP* (VoIP), *Television over IP* (TVoIP), and Web 2.0 applications have accelerated the need for ever-faster Ethernet switches. The following are market drivers for 100-Gbps Ethernet:

- *Data center/Internet Media Providers:* To support the growth of Internet multimedia content and Web applications, content providers have been expanding data centers, pushing 10-Gbps Ethernet to its limits. These providers are likely to be high-volume early adopters of 100-Gbps Ethernet.
- *Metro-Video/Service Providers:* Video on demand has been leading a new generation of 10-Gbps Ethernet metropolitan/core network build-outs. These providers are likely to be high-volume adopters in the medium term.
- *Enterprise LANs:* Continuing growth in convergence of voice/video/data and in unified communications is accelerating network switch demands. However, most enterprises still rely on 1-Gbps or a mix of 1-Gbps and 10-Gbps Ethernet, and adoption of 100-Gbps Ethernet is likely to be slow.
- *Internet exchanges/ISP Core Routing:* With the massive amount of traffic flowing through these nodes, these installations are likely to be early adopters of 100-Gbps Ethernet.

In 2007, the IEEE 802.3 working group authorized the *IEEE P802.3ba 40-Gbps and 100-Gbps Ethernet Task Force*. The 802.3ba project authorization request cited numerous examples of applications that require greater data-rate capacity than 10-Gbps Ethernet offers, including Internet exchanges, high-performance computing, and video-on-demand delivery. The authorization request justified the need for two different data rates in the new standard (40 Gbps and 100 Gbps) by recognizing that aggregate network requirements and end-station requirements are increasing at different rates.

An example of the application of 100-Gbps Ethernet is shown in Figure 3. The trend at large data centers, with substantial banks of blade servers, is the deployment of 10-Gbps ports on individual servers to handle the massive multimedia traffic provided by these servers. Typically, a single blade-server rack will contain multiple servers and one or two 10-Gbps Ethernet switches to interconnect all the servers and provide connectivity to the rest of the facility. The switches are often mounted in the rack and referred to as *Top-of-Rack* (ToR) switches. The term ToR has become synonymous with a server access switch, even if it is not located "top of rack." For very large data centers, such as cloud providers, the interconnection of multiple blade-server racks with additional 10-Gbps switches is increasingly inadequate. To handle the increased traffic load, switches operating at greater than 10 Gbps are needed to support the interconnection of server racks and to provide adequate capacity for connecting off-site through *Network Interface Controllers* (NICs).

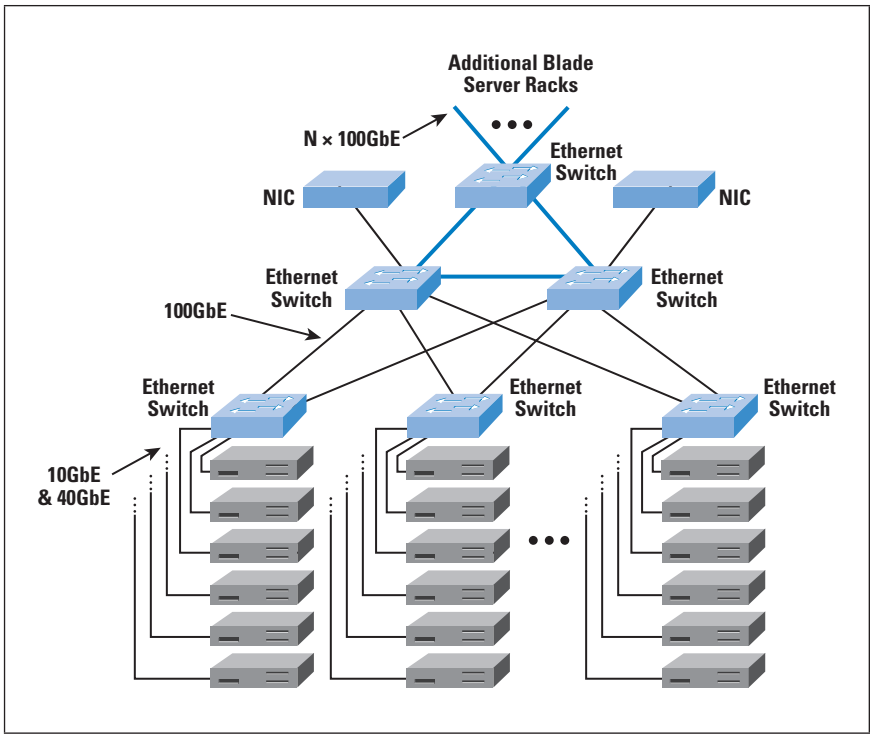


Figure 3: Example 100-Gbps Ethernet Configuration for Massive Blade-Server Cloud Site

The first products in this category appeared in 2009, and the IEEE 802.3ba standard was finalized in 2010. Initially, many enterprises are deploying 40-Gbps switches, but both 40- and 100-Gbps switches are projected to enjoy increased market penetration in the next few years[6, 7, 8].

IEEE 802.3ba specifies three types of transmission media as shown in Table 2: copper backplane, twin axial (a type of cable similar to coaxial cable), and optical fiber. For copper media, four separate physical lanes are specified. For optical fiber, either 4 or 10 wavelength lanes are specified, depending on data rate and distance[9, 10, 11].

Table 2: Media Options for 40- and 100-Gbps Ethernet

	40 Gbps	100 Gbps
1-m backplane	40GBASE-KR4	
10-m copper	40GBASE-CR4	100GBASE-CR10
100-m multimode fiber	40GBASE-SR4	100GBASE-SR10
10-km single-mode fiber	40GBASE-LR4	100GBASE-LR4
40-km single-mode fiber		100GBASE-ER4

Naming nomenclature:
Copper: K = Backplane; C = Cable assembly
Optical: S = Short reach (100 m); L - Long reach (10 km);
E = Extended long reach (40 km)
Coding scheme: R = 64B/66B block coding
Final number: Number of lanes (copper wires or fiber wavelengths)

Multilane Distribution

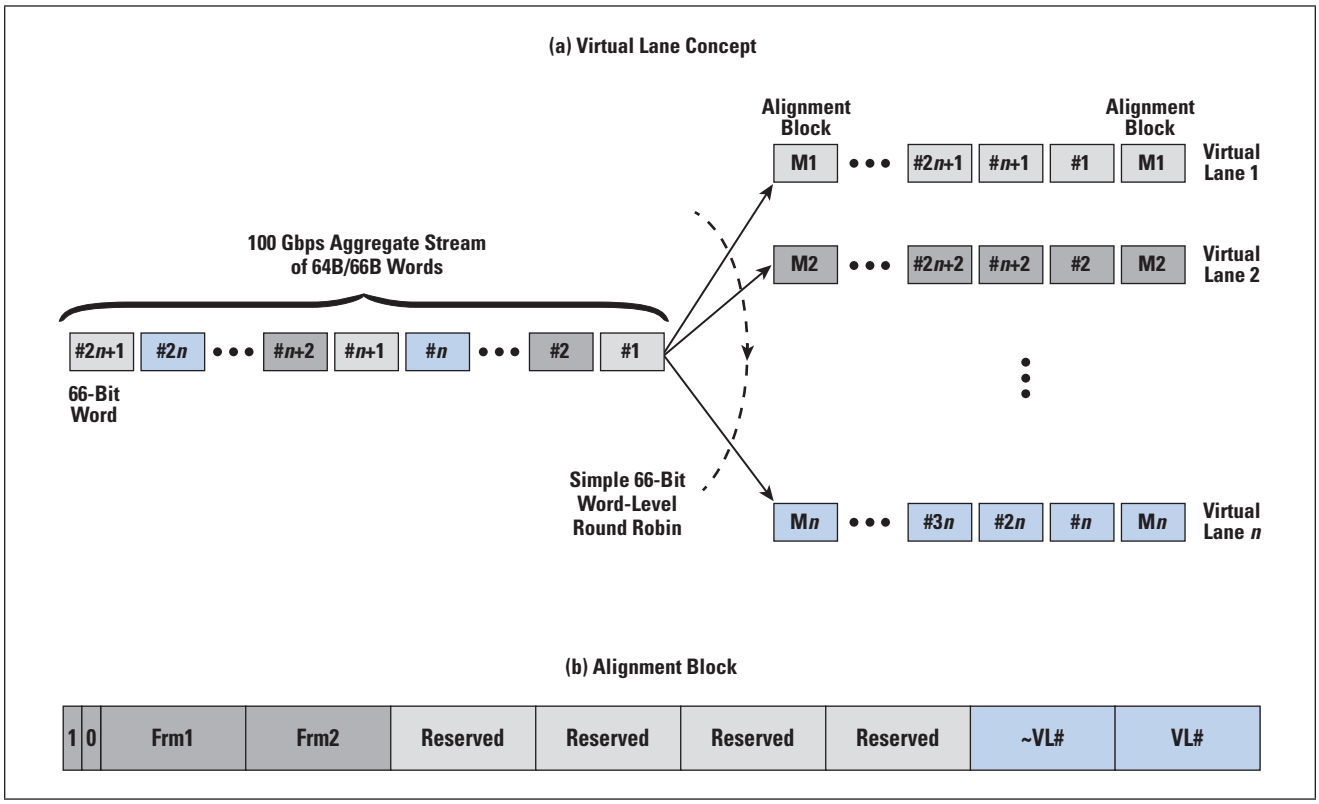
The 802.3ba standard uses a technique known as multilane distribution to achieve the required data rates. Two separate concepts need to be addressed: multilane distribution and virtual lanes.

The general idea of *multilane distribution* is that, in order to accommodate the very high data rates of 40 and 100 Gbps, the physical link between an end station and an Ethernet switch or the physical link between two switches may be implemented as multiple parallel channels. These parallel channels could be separate physical wires, such as four parallel twisted-pair links between nodes. Alternatively, the parallel channels could be separate frequency channels, such as provided by WDM over a single optical fiber link.

For simplicity and manufacturing ease, we would like to specify a specific multiple-lane structure in the electrical physical sublayer of the device, known as the *Physical Medium Attachment* (PMA) sublayer. The lanes produced are referred to as *virtual lanes*. If a different number of lanes are actually in use in the electrical or optical link, then the virtual lanes are distributed into the appropriate number of physical lanes in the *Physical Medium Dependent* (PMD) sublayer. This is a form of inverse multiplexing.

Figure 4a shows the virtual lane scheme at the transmitter. The user data stream is encoded using the 64B/66B, which is also used in 10-Gbps Ethernet. Data is distributed to the virtual lanes one 66-bit word at a time using a simple round robin scheme (first word to first lane, second word to second lane, etc.). A unique 66-bit alignment block is added to each virtual lane periodically. The alignment blocks are used to identify and reorder the virtual lanes and thus reconstruct the aggregate data stream.

Figure 4: Multilane Distribution for 100-Gbps Ethernet



The virtual lanes are then transmitted over physical lanes. If the number of physical lanes is smaller then the number of virtual lanes, then bit-level multiplexing is used to transmit the virtual lane traffic. The number of virtual lanes must be an integer multiple (1 or more) of the number of physical lanes.

Figure 4b shows the format of the alignment block. The block consists of 8 single-byte fields preceded by the 2-bit synchronization field, which has the value 10. The Frm fields contain a fixed framing pattern common to all virtual lanes and used by the receiver to locate the alignment blocks. The VL# fields contain a pattern unique to the virtual lane: one of the fields is the binary inverse of the other.

25-/50-Gbps Ethernet

One of the options for implementing 100 Gbps is as four 25-Gbps physical lanes. Thus, it would be relatively easy to develop standards for 25- and 50-Gbps Ethernet, using one or two lanes, respectively. Having these two lower-speed alternatives, based on the 100-Gbps technology, would give users more flexibility in meeting existing and near-term demands with a solution that would scale easily to higher data rates.

Such considerations have led to the formation of the *25 Gigabit Ethernet Consortium* by numerous leading cloud networking providers, including Google and Microsoft. The objective of the consortium is to support an industry-standard, interoperable Ethernet specification that boosts the performance and slashes the interconnect cost per Gbps between the NIC and ToR switch. The specification adopted by the consortium prescribes a single-lane 25-Gbps Ethernet and dual-lane 50-Gbps Ethernet link protocol, enabling up to 2.5 times higher performance per physical lane on twinax copper wire between the rack endpoint and switch compared to current 10- and 40-Gbps Ethernet links. The IEEE 802.3 committee is presently developing the needed standards for 25 Gbps, and it may include 50 Gbps[12, 13].

It is too early to say how these various options (25, 40, 50, and 100 Gbps) will play out in the marketplace. In the intermediate term, the 100-Gbps switch is likely to predominate at large sites, but the availability of these slower and cheaper alternatives gives enterprises numerous paths for scaling up to meet increasing demand.

400-Gbps Ethernet

The growth in demand never lets up. IEEE 802.3 is currently exploring technology options for producing a 400-Gbps Ethernet standard, although no timetable is yet in place[14, 15, 16, 17]. Looking beyond that milestone, there is widespread acknowledgment that a 1-Tbps (terabit per second, trillion bits per second) standard will eventually be produced[18].

2.5-/5-Gbps Ethernet

As a testament to the versatility and ubiquity of Ethernet, and at the same time the fact that ever-higher data rates are being standardized, consensus is developing to standardize two lower rates: 2.5 and 5 Gbps[19, 20]. These relatively low speeds are also known as *Multirate Gigabit BASE-T* (MGBASE-T). Currently, the *MGBASE-T Alliance* is overseeing the development of these standards outside of IEEE. It is likely that the IEEE 802.3 committee will ultimately issue standards based on these industry efforts.

These new data rates are intended mainly to support IEEE 802.11ac wireless traffic into a wired network. IEEE 802.11ac is a 3.2-Gbps Wi-Fi standard that is gaining acceptance where more than 1 Gbps of throughput is needed, such as to support mobile users in the office environment[21]. This new wireless standard overruns 1-Gbps Ethernet link support but may not require the next step up, which is 10 Gbps. Assuming that 2.5 and 5 Gbps can be made to work over the same cable that supports 1 Gbps, then this standard would provide a much-needed uplink speed improvement for access points supporting 802.11ac radios with their high-bandwidth capabilities.

Conclusion

Ethernet is widely deployed and is the preferred technology for wired local-area networking. Ethernet dominates enterprise LANs, broadband access, and data center networking, and it has also become popular for communication across MANs and even WANs. Further, it is now the preferred carrier wire-line vehicle for bridging wireless technologies, such as Wi-Fi and WiMAX, into local Ethernet networks. Further, the Ethernet marketplace is now large enough to accelerate the development of speeds for specific use cases, such as 25/50 Gbps for data center ToR designs and 2.5/5 Gbps for wireless infrastructure backhaul. The availability of a wide variety of standardized Ethernet data rates allows the network manager to customize a solution to optimize performance, cost, and energy consumption goals[22].

This popularity of Ethernet technology is due to the availability of cost-effective, reliable, and interoperable networking products from a variety of vendors. The development of converged and unified communications, the evolution of massive server farms, and the continuing expansion of VoIP, TVoIP, and Web 2.0 applications have accelerated the need for ever-faster Ethernet switches.

The success of Gigabit Ethernet and 10-Gbps Ethernet highlights the importance of network-management concerns in choosing a network technology. The 40- and 100-Gbps Ethernet specifications offer compatibility with existing installed LANs, network-management software, and applications. This compatibility has accounted for the survival of 30-year-old technology in today's fast-evolving network environment.

References

[1] Metcalfe, R., and Boggs, D., "Ethernet: Distributed Packet Switching for Local Computer Networks," *Communications of the ACM,* July 1976.

[2] Shoch, J., Dalal, Y., Redell, D., and Crane, R., "Evolution of the Ethernet Local Computer Network," *Computer,* August 1982.

[3] Stallings, W., "Gigabit Ethernet," *The Internet Protocol Journal,* Volume 2, No. 3, September 1999.

[4] Frazier, H., and Johnson, H. "Gigabit Ethernet: From 100 to 1,000 Mbps," *IEEE Internet Computing,* January/February 1999.

[5] GadelRab, S., "10-Gigabit Ethernet Connectivity for Computer Servers," *IEEE Micro,* May-June 2007.

[6] Mcgillicuddy, S., "40 Gigabit Ethernet: The Migration Begins," *Network Evolution E-Zine,* December 2012.

[7] Chanda, G., and Yang, Y., "40 GbE: What, Why & Its Market Potential," Ethernet Alliance White Paper, November 2010.

[8] Nowell, M., Vusirikala, V., and Hays, R., "Overview of Requirements and Applications for 40 Gigabit and 100 Gigabit Ethernet," Ethernet Alliance White Paper, August 2007.

[9] D'Ambrosia, J., Law, D., and Nowell, M., "40 Gigabit Ethernet and 100 Gigabit Ethernet Technology Overview," Ethernet Alliance White Paper, November 2008.

[10] Toyoda, H., Ono, G., and Nishimura, S., "100 GbE PHY and MAC Layer Implementation," *IEEE Communications Magazine,* March 2010.

[11] Rabinovich, R., and Lucent, A., "40 Gb/s and 100 Gb/s Ethernet Short Reach Optical and Copper Host Board Channel Design," *IEEE Communications Magazine,* April 2012.

[12] Morgan, T., "IEEE Gets Behind 25G Ethernet Effort," *Enterprise Tech,* July 27, 2014.

[13] Merritt, R., "50G Ethernet Debate Brewing," *EE Times,* September 3, 2014.

[14] Nolle, T., "Will We Ever Need 400 Gigabit Ethernet Enterprise Networks?" *Network Evolution E-Zine,* December 2012.

[15] D'Ambrosia, J., Mooney, P., and Nowell, M., "400 Gb/s Ethernet: Why Now?" Ethernet Alliance White Paper, April 2013.

[16] Hardy, S., "400 Gigabit Ethernet Task Force Ready to Get to Work," *Lightwave,* March 28, 2014.

[17] D'Ambrosia, J., "400GbE and High Performance Computing," *Scientific Computing Blog,* April 18, 2014.

[18] Duffy, J., "Bridge to Terabit Ethernet," *Network World,* April 20, 2009.

[19] D'Ambrosia, J., "TEF 2014: The Rate Debate," *Ethernet Alliance Blog,* June 23, 2014.

[20] Kipp, S., "5 New Speeds – 2.5, 5, 25, 50 and 400 GbE," *Ethernet Alliance Blog,* August 8, 2014.

[21] Stallings, W., "Gigabit Wi-Fi," *The Internet Protocol Journal,* Volume 17, No. 1, September 2014.

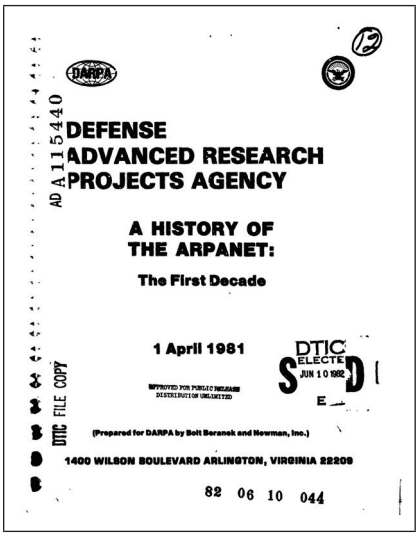
[22] Chalupsky, D., and Healey, A., "Datacenter Ethernet: Know Your Options," *Network Computing,* March 28, 2014.

WILLIAM STALLINGS is an independent consultant and author of numerous books on security, computer networking, and computer architecture. His latest book is *Data and Computer Communications* (Pearson, 2014). He maintains a computer science resource site for computer science students and professionals at **ComputerScienceStudent.com**. He has a Ph.D. in computer science from M.I.T. He can be reached at **ws@shore.net**

Fragments

ARPANET History

Bolt Beranek and Newman Report 4799 entitled "A History of the ARPANET: The First Decade," is a fascinating document for anyone interested in early Internet History. First published in 1981, a scanned PDF version can be downloaded from the following link:
www.darpa.mil/WorkArea/DownloadAsset.aspx?id=2677


AD A115440

DEFENSE ADVANCED RESEARCH PROJECTS AGENCY

A HISTORY OF THE ARPANET:

The First Decade

1 April 1981

DTIC ELECTE JUN 10 1982 S D E

DTIC FILE COPY

(Prepared for DARPA by Bolt Beranek and Newman, Inc.)

1400 WILSON BOULEVARD ARLINGTON, VIRGINIA 22209

82 06 10 044

IANA Transition

Since the United States *National Telecommunications and Information Administration* (NTIA) announced its intent to "...transition Key Internet Domain Name Functions to the global multistakeholder community" last March, a flurry of activity has been taking place within the Internet technical and policy communities. The following website provides further information and links to the relevant working groups and documents: **https://www.icann.org/stewardship**

Upcoming Events

The *North American Network Operators' Group* (NANOG) will meet in San Francisco, California, June 1–3, 2015 and in Montreal, Quebec, Canada, October 5–7, 2015. See: **http://nanog.org**

The *Internet Corporation for Assigned Names and Numbers* (ICANN) will meet 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 meet in Auckland, New Zealand, February 16–26, 2016. See: **http://www.apricot.net**

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

Call for Papers

The *Internet Protocol Journal* (IPJ) is a quarterly technical publication containing tutorial articles ("What is...?") as well as implementation/operation articles ("How to..."). The journal provides articles about all aspects of Internet technology. IPJ is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. In addition to feature-length articles, IPJ contains technical updates, book reviews, announcements, opinion columns, and letters to the Editor. Topics include but are not limited to:

- Access and infrastructure technologies such as: Wi-Fi, Gigabit Ethernet, SONET, xDSL, cable, fiber optics, satellite, and mobile wireless.
- Transport and interconnection functions such as: switching, routing, tunneling, protocol transition, multicast, and performance.
- Network management, administration, and security issues, including: authentication, privacy, encryption, monitoring, firewalls, troubleshooting, and mapping.
- Value-added systems and services such as: Virtual Private Networks, resource location, caching, client/server systems, distributed systems, cloud computing, and quality of service.
- Application and end-user issues such as: E-mail, Web authoring, server technologies and systems, electronic commerce, and application management.
- Legal, policy, regulatory and governance topics such as: copyright, content control, content liability, settlement charges, resource allocation, and trademark disputes in the context of internetworking.

IPJ will pay a stipend of US$1000 for published, feature-length articles. For further information regarding article submissions, please contact Ole J. Jacobsen, Editor and Publisher. Ole can be reached at **ole@protocoljournal.org** or **olejacobsen@me.com**

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

Diamond Sponsors

Ruby Sponsor

Sapphire Sponsors

Emerald Sponsors

Corporate Subscriptions

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, Andrea Montefusco, 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 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

Printed in the USA on recycled paper.