The Internet Protocol Journal

September 2014 Volume 17, Number 1

*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

It is with great pleasure and gratitude that I announce the re-launch of *The Internet Protocol Journal* (IPJ) after a hiatus of just over one year. To subscribe, send e-mail to **ipj@protocoljournal.org** and we will send you further information. Please bear with us as we deploy a new subscription system and web page.

The re-launch of IPJ is made possible by the support of The Internet Society, which provides administrative functions, and Cisco Systems, which allows us to use the subscriber list and journal name under license. Sponsorship for IPJ is provided by: Afilias; Asia Pacific Internet Association (APIA); Asia Pacific Network Information Centre (APNIC); Cisco Systems; Comcast; Dyn; Google, Inc.; The Internet Corporation for Assigned Names and Numbers (ICANN); Juniper Networks; Limelight Networks; Netnod; Network Startup Resource Center (NSRC); Réseaux IP Européens Network Coordination Centre (RIPE NCC); Stichting Internet Domeinregistratie Nederland (SIDN); Team Cymru; Verisign Labs; Widely Integrated Distributed Environment (WIDE); 21Vianet Group; Dave Crocker; Jay Etchings; Dennis Jennings; Jim Johnston; Bill Manning; George Sadowsky; Helge Skrivervik; Rob Thomas; and Tom Vest. If you are interested in sponsoring IPJ, please contact us at **ipj@protocoljournal.org**

We have augmented our Editorial Advisory Board and welcome Fred Baker, Cisco Fellow, Cisco Systems; Steve Crocker, Chairman, ICANN; Geoff Huston, Chief Scientist, APNIC; and Olaf Kolkman, Chief Internet Technology Officer, The Internet Society. Our thanks go to our outgoing members David Farber, Deepinder Sidhu, and Peter Löthberg. The complete Editorial Advisory Board is listed on the back cover.

The increasing demand for more bandwidth in all aspects of networking has led to technology changes in wide-area, local-area, and mobile networks. Newer, faster technologies have been developed and deployed. In our first article, William Stallings gives an overview of *Gigabit Wi-Fi,* also known as IEEE 802.11ac. This relatively new version of Wi-Fi promises transmission speeds of up to 3.2 Gbps.

In our second article, Geoff Huston describes several Denial-of-Service attacks that were launched using the *Domain Name System* (DNS) as the attack vector and discusses details of DNS protocol interactions.

This journal will continue to cover all aspects of internetworking just as it always has. We welcome your feedback and suggestions!

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

# Gigabit Wi-Fi

*by William Stallings*

Just as businesses and home users have generated a need to extend the Ethernet standard to speeds in the gigabit-per-second (Gbps) range, the same requirement exists for the wireless network technology known as *Wi-Fi*. Accordingly, IEEE 802.11, the committee responsible for wireless LAN standards, has recently introduced two new standards[1], 802.11ac[2] and 802.11ad[3, 4], which provide for Wi-Fi networks that operate at well in excess of 1 Gbps. These two new standards build on previous work by the IEEE 802.11 committee, which has introduced numerous versions of the wireless LAN standard over the years (Table 1).

*Table 1: IEEE 802.11 Physical Layer Standards*

| Standard | 802.11a | 802.11b | 802.11g | 802.11n | 802.11ac | 802.11ad |
|---|---|---|---|---|---|---|
| Year introduced | 1999 | 1999 | 2003 | 2000 | 2012 | 2014 |
| Maximum data-transfer speed | 54 Mbps | 11 Mbps | 54 Mbps | 65 to 600 Mbps | 78 Mbps to 3.2 Gbps | 6.76 Gbps |
| Frequency band | 5 GHz | 2.4 GHz | 2.4 GHz | 2.4 or 5 GHz | 5 GHz | 60 GHz |
| Channel bandwidth | 20 MHz | 20 MHz | 20 MHz | 20, 40 MHz | 40, 80, 160 MHz | 2160 MHz |
| Antenna configuration | 1 × 1 SISO | 1 × 1 SISO | 1 × 1 SISO | Up to 4 × 4 MIMO | Up to 8 × 8 MIMO, MU-MIMO | 1 × 1 SISO |

The evolution of Wi-Fi from the Mbps range to the Gbps range has required the use of three key technologies to enable the higher data rate: *Multiple-Input, Multiple-Output* (MIMO) antennas, *Orthogonal Frequency-Division Multiplexing* (OFDM), and *Quadrature Amplitude Modulation* (QAM). In this article, we first introduce each of these technologies, with a brief mention of their evolution from simpler technologies, and then look at the two new Gigabit Wi-Fi standards.

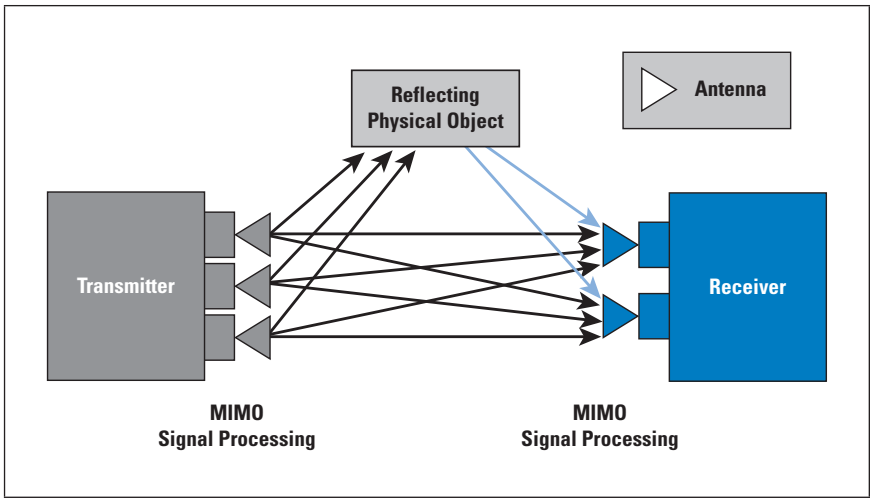
## MIMO Antennas

In traditional two-way communication between two wireless stations, each station employs a single antenna for transmission and reception, referred to as *Single-Input, Single-Output* (SISO). In any wireless communication system, there are numerous forms of transmission impairments to deal with, and these impairments become increasingly significant at higher data rates. Of particular concern are noise and *multipath* effects. The latter term refers to the fact that a transmitted signal may reach a destination antenna by not just a direct path but by one or more paths that involve a reflection between source and destination.

These multiple arriving paths interfere with each other and make recovery of the data from the signal more challenging. One effective approach is to use multiple antennas, either at the transmitting end or the receiving end, or both.

In a MIMO scheme, the transmitter and receiver employ multiple antennas[5]. The source data stream is divided into *n* substreams, one for each of the *n* transmitting antennas. The individual substreams are the input to the transmitting antennas (multiple inputs). At the receiving end, *m* antennas receive the transmissions from the *n* source antennas via a combination of line-of-sight transmission and multipath caused by reflection (Figure 1). The output signals from all of the *m* receiving antennas (multiple outputs) are combined. With a lot of complex math, the result is a much better received signal than can be achieved with either a single antenna or multiple frequency channels. Note that the terms *input* and *output* refer to the input to the transmission channel and the output from the transmission channel, respectively.

*Figure 1: MIMO Scheme*




MIMO systems are characterized by the number of antennas at each end of the wireless channel. Thus an 8 × 4 MIMO system has 8 antennas at one end of the channel and 4 at the other end. In configurations with a base station, such as a cellular network or a Wi-Fi hotspot, the first number typically refers to the number of antennas at the base station. There are two types of MIMO transmission schemes:

- *Spatial diversity:* The same data is coded and transmitted through multiple antennas, effectively increasing the power in the channel proportional to the number of transmitting antennas. This process improves the *Signal-to-Noise Ratio* (SNR) for cell edge performance. Further, diverse multipath fading offers multiple "views" of the transmitted data at the receiver, thus increasing robustness. In a multipath scenario where each receiving antenna would experience a different interference environment, there is a high probability that if one antenna is suffering a high level of fading, another antenna has sufficient signal level.

- *Spatial multiplexing:* A source data stream is divided among the transmitting antennas. The gain in channel capacity is proportional to the available number of antennas at the transmitter or receiver, whichever is less. Spatial multiplexing can be used when transmitting conditions are favorable and for relatively short distances compared to spatial diversity. The receiver must do considerable signal processing to sort out the incoming substreams, all of which are transmitting in the same frequency channel, and to recover the individual data streams.

*Multiple-user MIMO* (MU-MIMO) extends the basic MIMO concept to multiple endpoints, each with multiple antennas. The advantage of MU-MIMO compared to single-user MIMO is that the available capacity can be shared to meet time-varying demands. MU-MIMO techniques are used in both Wi-Fi and *Fourth-Generation* (4G) cellular networks.

MU-MIMO has two applications:

- *Uplink—Multiple Access Channel (MAC):* Multiple end users transmit simultaneously to a single base station.
- *Downlink—Broadcast Channel (BC):* The base station transmits separate data streams to multiple independent users.

MIMO-MAC is used on the uplink channel to provide multiple access to subscriber stations. In general, MIMO-MAC systems outperform point-to-point MIMO, particularly if the number of receiver antennas is greater than the number of transmit antennas at each user. A variety of multiuser detection techniques are used to separate the signals transmitted by the users.

MIMO-BC is used on the downlink channel to enable the base station to transmit different data streams to multiple users over the same frequency band. MIMO-BC is more challenging to implement. The techniques employed involve processing of the data symbols at the transmitter to minimize interuser interference.

### OFDM, OFDMA, and SC-FDMA

The technologies discussed in this section all derive from one of the oldest techniques used in communications: *Frequency-Division Multiplexing* (FDM). FDM simply means the division of a transmission facility into multiple channels by splitting the frequency band transmitted by the facility into narrower bands, each of which is used to constitute a distinct channel. Common examples of FDM are cable TV, broadcast radio, and broadcast television.

A common application of FDM is *Frequency-Division Multiple Access* (FDMA), which is a technique used to share the spectrum among multiple stations. In a typical configuration, a base station communicates with numerous subscriber stations.

Such a configuration is found in satellite networks, cellular networks, Wi-Fi, and WiMAX. Typically, the base station assigns bandwidths to stations within the overall bandwidth available. Key features of FDMA include the following:

- Each channel is dedicated to a single station; it is not shared.
- If a channel is not in use, it is idle and the capacity is wasted.
- Individual channels must be separated by guard bands to minimize interference.

Thus, this scheme divides the available bandwidth into multiple nonoverlapping bands, or channels, as with FDM. The channels are allocated across multiple stations, thus allowing multiple access to the available bandwidth.

*Orthogonal Frequency-Division Multiplexing* (OFDM), also called *multicarrier modulation,* is a form of FDM in which a single data stream transmits over the available bandwidth, sending some of the bits on each channel. Thus, with OFDM, all of the channels are dedicated to a single data source.

Suppose we have a data stream operating at $R$ bps and an available bandwidth of $NB$, centered at $f$. The entire bandwidth could be used to send the data stream, in which case each bit duration would be $1/R$. The alternative is to split the data stream into $N$ substreams, using a serial-to-parallel converter. Each substream has a data rate of $R/N$ bps and is transmitted on a separate subcarrier, with spacing between adjacent subcarriers of $B$. Now the bit duration is $N/R$.

The OFDM scheme uses advanced digital-signal-processing techniques to distribute the data over multiple carriers at precise frequencies. The relationship among the subcarriers is referred to as *orthogonality.* The result is that the peaks of the power spectral density of each subcarrier occur at a point at which the power of other subcarriers is zero. With OFDM, the subcarriers can be packed tightly together because there is minimal interference between adjacent subcarriers.

OFDM has several advantages. First, frequency selective fading affects only some subcarriers and not the whole signal. If the data stream is protected by a forward error-correcting code, this type of fading is easily handled. More important, OFDM overcomes *Intersymbol Interference* (ISI) in a multipath environment. ISI has a greater impact at higher bit rates, because the distance between bits, or symbols, is smaller. With OFDM, the data rate is reduced by a factor of $N$, increasing the symbol time by a factor of $N$. Thus, if the symbol period is $T$ for the source stream, the period for the OFDM signals is $NT$. This modulation scheme dramatically reduces the effect of ISI. As a design criterion, $N$ is chosen so that $NT$ is significantly greater than the root-mean-square delay spread of the channel.

A variant of OFDM is *Orthogonal Frequency-Division Multiple Access* (OFDMA). Like OFDM, OFDMA employs multiple closely spaced subcarriers, but the subcarriers are divided into groups of subcarriers. Each group is named a subchannel. The subcarriers that form a subchannel need not be adjacent. In the downlink, different subchannels may be intended for different receivers. In the uplink, a transmitter may be assigned one or more subchannels.

Subchannelization defines subchannels that can be allocated to *Subscriber Stations* (SSs) depending on their channel conditions and data requirements. Using subchannelization, within the same time slot a *Base Station* (BS) can allocate more transmit power to user devices (SSs) with lower SNR, and less power to user devices with higher SNR. Subchannelization also enables the BS to allocate higher power to subchannels assigned to indoor SSs, resulting in better in-building coverage. Subchannels are further grouped into *bursts,* which can be allocated to wireless users. Each burst allocation can be changed from frame to frame as well as within the modulation order, allowing the base station to dynamically adjust the bandwidth usage according to the current system requirements.

Subchannelization in the uplink can save user-device transmit power because it can concentrate power only on certain subchannel(s) allocated to it. This power-saving feature is particularly useful for battery-powered user devices.

Another variant of OFDM is *Single-Carrier FDMA* (SC-FDMA), which is a relatively recently developed multiple access technique with similar structure and performance to OFDMA. One prominent advantage of SC-FDMA over OFDMA is the lower *Peak-to-Average Power Ratio* (PAPR) of the transmit waveform, which benefits the mobile user in terms of battery life and power efficiency. OFDMA signals have a higher PAPR because, in the time domain, a multicarrier signal is the sum of many narrowband signals. At some time instances, this sum is large and at other times small, meaning that the peak value of the signal is substantially larger than the average value.

Thus, SC-FDMA is superior to OFDMA. However, it is restricted to uplink use because the increased time-domain processing of SC-FDMA would entail considerable burden on the base station. SC-FDMA performs a complex digital-signal-processing operation, which spreads the data symbols over all the subcarriers carrying information and produces a virtual single-carrier structure. This structure then is passed through the OFDM processing modules to split the signal into subcarriers. Now, however, every data symbol is carried by every subcarrier.

For OFDM, a source data stream is divided into *N* separate data streams and these streams are modulated and transmitted in parallel on *N* separate subcarriers, each with bandwidth *B*. The source data stream has a data rate of *R* bps, and the data rate on each subcarrier is *R/N* bps. For SC-FDMA, it appears that the source data stream is modulated on a single carrier (hence the SC prefix to the name) of bandwidth *N* x *B* and transmitted at a data rate of *R* bps. The data is transmitted at a higher rate, but over a wider bandwidth compared to the data rate on a single subcarrier of OFDM. However, because of the complex signal processing of SC-FDMA, the preceding description is not accurate. In effect, the source data stream is replicated *N* times, and each copy of the data stream is independently modulated and transmitted on a subcarrier, with a data rate on each subcarrier of *R* bps. Compared with OFDM, we are transmitting at a much higher data rate on each subcarrier, but because we are sending the same data stream on each subcarrier, it is still possible to reliably recover the original data stream at the receiver.

A final observation concerns the term *multiple access*. With OFDMA, it is possible to simultaneously transmit either from or to different users by allocating the subcarriers during any one time interval to multiple users. This transmission is not possible with SC-FDMA: At any given point in time, all of the subcarriers are carrying the identical data stream and hence must be dedicated to one user. But over time, it is possible to provide multiple access by allocating the bandwidth to different users at different times.

### Quadrature Amplitude Modulation

To transmit digital data over an analog signal, such as a Wi-Fi radio signal, it is necessary to encode the data onto the signal by some form of modulation. The simplest approach is to provide two different signals to be transmitted during a bit time, with one signal element representing binary one and one representing binary zero. Thus, *Amplitude Shift Keying* (ASK) involves transmitting a constant-frequency signal but varying the signal amplitude between two values. With *Phase Shift Keying* (PSK), two different phase shifts of the same carrier frequency are used to represent the two binary digits. As the data rate increases, the length of each signal element representing a single bit shortens. That is, the signal element is shorter both in duration and in physical length while being transmitted. Thus, a short noise burst or a short transmission impairment of any sort affects more bits as the data rate increases. One standard way of coping with this problem is to encode more than a single bit in each signal element. For example, if four amplitudes are used instead of two, then each signal element can encode two bits. One of the most effective techniques for encoding multiple bits per signal element is *Quadrature Amplitude Modulation* (QAM).

QAM uses two basic principles for encoding digital data onto an analog signal: ASK and PSK. QAM takes advantage of the fact that it is possible to send two different signals simultaneously on the same carrier frequency by using two copies of the carrier frequency, one shifted by 90° with respect to the other. For QAM, each carrier is ASK modulated. The two independent signals are simultaneously transmitted over the same medium. At the receiver, the two signals are demodulated and the results are combined to produce the original binary input.

If two-level ASK is used, then each of the two streams can be in one of two states and the combined stream can be in one of $4 = 2 \times 2$ states. If four-level ASK is used (that is, four different amplitude levels), then the combined stream can be in one of $16 = 4 \times 4$ states. This modulation is known as 16-QAM. Systems using 64 (64-QAM) and even 256 states have been implemented. The greater the number of states, the higher the data rate that is possible within a given bandwidth. However, the greater the number of states, the higher the potential error rate due to noise and attenuation.

### IEEE 802.11ac

IEEE 802.11ac operates in the 5-GHz band, as do the older and slower standards 802.11a and 802.11n. It is designed to provide a smooth evolution from 802.11n. This new standard uses advanced technologies in antenna design and signal processing to achieve much greater data rates, at lower battery consumption, all within the same frequency band as the older versions of Wi-Fi. The new standard achieves much higher data rates than 802.11n by means of enhancements in three areas:

- *Bandwidth:* The maximum bandwidth of 802.11n is 40 MHz; the maximum bandwidth of 802.11ac is 160 MHz.
- *Signal encoding:* The 802.11n standard uses 64 QAM with OFDM, and 802.11ac uses 256 QAM with OFDM. Thus, more bits are encoded per symbol. Both schemes use forward error correction with a code rate of 5/6 (ratio of data bits to total bits).
- *MIMO:* With 802.11n, the maximum number of antennas is 4 channel input and 4 channel output antennas. The 802.11ac standard increases this maximum to $8 \times 8$.

Two other changes going from 802.11n to 802.11ac are noteworthy. The 802.11ac standard includes the option of MU-MIMO, meaning that on the downlink, the transmitter can use its antenna resources to transmit multiple frames to different stations, all at the same time and over the same frequency spectrum. Thus, each antenna of a MU-MIMO access point can simultaneously communicate with a different single-antenna device, such as a smartphone or tablet, thereby enabling the access point to deliver significantly more data in many environments.

### IEEE 802.11ad

IEEE 802.11ad is a version of 802.11 operating in the 60-GHz frequency band. This band offers the potential for much wider channel bandwidth than the 5-GHz band, enabling high data rates with relatively simple signal encoding and antenna characteristics. Few devices operate in the 60-GHz band, meaning that communication experiences less interference than in the other bands used for Wi-Fi. However, at 60 GHz, 802.11ad operates in the millimeter range, resulting in some undesirable propagation characteristics:

- Free space loss increases with the square of the frequency, so losses are much higher in this range than in the ranges used for traditional microwave systems.
- Multipath losses can be quite high. *Reflection* occurs when an electromagnetic signal encounters a surface that is large relative to the wavelength of the signal; *scattering* occurs if the size of an obstacle is on the order of the wavelength of the signal or less; and *diffraction* occurs when the wavefront encounters the edge of an obstacle that is large compared to the wavelength.
- Millimeter-wave signals generally don't penetrate solid objects.

For these reasons, 802.11ad is likely to be useful only within a single room. Because it can support high data rates and, for example, could easily transmit uncompressed high-definition video, it is suitable for applications such as replacing wires in a home entertainment system, or streaming high-definition movies from your cell phone to your television.

### Prospects for Gigabit Wi-Fi

Gigabit Wi-Fi holds attractions for both office and residential environments, and commercial products are beginning to roll out. In the office environment, the demand for ever greater data rates has led to Ethernet offerings at 10 Gbps, 40 Gbps, and most recently 100 Gbps. These stupendous capacities are needed to support blade servers, heavy reliance on video and multimedia, and multiple offsite broadband connections. At the same time, the use of wireless LANs has grown dramatically in the office setting to meet needs for mobility and flexibility. With the gigabit-range data rates available on the fixed portion of the office LAN, gigabit Wi-Fi is needed to enable mobile users to use the office resources effectively. IEEE 802.11ac is likely to be the preferred gigabit Wi-Fi option for this environment.

In the consumer and residential market, IEEE 802.11ad is likely to be popular as a low-power, short-distance wireless LAN capability with little likelihood of interfering with other devices. IEEE 802.11ad is also an attractive option in professional media production environments in which massive amounts of data need to be moved short distances.

## References

[1] Garber, L., "Wi-Fi Races into a Faster Future," *Computer,* March 2012.

[2] Alsabbagh, E.; Yu, H.; and Gallagher, K., "802.11ac Design Consideration for Mobile Devices," *Microwave Journal,* February 2013.

[3] Cordeiro, C.; Akhmetov, D.; and Park, M., "IEEE 802.11ad: Introduction and Performance Evaluation of the First Multi-Gbps WiFi Technology," *Proceedings of the 2010 ACM International Workshop on mmWave Communications: From Circuits to Networks,* 2010.

[4] Perahia, E., et al., "IEEE 802.11ad: Defining the Next Generation Multi-Gbps Wi-Fi," *Proceedings, 7th IEEE Consumer Communications and Networking Conference,* 2010.

[5] Halperin, D., et al., "802.11 with Multiple Antennas for Dummies," *Computer Communication Review,* January 2010.

[6] Danielyan, E., "IEEE 802.11," *The Internet Protocol Journal,* Volume 5, No. 1, March 2002.

[7] Sridhar, T., "Wi-Fi, Bluetooth and WiMAX—Technology and Implementation," *The Internet Protocol Journal,* Volume 11, No.4, December 2008.

WILLIAM STALLINGS is an independent consultant and author of numerous books about 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`**

# A Question of DNS Protocols

*by Geoff Huston, APNIC*

One of the most prominent *Denial of Service* attacks in recent years was one that occurred in March 2013, launched against Spamhaus and Cloudflare.

One description of this attack is at [1]. I'm not sure about the claim that this attack "almost broke the Internet," but with a peak volume of attack traffic of some 120 Gbps, it was nevertheless a very significant attack.

How did the attackers generate such massive volumes of attack traffic? The answer lies in the *Domain Name System* (DNS). The attackers asked about domain names, and the DNS system answered. Something we all do all the time on the Internet. So how can a conventional activity of translating a domain name into an IP address be turned into a massive attack?

A few aspects of the DNS make it a readily coercible means of generating an attack:

- The DNS uses very simple *User Datagram Protocol* (UDP) transactions, where the clients send queries to resolvers, and resolvers send back responses.
- Within the DNS it is possible to send a relatively small query packet and get the resolver to reply with a much larger response. The DNS resolver becomes, in effect, a traffic "amplifier."
- There are many so-called "open resolvers," who are willing to respond to queries from any clients on the Internet. In other words, these resolvers will accept DNS queries from anyone and send DNS responses to anyone. The *Open Resolver Project*[2] claims that there are some 28 million open resolvers on the Internet at present that represent "a significant threat." That's a disturbingly high number if you are worried about ways to subvert the DNS to launch a platform for such attacks.
- Finally, it appears that way too few networks implement source address egress filtering, as described in *Best Current Practice* (BCP) 38[3]. This mechanism is a packet filtering mechanism, which if universally implemented, would make it far more challenging to mount attacks that relied on the ability to lie about the source address in an IP packet. If a network implemented the measures described in BCP 38, the network could emit only packets whose source address is reachable through the same network.

The combination of these four factors produces a comprehensive vulnerability for the Internet. In performing experiments about the behavior of the DNS, we see a background level of DNS "probes" that contain a query for "ANY," often querying the domain `isc.org`. In this case the original UDP request of 64 bytes generates a UDP response of 3,475 bytes, or an amplification factor of 54. If an attacker can send this DNS UDP query to 100 of these open resolvers each second, using a common source address of the intended victim, then the attacker will be generating a query traffic volume of some 51 Kbps, and the victim will receive an incoming traffic load of 2.78 Mbps. If you then enlist a bot army to replicate this simple attack one thousand-fold, then the attack traffic volume has exceeded a gigabit per second. If a query for a domain name that is signed using the *Domain Name System Security Extensions* (DNSSEC) is used, where the query explicitly requests the DNSSEC signature information in the response, the response sizes are far larger, and the unwitting accomplices in such attacks can expand to include the authoritative name servers of DNSSEC-signed domains.

The problem with this attack vector is that, in flight, the traffic looks like all other traffic. These responses are simple DNS responses, and the network carries DNS responses as one of the more common packet types. So simple filtering of all DNS traffic is simply not an option, and we need to look deeper to see how we could mitigate this rather worrisome vulnerability.

This development is not a recent one, and the attack vector has been used for many years. For example, a presentation on this topic was given in May 2006 at an IETF meeting[4]. The major difference between then and now is that the estimated number of open resolvers that can be used to assist in the attack has jumped from 500,000 to in excess of 25 million!

This topic has appeared in numerous threads of discussion.

One thread of conversation goes along the lines that if everyone implemented the measures described in BCP 38, endpoints would be prevented from emitting DNS query packets with a false source address, thereby preventing these reflection attacks to be mounted using the DNS. Of course this conversation is long-standing, in fact older than BCP 38 itself. BCP 38 is now 13 years old as a document, and the somewhat worrisome observation is that nothing much appears to have happened in terms of improving the situation about source address spoofing over the past 13 years, so there is not a lot of optimism that anything will change in the coming months, if not years.

Another conversation thread says that resolvers should implement *Response Rate Limiting* (RRL), and silently discard repetitive queries that exceed some locally configured threshold. Although this solution is relatively effective in the case of authoritative name servers, it is less effective in the face of a massive pool of open recursive resolvers, because in this latter case the query load can be spread across the entire resolver pool so that each resolver may not experience a detectable level of repeated queries. It is also possible to use a very large pool of authoritative name servers in the same manner. However, this consideration does not weaken the advice that authoritative name servers should implement RRL in any case. It just lowers the level of optimism that this measure alone would eliminate this vulnerability. (Randy Bush gave an informative presentation at the 2013 *Asia Pacific Regional Internet Conference on Operational Technologies* (APRICOT) conference[5], illustrating the before and after states of the application of RRL for an authoritative name server.)

Still another thread of conversation is exploring ways to shut down the pool of open recursive resolvers. The *Open Resolver Project*[2] is working on a "name and shame" approach to the problem, and is hopeful that if individual resolver operators are allowed to check their own status, these resolvers will be closed down. Like the universal adoption of BCP 38, it's hard to be overly optimistic about this approach. Part of the problem is that a large volume of unmanaged or semi-managed systems are connected to the Internet, and the vulnerabilities they create are not necessarily known to the parties who deployed these systems in the first place. For example, one way to create more robust "plug and play" systems is to reduce the amount of environmental configuration that needs to be loaded into such systems in the first place. Equipping such standalone units with their own DNS resolver appears to be a way to remove an additional configuration element from the unit. The downside is that if these units are configured as open recursive resolvers, then they become part of the overall problem of the massive population of open resolvers on today's Internet.

The behavior of the DNS where a small-size query generates a large response is one that appears to be intrinsic to the DNS, and particularly more so with the use of DNSSEC-signed domain names. If we want some form of security in the DNS so that the client can be assured that the response it receives is authentic and current and has not been tampered with in any way, then the overheads of cryptographic signature blocks and the size these signature blocks take up appears to be an intrinsic part of the security architecture of DNS. We appear to want a lightweight, fast DNS, so we like the performance properties of a DNS resolution framework that uses UDP, coupled with a ubiquitous distribution of recursive resolvers and the associated resolver caches.

But we also want DNS responses that can be verified, so we like DNSSEC. So the responses get larger, and the outcome is that the DNS is a highly effective platform for massive traffic attacks where there is a very limited space to mitigate the associated risks.

If we want to close the door on using the DNS to mount large-scale attacks, there does not appear to be much space left to maneuver here. However, there is a conversation that has not quite petered out yet. That conversation is about the use of the *Transmission Control Protocol* (TCP) in the DNS.

The original specification of the DNS[6] allowed for the use of both UDP and TCP as the transport service for DNS queries and responses. The relevant discussion of this specification in "Requirements for Internet Hosts—Application and Support"[7] reads:

> "... it is also clear that some new DNS record types defined in the future will contain information exceeding the 512 byte limit that applies to UDP, and hence will require TCP. Thus, resolvers and name servers should implement TCP services as a backup to UDP today, with the knowledge that they will require the TCP service in the future."

Why 512 bytes? The 512-byte limit was based on the IPv4 "Requirements for Internet Hosts—Communication Layers,"[8], where it is required that all IPv4 host systems must accept an IP packet that is at least 576 octets in size. Allowing for 20 bytes of IP header, room for the maximum of 40 bytes of IP options and 8 bytes of UDP header, the implication is that the maximum payload in a UDP packet that will be accepted by all IPv4 hosts is restricted to 512 bytes.

It should be noted that it is theoretically possible that there are links that do not support IP packets of 576 bytes, so even these 576-byte IP packets may possibly be fragmented in flight. The limit being referred to here is the largest (possibly reassembled) packet that a host will assuredly accept.

Now of course it is possible to generate larger packets in IPv4, to a theoretical maximum of 65,535 bytes, which, by the same reckoning, accommodates a UDP payload of 65,507 bytes, but such larger packets will probably be fragmented in flight. In such a case, when this behavior is combined with typical firewall behavior, then the trailing packet fragments may not be passed onward to a client at all, because many firewall configurations use acceptance rules based on the UDP and TCP port numbers. Because the trailing fragments of a fragmented IP datagram have no UDP or TCP packet header, the firewall is left with a quandary. Should the firewall simply accept all IP fragments? In this case the security role of the firewall may be compromised through the transmission of fragments that form part of a hostile attack.

Or should the firewall discard all IP fragments? In this case a sender of a large packet should be aware that if there is fragmentation, then the trailing packet fragments may not be passed to the receiver. So with the two considerations that hosts are not required to accept UDP datagrams that are larger than 576 bytes, and firewalls may discard trailing fragments of a fragmented IP datagram, the original DNS response to this situation was to limit all of its UDP responses to 512 bytes, and always use TCP as the backup plan if the DNS response was larger than 512 bytes.

However, clients may not know in advance that a DNS response is larger than 512 bytes. To signal to a client that it should use TCP to retrieve the complete DNS response, when the response would be greater than 512 bytes the DNS resolver will respond in UDP with a partial response that fits within the 512-byte limit, and set the "truncated" bit in the DNS flags that form part of the response.

We continued for some years with this approach. The DNS used UDP for the bulk of its transactions, which all fit within the 512-byte limit, and for the relatively infrequent case where larger DNS responses were being generated, the UDP response was truncated, and the client was expected to repeat the question over a TCP connection.

This situation was not altogether comfortable when we then considered adding security credentials to the DNS through DNSSEC. The inclusion of digital signatures in DNS responses implied that very few DNSSEC responses would fit within this 512-byte limit. But if the process of switching to TCP was to respond to the UDP query with a UDP response that essentially said "Please use TCP," then this response adds a considerable delay to the DNS function. Each query would now involve a minimum of three round-trip time interactions with the server, rather than just the single round-trip time interval for UDP (a TCP transaction still includes one round-trip time transaction for the UDP query and truncated UDP response, one round-trip interval for the TCP connection establishment handshake, and one for the TCP query and response). The next refinement of the DNS was to include a way to signal that a client was able to handle larger DNS responses in UDP, and thereby bypass the fall-back to TCP in the case where the client is able to handle larger IP packets and the client is confident that there is no IP fragment filtering in intervening middleware.

As pointed out in RFC 5966[9]:

> "Since the original core specifications for DNS were written, the *Extension Mechanisms for DNS* (EDNS0)[10] have been introduced. These extensions can be used to indicate that the client is prepared to receive UDP responses larger than 512 bytes. An EDNS0-compatible server receiving a request from an EDNS0-compatible client may send UDP packets up to that client's announced buffer size without truncation."

EDNS0 allows for the UDP-based DNS response to grow to far higher sizes. As a result, it appears that the DNS largely uses UDP and EDNS0, and EDNS0 is used to allow for the larger-size responses, most commonly set at 4096 bytes. TCP is now often regarded as a somewhat esoteric option, being used only for DNS zone transfer operations, and if the zone does not want to support zone transfers as a matter of local policy, then it is commonly thought that the role of TCP is no longer essential for the DNS.

Section 6.1.3.2 of "Requirements for Internet Hosts—Application and Support"[7] states:

> "DNS resolvers and recursive servers MUST support UDP, and SHOULD support TCP, for sending (non-zone-transfer) queries."

In the world of standards specifications and so-called normative language, that "SHOULD" in the quoted text is different from a "MUST." It's a little stronger than saying "well, you can do that if you want to," but it's a little weaker than saying "You really have to do this. There is no option not to." Little wonder that some implementors of DNS resolvers and some folks who configure firewalls came to the conclusion that DNS over TCP was an optional part of the DNS specification that does not necessarily need to be supported.

But DNS over TCP is not just a tool to allow for large DNS responses. If we review the preconditions for the use of the DNS in large-scale reflector attacks, namely the widespread support of UDP for large packet responses, and the relatively sparse use of BCP 38, then we've effectively allowed attackers to mount reflection attacks by co-opting a large set of open resolvers to send their responses to the target system, by using UDP queries whose IP source address is the IP address of the intended victim.

TCP does not have the same vulnerability. If an attacker were to attempt to open a TCP session using an IP source address of the intended victim, the victim would receive a short IP packet (IP and TCP header only, which is a 40-byte packet) containing only the SYN and ACK flags set. Because the victim system has no preexisting state for this TCP connection, it will discard the packet. Depending on the local configuration, it may send a TCP RESET to the other end to indicate that it has no state, or the discard of this unsolicited packet may be completely silent. This behavior removes one of the essential preconditions for a reflector amplification attack. If the attack traffic with the spoofed source address of the intended victim uses a 40-byte SYN TCP packet, then the victim will receive a 40-byte SYN/ACK TCP packet. The DNS attack amplification factor would be effectively removed.

If the DNS represents such a significant vulnerability for the Internet through these UDP-based reflection attacks, then does TCP represent a potential mitigation? Could we realistically contemplate moving away from the ubiquitous use of EDNS0 to support large DNS responses in UDP, and instead use DNS name servers that limit the maximal size of their UDP responses, and turn to TCP for larger responses? Could we contemplate a rate-limiting approach where, instead of not responding to what are possibly considered to be "excess" queries, the DNS server responds with a truncated UDP response to indicate that the client should use TCP and repeat the query?

Again, let's turn to RFC 5966:

> "The majority of DNS server operators already support TCP, and the default configuration for most software implementations is to support TCP. The primary audience for this document is those implementors whose failure to support TCP restricts interoperability and limits deployment of new DNS features."

The question we are looking at here is: Can we quantify the extent to which DNS resolvers are capable of using TCP for DNS queries and responses? How big is this "majority of DNS server operators" being referred to in the quote?

### The Experiment

We conducted an experiment using a modified DNS name server, where the maximal UDP packet size was configured to 512 bytes, and then set up an experiment where a simple query to resolve a DNS name would generate a response that could not fit within 512 bytes.

Although such a response is relatively easy trigger if it includes DNSSEC, if we want to set up a condition where all DNS responses are larger than 512 bytes for a domain, then we need to use a slightly different approach. The approach used in this iteration of the experiment is to use the DNS name alias function, the *Canonical Name* (CNAME) record.

Here is a sample zone:

```
$TTL1h
@ IN    SOA   nsx.dotnxdomain.net. research.apnic.net. (
                 2013011406       ; Serial
                 3600             ; Refresh
                 900              ; Retry
                 1                ; Expire
                 1 )              ; Minimum
  IN    NS    nsz1.z.dotnxdomain.net.

  z1    IN    A     199.102.79.186

  *     IN    A     199.102.79.186

4a9c317f.4f1e706a.6567c55c.0be33b7b.2b51341.a35a853f.59c4df1d.3b069e4e.87ea53bc.2b4cfb
4f.987d5318.fc0f8f61.3cbe5065.8d9a9ec4.1ddfa1c2.4fee4676.1ffb7fcc.ace02a11.a3277bf4.22
52b9ed.9b15950d.db03a738.dde1f863.3b0bf729 IN CNAME
33d23a33.3b7acf35.9bd5b553.3ad4aa35.09207c36.a095a7ae.1dc33700.103ad556.3a564678.16395
067.a12ec545.6183d935.c68cebfb.41a4008e.4f291b87.479c6f9e.5ea48f86.7d1187f1.7572d59a.9
d7d4ac3.06b70413.1706f018.0754fa29.9d24b07c

33d23a33.3b7acf35.9bd5b553.3ad4aa35.09207c36.a095a7ae.1dc33700.103ad556.3a564678.16395
067.a12ec545.6183d935.c68cebfb.41a4008e.4f291b87.479c6f9e.5ea48f86.7d1187f1.7572d59a.9
d7d4ac3.06b70413.1706f018.0754fa29.9d24b07c IN A 199.102.79.187
```

The use of the combination of long label strings and a CNAME construct forces a large response, which, in turn, triggers DNS to send a truncated UDP response in response to a conventional query for the address record for the original domain name. The truncated UDP response should force the client resolver to open a TCP session with the name server, and ask the same query again.

To ensure that the authoritative name server directly processed every name query, we used unique labels for each presented experiment, and ensured that the DNSSEC singed zones were also unique, ensuring that local DNS resolver caches could not respond to these name queries.

The first question we are interested in is: How many clients were able to successfully switch to TCP following the receipt of a truncated response in UDP?

We conducted an experiment by embedding numerous test cases inside an online ad, and over 8 days at the end of July 2013 we presented these tests to 2,045,287 end clients. We used four experiments. Two experiments used a name where the query and response would fit within a 512-byte payload as long as the query did not include a request for DNSSEC. One of these domain names was unsigned, and the other was signed. The other two experiments used the CNAME approach to ensure that the response would be larger than 512 bytes.

Again, one zone was signed, and the other was unsigned (Table 1).

*Table 1: DNS Resolution Using TCP, with CNAME Names*

| Experiment | UDP Queries | Truncated UDP Responses | TCP Responses | Truncated UDP to TCP Fail |
|---|---|---|---|---|
| Short, unsigned | 2,029,725 | 2 | 6 | 0 |
| Short, signed | 2,037,563 | 1,699,935 (83.4%) | 1,660,754 (81.5%) | 39,101 (1.9%) |
| Long, unsigned | 2,023,205 | 2,021,212 (99.9%) | 1,968,927 (97.3%) | 52,285 |
| Long, signed | 2,033,535 | 2,032,176 (99.9%) | 1,978,396 (97.3%) | 53,780 (2.6%) |

This data appears to point to a level of failure to follow up from a truncated UDP response to a TCP connection of some 2.6 percent of clients.

That level of failure to switch from a truncated UDP response to rephrase the query over TCP is large enough to be significant. The first question: Is this failure due to some failure of the DNS authoritative name server or a failure of the client resolver? If the name server is experiencing a high TCP session load, it will reject new TCP sessions by responding to incoming TCP session establishment packets with a TCP RESET. We saw no evidence of this session overload behavior in the packet traces that the authoritative name server gathered. So the TCP failure looks to be occurring closer to the client resolver than to the authoritative name server.

We can also look at this example from the perspective of the set of DNS resolvers. How many resolvers will switch to use TCP when they receive a UDP response that is truncated? Before looking at the results, it needs to be noted that the only resolvers that are exposed in this experiment are those resolvers that directly query our authoritative name server (*visible* resolvers). If a resolver is configured to forward its queries onto a recursive resolver, then its behavior will not be directly exposed in this experiment. It should also be noted that even when the visible recursive resolver is forced to use TCP to query the authoritative name server, this resolver may still relay the response back to its client by UDP, using EDNS0 to ensure that the larger UDP response is acceptable to the client. Thus the scope of these measurements refers specifically to this subset of resolvers who are visible resolvers.

The 2 million clients in this experiment used a total of 80,505 visible resolvers. Some 13,483 resolvers, or 17 percent of these visible resolvers, did not generate any TCP transactions with the authoritative name server. These 13,483 UDP-only resolvers made a total of 4,446,670 queries, and of these some 4,269,495 responses were truncated, yet none of these resolvers switched to TCP.

There is a second class of filtering middleware that operates on incoming traffic. In such cases the authoritative server sees an incoming TCP SYN packet to establish the DNS connection, and the server responds with a SYN+ACK packet. Because this packet will be blocked by the filtering middleware, it will never get passed through to the resolver client, and the TCP connection will not be established. It has been observed in discussions on DNS reliability that some security middleware permits only inbound traffic on UDP port 53, and discards inbound TCP port 53 traffic. The way in which this filtering behavior would manifest itself at the authoritative name server is that the name server would see and respond to the initial TCP SYN, and not see the ensuing TCP ACK that would complete the TCP handshake. This SYN-only behavior was observed in just 337 resolvers, a count that represents 0.4 percent of the set of visible resolvers. These resolvers generated a total of 1,719,945 queries, and received 1,575,328 truncated UDP responses.

Why is the client-level TCP failure rate at 2.6 percent of clients, whereas at the resolver level the TCP failure rate is 17 percent of visible resolvers? There are at least three possible reasons for this result:

- First, in some cases we observe service providers using DNS forwarder farms, where queries are spread across many query engines. When a DNS query is rephrased using TCP, it may not use the same forwarder to make the query.
- Second, we should factor in end-client failover to another DNS resolver that can support DNS transactions over TCP. Most clients are configured with multiple resolvers, and when one resolver fails to provide a response the client asks the query of the second and subsequent resolvers in its resolver set. If any of the visible resolvers associated with the resolvers listed in the client's resolver set are capable of using TCP, then at some stage we will see a TCP transaction at the authoritative name server. In this more prevalent case of TCP failure, either the resolver itself is not capable of generating a DNS query using TCP (presumably because of local limitations in the resolver software or local configuration settings), or some network middleware is preventing the resolver from performing TCP connections to port 53.
- Finally, the distribution of end clients across the set of visible resolvers is not even, and whereas some resolvers, such as the set used by Google's Public DNS service, serve some 7 precent of all end clients, others serve a single end client. We observed that 53,000 experiments, out of a total of 2 million experiments, failed to complete a TCP-based DNS resolution, so it is also possible that these 13,483 visible resolvers that do not support TCP queries are entirely consistent in volume with this level of end-client failure to resolve the DNS label of the experiment.

There is a slightly different way to look at this question. Although we saw some 53,000 experiments that failed to complete the DNS resolution at all, how many experiments were affected by this deliberate effort to force resolvers to use TCP? How many clients were affected in terms of longer DNS resolution time through the use of DNS resolvers that failed to switch to use TCP?

Table 2 shows that slightly more than 6 percent of all clients used a DNS resolver that was unable to repeat the DNS query over TCP. Some 75,000 clients used an alternate resolver that was capable of performing the TCP query, whereas the remainder were unable to resolve the DNS name at all.

*Table 2: Client Use of Resolvers That Fail to Complete a TCP Query*

| Experiment | UDP Queries | Truncated UDP Responses | TCP Responses | Truncated UDP to TCP Fail | Used TCP Fail Resolvers |
|---|---|---|---|---|---|
| Long, unsigned | 2,023,205 | 2,021,212 (99.9%) | 1,968,927 (97.3%) | 52,285 (2.6%) | 124,881 (6.1%) |
| Long, signed | 2,033,535 | 2,032,176 (99.9%) | 1,978,396 (97.3%) | 53,780 (2.6%) | 129,555 (6.4%) |

After running this initial experiment, we considered our use of the CNAME construct to inflate the DNS response to more that 512 bytes, and wondered if this additional DNS indirection created some problems for some resolver clients. Another approach to coerce client resolvers to use TCP is to modify the name-server code used by the authoritative name server, and drop its UDP maximum size to 275 octets, so that the name server will truncate the UDP response for any response of 276 bytes or larger. In this way a DNS query for the short unsigned name would fit within the new UDP limit, but in all other cases the UDP response would be truncated.

The results we saw for this second experiment, which removed the CNAME entry and used an authoritative name server with a 275-byte UDP payload limit, with 3 days of collected data, are summarized in Table 3.

*Table 3: DNS Resolution Using TCP, Using 275-Byte UDP Truncation*

| Experiment | UDP Queries | Truncated UDP Responses | TCP Responses | Truncated UDP to TCP Fail |
|---|---|---|---|---|
| Short, unsigned | 936,007 | 0 | 3 | 3 |
| Short, signed | 936,116 | 936,116 (100.0%) | 911,751 (97.4%) | 24,365 (2.6%) |
| Long, unsigned | 920,613 | 920,613 (100.0%) | 896,953 (97.4%)) | 23,530 (2.6%) |
| Long, signed | 934,446 | 934,446 (100.0%) | 910,757 (97.5%) | 25,573 (2.5%) |

These results are consistent with the results of the original experiment, indicating that the use of the CNAME construct is not causing additional problems for resolver clients.

### Conclusion

The original specification of the DNS called for resolvers to use UDP when the response was 512 bytes or smaller, and TCP was to be used for larger DNS transactions. DNS clients would interpret the truncated flag in a DNS UDP response to trigger a re-query using TCP.

With the introduction of EDNS0, clients can now signal their capability to accept larger UDP datagrams, with the result that the fallback to TCP for large DNS responses is used less frequently, to the extent that there is now a concern that a significant set of clients cannot resolve a DNS name if that resolution operation is required to occur using TCP.

However, DNS UDP is being used in various forms of malicious attacks, using DNS queries where the response is far larger than the query. The combination of source address spoofing and DNS over UDP is presenting us with some significant concerns. For that reason there is a renewed consideration of the viability of reverting to TCP for various forms of larger DNS responses, which effectively prevents source address spoofing in the DNS query/response interaction.

In this experiment we've looked at the impact a forced switch to DNS over TCP would have on clients. In particular, what proportion of clients would no longer be able to complete a DNS name-resolution process if the process necessarily involves the use of TCP? Our measurements of a sample of 2 million clients in early August 2013 points to a DNS resolution failure rate for 2 percent of clients.

The picture for individual DNS resolvers appears to be somewhat worse, in that 17 percent of visible resolvers do not successfully follow up with a TCP connection following the reception of a truncated UDP response.

Although that 17-percent number is surprisingly high, there are two mitigating factors here.

It appears that clients use multiple DNS resolvers in their local DNS configuration, so that failure of an initially selected resolver to respond to a query because of a lack of support for TCP may be resolved by the clients selecting the next resolver from their local resolver set. For this set of clients, which appears to encompass some 4 percent of the total client population, the penalty is increased DNS resolution time, where the resolution of a name requires the client to fail over to the other resolvers listed in their local DNS resolver set.

Secondly, the more extensively used visible DNS resolvers appear to be capable of supporting TCP-based queries, so the problems with TCP support in the DNS appear to be predominately concerned with resolvers that are used by a relatively small pool of end clients.

## References


[1] "The DDoS That Almost Broke the Internet," **http://blog.cloudflare.com/the-ddos-that-almost-broke-the-internet**

[2] **http://openresolverproject.org**

[3] Paul Ferguson, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing," RFC 2827, BCP 38, May 2000.

[4] Frank Scalzo, "Recent DNS Reflector Attacks From the Victim and the Reflector POV," **http://www.iepg.org/july2006/1-frank-scalzo.pdf**

[5] Randy Bush, "DNS Rate Limiting—a Hard Lesson," **http://conference.apnic.net/__data/assets/pdf_file/0011/58880/130226.apops-dns-rate-limit_1361839670.pdf**

[6] Paul V. Mockapetris, "Domain Names—Implementation and Specification," RFC 1035, November 1987.

[7] Robert Braden, "Requirements for Internet Hosts—Application and Support," RFC 1123, October 1989.

[8] Robert Braden, "Requirements for Internet Hosts—Communication Layers," RFC 1122, October 1989.

[9] Ray Bellis, "DNS Transport over TCP—Implementation Requirements," RFC 5966, August 2010.

[10] Paul Vixie, "Extension Mechanisms for DNS (EDNS0)," RFC 2671, August 1999.



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**

### IANA Transition

In March, 2014, the United States *National Telecommunications and Information Administration* (NTIA) announced its intent to "...transition Key Internet Domain Name Functions to the global multistakeholder community." See [1] for the full announcement.

Quoting from the announcement: "As the first step, NTIA is asking the *Internet Corporation for Assigned Names and Numbers* (ICANN) to convene global stakeholders to develop a proposal to transition the current role played by NTIA in the coordination of the Internet's *Domain Name System* (DNS).

NTIA's responsibility includes the procedural role of administering changes to the authoritative root zone file—the database containing the lists of names and addresses of all top-level domains—as well as serving as the historic steward of the DNS. NTIA currently contracts with ICANN to carry out the *Internet Assigned Numbers Authority* (IANA) functions and has a Cooperative Agreement with Verisign under which it performs related root zone management functions.

ICANN is uniquely positioned, as both the current IANA functions contractor and the global coordinator for the DNS, as the appropriate party to convene the multistakeholder process to develop the transition plan. NTIA has informed ICANN that it expects that in the development of the proposal, ICANN will work collaboratively with the directly affected parties, including the *Internet Engineering Task Force* (IETF), the *Internet Architecture Board* (IAB), the *Internet Society* (ISOC), the *Regional Internet Registries* (RIRs), top level domain name operators, VeriSign, and other interested global stakeholders.

NTIA has communicated to ICANN that the transition proposal must have broad community support and address the following four principles:

- Support and enhance the multistakeholder model;
- Maintain the security, stability, and resiliency of the Internet DNS;
- Meet the needs and expectation of the global customers and partners of the IANA services; and,
- Maintain the openness of the Internet.

While stakeholders work through the ICANN-convened process to develop a transition proposal, NTIA's current role will remain unchanged. The current IANA functions contract expires September 30, 2015."

Since the announcement, a lot of discussion has taken place in many fora and comments have been produced by various groups. Below we include some links to various documents, groups and events that captures some of these activities. The IANA transition is likely to remain a "hot topic" for the next couple of years.

### References


[0] Internet Assigned Numbers Authority (IANA)
**http://www.iana.org/about**

[1] "NTIA Announces Intent to Transition Key Internet Domain Name Functions,"
**http://www.ntia.doc.gov/press-release/2014/ntia-announces-intent-transition-key-internet-domain-name-functions**

[2] "IANA Functions and Related Root Zone Management Transition Questions and Answers,"
**http://www.ntia.doc.gov/other-publication/2014/iana-functions-and-related-root-zone-management-transition-questions-and-answ**

[3] "Comments of the Internet Architecture Board (IAB) Regarding the 'Draft Proposal, Based on Initial Community Feedback, of the Principles and Mechanisms and the Process to Develop a Proposal to Transition NTIA's Stewardship of the IANA Functions,'"
**http://www.iab.org/wp-content/IAB-uploads/2014/04/iab-response-to-20140408-20140428a.pdf**

[4] "NTIA IANA Functions' Stewardship Transition," ICANN microsite, **https://www.icann.org/stewardship**

[5] Jari Arkko, "ICANN and Transition of NTIA's Stewardship," IETF Blog,
**http://www.ietf.org/blog/2014/07/icann-and-transition-of-ntias-stewardship/**

[6] ARIN, "IANA Globalization," **http://teamarin.net/education/internet-governance/iana-globalization/**


### The Internet Protocol Journal Needs your Help

We are delighted to be publishing IPJ once again, but we cannot do so without your help. Please consider sponsoring the journal. We have a range of sponsorship levels to suit most budgets, just send an e-mail to **ipj@protocoljournal.org** for more information. We also need suggestions for topics as well as actual articles, so do get in touch!

# 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*

Dave Crocker, Jay Etchings, Dennis Jennings, Jim Johnston, Bill Manning, George Sadowsky, Helge Skrivervik, Rob Thomas, Tom Vest.

For more information about sponsorship, please contact **ipj@protocoljournal.org**

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

ADDRESS SERVICE REQUESTED

PRSRT STD
U.S. Postage
PAID
PERMIT No. 5187
SAN JOSE, CA



*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*