The Internet Protocol Journal

March 2007 Volume 10, Number 1

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From The Editor

Every time you dial into a service provider network or connect to a wired or wireless network that offers Internet access, you are most likely using several components of what is referred to as *Authentication, Authorization, and Accounting*, or "AAA" for short. The AAA space is quite complex, so when we asked Sean Convery to give us an overview of these technologies, he decided to divide his survey into two parts. Part One—subtitled "Concepts, Elements, and Approaches"—is included in this issue. Part Two, which discusses protocol details and applications, will follow in our next issue.

The *Domain Name System* (DNS) has been discussed previously in this journal. The most critical part of the DNS is the collection of *Root Servers*. For protocol reasons, there are only 13 "logical" root servers, but a system of more than 100 servers has been deployed using a technique known as *anycast*. Steve Gibbard examines the distribution of the root servers in different parts of the world and discusses operational aspects of the DNS.

If you are tracking any part of the IETF process, you should be aware of several important resources. First, the IETF *Education Team* (**`http://edu.ietf.org/`**) offers training sessions and educational materials. Second, the *IETF Journal* (**`http://www.isoc.org/ietfjournal`**) publishes timely reports and updates on the activities of the IETF.

Finally, the IETF *Tools Team* (**`http://www.ietf.org/tools.html`** and **`http://tools.ietf.org`**) provides many tools and applications for protocol developers. Marshall Rose and Carl Malamud take a closer look at one of these tools, namely a system for writing Internet Drafts and RFCs using XML.

Please take a moment to renew and update your subscription. You can access your subscription record by clicking on the "Subscriber Services" link at **`http://www.cisco.com/ipj`**.

—Ole J. Jacobsen, Editor and Publisher
`ole@cisco.com`

Network Authentication, Authorization, and Accounting
Part One: Concepts, Elements, and Approaches

by Sean Convery, Identity Engines

Network *Authentication, Authorization,* and *Accounting* (AAA, pronounced "triple-A") is a technology that has been in use since before the days of the Internet as we know it today. Authentication asks the question, "Who or what are you?" Authorization asks, "What are you allowed to do?" And finally, accounting wants to know, "What did you do?" These fundamental security building blocks are being used in expanded ways today. This article, the first in a two-part series, focuses on the overall concepts of AAA, defines the elements involved in AAA communications, and discusses high-level approaches to achieving specific AAA goals. Part two of the article, to be published in a future issue of IPJ, will discuss the protocols involved, specific AAA applications, and considerations for the future of AAA.

AAA, at its core, is all about enabling mobility and dynamic security. Without AAA, a network must be statically configured to control access; IP addresses must be fixed, systems cannot move, and connectivity options should be well defined. Even the earliest days of dialup access broke this static model, thereby requiring AAA. Today, the proliferation of mobile devices, diverse network consumers, and varied network access methods combine to create an environment that places greater demands on AAA.

AAA has a part to play in almost all the ways we access a network today. Emerging technologies such as *Network Access Control* (NAC) extend AAA even into corporate Ethernet access (historically the "trusted" network that set the benchmark level of security that all other types of access had to match). Today, wireless hotspots need AAA for security, partitioned networks require AAA to enforce segmentation, and remote access of every kind uses AAA to authorize remote users.

It is not clear when the term AAA first gained acceptance, but an examination of academic papers finds "authentication, authorization, and accounting" used as a discrete term (albeit without the AAA acronym) as early as 1983 in an IEEE paper[1]. Though mired in pre-Internet *Open Systems Interconnection* (OSI)-centric terminology, the ordering of the "A's" is the same as today's usage.

For most network administrators, the genesis of AAA coincided with the development of the *Remote Authentication Dial-In User Service* (RADIUS) protocol[2]. RADIUS was developed by Livingston Enterprises (now part of Alcatel-Lucent) in the early 1990s, became an Internet standard through the IETF in 1997, and today is the most widely accepted AAA protocol.

Another widely adopted AAA protocol, which predates RADIUS as an RFC by four years, is the *Terminal Access Controller Access Control System* (TACACS)[3]. Though never an Internet standard, TACACS evolved into XTACACS and then TACACS+, the latter of which is the only version of TACACS in use today.

Before we delve into the details of these protocols, it is important to understand the roles played within a AAA system.

Core Components of AAA

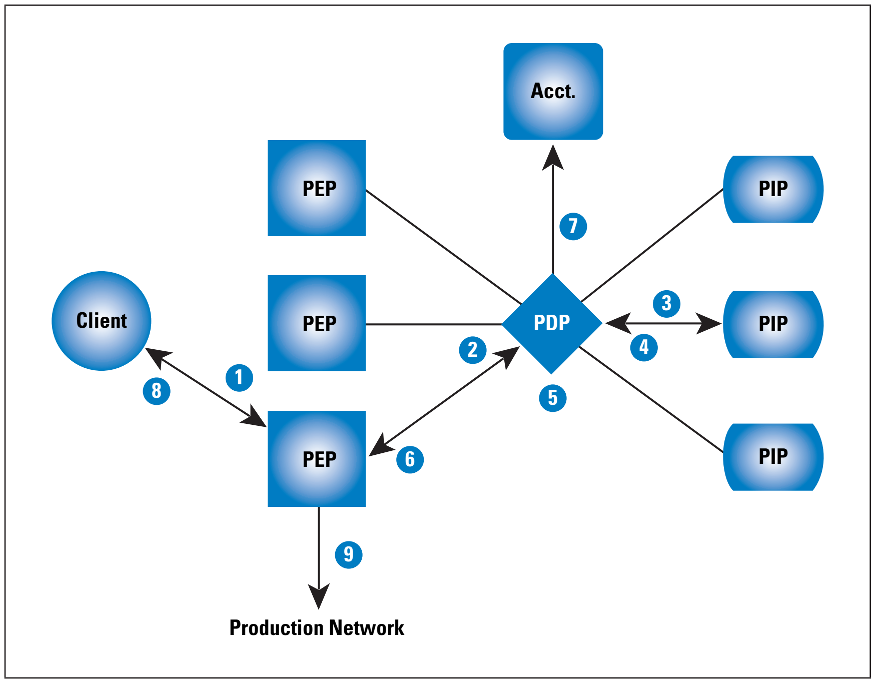
- *Client:* The client is the device attempting to access the network. The client either authenticates itself, or it acts as a proxy to authenticate the user.
- *Policy Enforcement Point (Authenticator):* The Policy Enforcement Point (PEP) is sometimes called the authenticator or *Network Access Server* (NAS). The PEP is the network device that brokers the access request for the client. The PEP can be a dial-in server, VPN concentrator, firewall, gateway *General Packet Radio Service* (GPRS) support node, Ethernet switch, wireless access point, or an inline security gateway. The PEP is responsible for enforcing the terms of a client's access. This enforcement varies based on the capabilities of the PEP and is discussed later in this article.
- *Policy Information Point:* The Policy Information Point (PIP) is a repository of information to help make the access decision. It could be a database of device IDs, a user directory such as the *Lightweight Directory Access Protocol* (LDAP), a *one-time password* (OTP) token server, or any other system that houses data relevant to a device or user access request.
- *Policy Decision Point (AAA Server):* The Policy Decision Point (PDP) is the brain of the AAA decision. It collects the access request from the client through the PEP. It also queries any relevant PIPs to gather the information it needs to make the access decision. The PDP, as its name implies, is the entity that makes the final decision around network access. It also can send specific authorizations back to the PEP that apply settings or constraints to the client's network traffic.
- *Accounting and Reporting System:* Whether on a dedicated system or built as part of a PDP, tracking use of the network with accounting is one of the best features of AAA. With all forms of network access now offering controlled access, the AAA service can tell you who got on the network, from where, and what that person was granted access to.

It is important to note that the preceding categories are logical containers of functions and not necessarily dedicated physical devices. Often elements are combined, such as PEP with PDP, and PDP with PIP.

Example AAA Flow

Now that we have examined the components of a AAA solution, walking through a typical use case will help cement our understanding of the role that each entity plays. Figure 1 shows an example of a client attempting to gain access to the network.

Figure 1: A Client Connects to a AAA-Protected Network



1. The client attempts to connect to the network, is challenged for identity information, and sends this information to the PEP. In this example, let's assume the client is a laptop with a worker attempting to access an organization's VPN from a remote location. Additionally, we'll assume this is a valid, permitted use of the network.
2. The PEP sends the collected identity information to the PDP. In some cases (discussed in part two of this article), the PEP cannot see the specific identity information provided but instead relays the information directly to the PDP.
3. The PDP queries any configured PIPs for information about the client and validates that the credential provided by the client is valid. In this example, the PIP is an LDAP directory.
4. The PIP returns a success or failure message from the credential validation step and sends additional information about the client to the PDP for evaluation. This information could include the role of the user, the home location for the user, and so on.
5. The PDP evaluates information learned about the client through the client, PEP, and PIP; the role of the PEP and PIP that serviced the request; and any contextual information (such as time of day) against its configured policies. Based on this information, the PDP makes an authorization decision.

6. The PDP sends the PEP the authentication result and any authorizations specific to the client. These authorizations trigger specific PEP actions to apply to the client. For example, the authorization data might trigger specific *Access Control Lists* (ACLs) or IP pool assignments for the client.
7. The PDP also sends the result of this transaction to the accounting system.
8. The PEP applies the authorization profile learned from the PDP and sends the "authentication successful" message to the client. The PEP can also be configured to send accounting information on this new connection to the accounting and reporting system.
9. The client accesses the production network through the PEP.

Elements of Authentication

When performing authentication, numerous elements can be evaluated before a PDP reaches its access decision. At a high level, these elements can be broken down into three categories: the principal itself (the user, device, or service requesting access), the credential the principal submits (shared key, one-time password, digital certificate, or biometric credential), and the contextual information describing the transaction (location, time of day, software state, and so on).

- *Principal:* The principal is the entity requesting authorization. It is generally some combination of user, device, or service. When concerned with a user, the PIP can provide attributes about the user such as role or group affiliations, job title, e-mail address, physical address, and so on. In specific applications, it can include much more granular information. For example, a higher-education facility might be interested in knowing a student's class schedule when servicing the student's authentication request. When the principal is a device, the same thinking applies. The PIP can inform the PDP if the device is a managed asset, what its basic usage parameters are, and so on. User and device authentication can be carried out sequentially for the same transaction, often involving device authentication first and then user authentication. Lastly, a service such as a network management process can authenticate. In this case, the service almost always looks like a user to the AAA infrastructure and is handled accordingly.
- *Credential:* The next element the PDP considers is the credential the user or device submits as proof of identity. There are four main types of credentials: shared key (password), *one-time password* (OTP), digital certificate, and biometric credential. This section examines each of these types. The first and most widely used form of credential is the shared key, typically a user password. AAA deployments that use shared keys can be subdivided based on the protocol the system uses to verify the password, including the *Password Authentication Protocol* (PAP)[4], *Challenge Handshake Authentication Protocol* (CHAP)[5], and *Microsoft CHAP Extensions* (MS-CHAP) Versions 1[6] and 2[7]. PAP authentication is a plaintext authentication method that is not recommended for use in security-sensitive environments.

However, many newer protocols provide a secure transport for PAP, making its use in AAA still quite common. Some of these methods are discussed in part two of this article. CHAP improves on the security of PAP by not sending the password in the clear but rather a challenge based on a hash of the password. MS-CHAP is a Microsoft extension to CHAP that tunes things a little bit for Microsoft environments. Version 2 of MS-CHAP addresses security weaknesses in Version 1. MS-CHAPv2 is quite common today in Microsoft environments. CHAP in all its forms is vulnerable to dictionary attacks because even if a hash cannot be decrypted, common passwords can be guessed and those hash values can be computed.

A second, also widely used credential type is the OTP. At login time, users refer to their personal token to get the OTP they will type in. The token is generally provided in hardware or software form. Tokens are designed to generate seemingly random passwords that are synchronized with a token server acting as a PIP. The OTP can be sent in the clear because it is used only once; after a configurable time (for example, 30 seconds) a new password is generated. When an OTP is combined with a *Personal Identification Number* (PIN), two-factor authentication is achieved because the client needs to have something (the token) and know something (the PIN).

The third type of credential is the *digital certificate*. Digital certificates can be stored either locally on the client or on some sort of removable device such as a smartcard. A full discussion of asymmetric-key cryptography is outside the scope of this article, but at a high level, certificates work by asserting the identity of their bearer by having the certificate signed by a trusted *Certificate Authority* (CA). CAs can be external entities such as a government or commercial enterprise or they can be internal to a given organization. The certificate itself can be freely distributed, because the only way it can be validated as belonging to the rightful owner is in combination with the private key. Because they reside on the client, certificates are most often used to authenticate a physical entity rather than an individual. However, smartcards are changing this paradigm by enabling users to take their digital certificate (and private keys) with them, thereby disassociating the certificate from the machine itself. Similar to an OTP without a PIN, a digital certificate or smartcard alone does not provide two-factor authentication. Certificate deployments, particularly smartcards, are addressing this problem by requiring a PIN to unlock access to the credential.

The fourth and least widely deployed type of credential is the *biometric credential*. Biometrics[12] ignores something you *have* and something you *know* and instead focus on something you *are*. Fingerprint scanners, iris scanners, and facial recognition are all forms of biometric authentication. Because biometrics is the newest form of credential, it is currently experiencing heightened anticipation among users regarding potential applications—and also scrutiny for potential weaknesses.

- *Contextual:* The last element the PDP typically considers in its authentication decision is the contextual information associated with the AAA request, including the network and physical location of the request, the type of access provided by the PEP, the time of day, and potentially other elements such as network load, security threat level, and so on. A relatively new entrant into this set of contextual information is client device posture, typically discussed under the rubric of *Network Access Control* (NAC). NAC or posture checks examine the software state of the client before it connects. NAC data allows the PDP to assess the degree of risk posed by the connecting client before granting the client access to the network. For example, if a system is running an out-of-date operating system, has no current security applications running, or otherwise exhibits high-risk behavior, it may not be granted access to the network. NAC will be discussed in more detail in part two of this article.

Authorization Approaches

At its core, authorization means determining what a client is allowed to do on the network. However, the granularity of this authorization is only as good as the sophistication of the PDP and the enforcement capabilities of the PEP. This section examines the authorization options for network AAA, including Layer 2 segmentation, Layer 3 filtering, and Layer 7 entitlements. It closes with an examination of some of the challenges encountered when sending or "provisioning" the authorizations from the PDP to the PEP.

- *Null Authorization (Authentication Only):* Strangely the most common authorization in AAA is no authorization at all. After the authentication event occurs, the client is immediately granted full access to the network. This characteristic is a holdover from the original goal of remote-access AAA: to perform an authentication check that simply determines whether the client should be trusted as if it were connected to the organization's home network. Because these home networks employed no segmentation or filtering within them, it was natural that remote-access techniques such as dialup and VPN would likewise employ neither. Today however, authentication is increasingly being used for all forms of network access, with a goal of providing clients with network rights commensurate with their role in the organization. This latter goal requires a strong authorization foundation through the cooperation of the PDP and PEP.

- *Layer 2 Segmentation:* For wireless access points and Ethernet switches, the most common form of authorization enforcement is Layer 2 segmentation, which works by splitting the network into multiple logical segments, isolating certain classes of client from one another. This process is most typically achieved by deploying *Virtual LANs* (VLANs), which separate the members of one VLAN from other VLANs in the same Layer 2 network—even though the VLANs traverse the same physical network infrastructure.

VLANs can be used to restrict access to specific resources by working in coordination with VLAN-specific ACLs on Layer 3 devices upstream from the Layer 2 device. For access points, a given wireless *Service Set Identifier* (SSID) can be associated with a VLAN on the wired side of the access point. *Multiprotocol Label Switching* (MPLS) is more commonly associated as a WAN transport, but there is nothing to prevent labels for traffic based on AAA. More commonly, the client is associated with a VLAN and the VLAN is associated with an MPLS label further into the infrastructure.

- *Layer 3 Filtering:* Layer 3 filtering authorizes access to resources through ACLs configured on Layer 3 devices (routers, Ethernet switches, security gateways, and so on). These ACLs (which generally encompass Layer 4 of the OSI stack as well) can enforce authorizations to a range of hosts, specific hosts, or services on those hosts. As mentioned earlier, Layer 3 filtering can be combined with Layer 2 segmentation to provide aggregate authorizations for an entire VLAN. This filtering is the most common technique on network infrastructure devices, whereas security gateways tend to apply ACLs to specific clients. Additionally, technologies such as *IP Security* (IPsec)[8] provide a Layer 3 filtering capability by allowing only certain types of traffic to travel through the VPN tunnel.
- *Layer 7 Entitlements:* Increasingly, security gateways are able to go beyond Layer 3 and 4 filtering and are starting to become application-aware, meaning that the authorizations handed from the PDP to the PEP can be very granular, focusing on the specific applications that are needed rather than broader filters based on segments or hosts on the network. Because this technology is still relatively new, there are no standards yet to make this interaction work transparently. As a result, most granular application filters are written on the PEP itself in order to allow the PDP to trigger a preexisting profile on the PEP. These sorts of provisioning challenges are discussed further in the next section.
- *Provisioning Challenges:* In AAA parlance, the term "provisioning" refers to communicating a user's session rights and constraints to the PEP so that the PEP can grant and enforce these permissions. One of the most difficult aspects of provisioning access rights on a PEP is communicating the decision of the PDP in a format the PEP can understand. This fact is one of the reasons that many PEPs come with a lightweight PDP. This approach solves the narrow problem for that PEP but creates management challenges when coordinating network AAA across a broader enterprise, because the enterprise AAA policies must be implemented individually on each unique type of PEP on the network. Because RADIUS is the most commonly used network AAA protocol, it is natural to communicate the PDP decision using that protocol. RADIUS attributes such as the "filter-id" allow the PDP to trigger a preexisting filter on the PEP.

In addition, many PEP vendors support *Vendor Specific Attributes* (VSAs) in RADIUS to enable the PDP to speak the language of the PEP more specifically. This process works well but creates a significant amount of work on the PDP to enable it to translate the policy result and correctly communicate it to each type of PEP. Another option soon to be sanctioned by the standards bodies is an extension to RADIUS that enables the sending of standard IP ACLs using RADIUS attributes[9].

One further option for provisioning is through the *Simple Network Management Protocol* (SNMP), which is typically used to assign Layer 2 ports to VLANs or to enable or disable interfaces. This process can work, but remember that the version of SNMP typically in deployment is still SNMPv2c, which is *User Datagram Protocol* (UDP)-based (connectionless) and unencrypted. Therefore, the SNMP traffic is prone to packet loss when links are congested or devices are busy, thereby requiring costly application layer retransmission schemes. It also means the transmissions themselves are vulnerable to inspection or modification. These attributes make SNMP generally a poor choice for security-sensitive tasks. RADIUS also uses UDP, but supports basic retransmission as part of the protocol.

Another provisioning method used today is standard *Secure Shell* (SSH) Protocol or HTTPS-based configuration. This method manages a device through standard administrative interfaces to set enforcement techniques. Although this method gives the PDP full access to the features of the PEP, it is very difficult to coordinate the dynamic aspects of the client AAA event with the static elements of the running configuration of the PEP. Finally, new protocols are emerging to make provisioning easier. NETCONF[10] is an *Extensible Markup Language* (XML)-based protocol designed as a replacement for network management applications connecting to devices over the *command-line interface* (CLI).

As this section has shown, there are numerous approaches to authorization in AAA. Each PEP has its own capabilities, but the challenge for a diverse network is to consistently authorize clients, regardless of the given PEP they access the network through.

Accounting Techniques

Accounting is an increasingly critical step in the overall AAA process. Regulatory controls are starting to mandate better auditing of network access. The last stage of AAA, accounting simply records which clients accessed the network, what they were granted access to, and when they disconnected from the network. Accounting has always been widely used in the *Internet Service Provider* (ISP) space because auditing network access is the basis for billing ISP customers. Increasingly, accounting is being used as a way to correlate client attribute information (username, IP address, etc.) with actions and events on the network.

This correlation can make other systems that are not user-aware more intelligent in the security decisions that they make. For example, a network *Intrusion Detection System* (IDS) can learn a lot about the behavior of a given IP address. However, when that information is correlated with the user assigned to that IP address—and the permissions that user should have—the relevance of the IDS data increases dramatically.

One of the design considerations of accounting systems is that, given the centralized nature of audit and the decentralized nature of access, they are generally out-of-band with the client's normal communications. This makes them excellent resources to refer to when the network administrator wants to know when the client connected and what the client was granted access to. However, their out-of-band nature makes them poor resources for determining what the client actually did while connected to the network. This information can be learned by the network, as mentioned earlier, by coordinating the AAA accounting information with the rest of the network enforcement and monitoring systems.

Summary and Part Two Teaser

This first part of this article introduced AAA and described many of the foundation concepts necessary to gain a sound understanding of the overall system. After defining the elements involved, a sample flow of a AAA event was described. Additionally, the high-level approaches to authentication, authorization, and accounting were discussed. Part two of this article will discuss the protocols used in AAA, including not just RADIUS, *Extensible Authentication Protocol* (EAP), TACACS+, and Diameter, but many others. Additionally, specific applications of AAA technology will be described, and some conclusions will be drawn as to what the future holds for AAA.

References

[1] Lagsford et. al., "OSI Management and Job Transfer Services," *Proceedings of the IEEE,* Volume 71, No. 12, December 1983.

[2] Rigney et. al., "Remote Authentication Dial In User Service (RADIUS)," RFC 2865 (Obsoletes RFC 2138, 2058), June 2000.

[3] Finseth C., "An Access Control Protocol, Sometimes Called TACACS," RFC 1492, July 1993.

[4] Lloyd et. al., "PPP Authentication Protocols," RFC 1334, October 1992.

[5] Simpson W., "PPP Challenge Handshake Authentication Protocol (CHAP)," RFC 1994, August 1996.

[6] Zorn et. al., "Microsoft PPP CHAP Extensions," RFC 2433, October 1998.

[7] Zorn et. al., "Microsoft PPP CHAP Extensions, Version 2," RFC 2759, January 2000.

[8] Kent et. al., "Security Architecture for the Internet Protocol," RFC 2401, November 1998.

[9] Congdon et. al., "RADIUS Filter Rule Attribute," Internet Draft, Work in Progress, January 2007, **`draft-ietf-radext-filter-08.txt`**

[10] Enns et. al., "NETCONF Configuration Protocol," RFC 4741, December 2006.

[11] Dory Leifer, "Visitor Networks," *The Internet Protocol Journal,* Volume 5, No. 3, September 2002.

[12] Edgar Danielyan, "The Lures of Biometrics," *The Internet Protocol Journal,* Volume 7, No. 1, March 2004.

SEAN CONVERY is CTO at Identity Engines, a venture-backed startup developing innovative identity management solutions for enterprise networks. Prior to Identity Engines, Sean (CCIE® no. 4232) worked for seven years at Cisco Systems, most recently in the office of the security CTO. Sean is best known as the principal architect of the SAFE Blueprint from Cisco and the author of *Network Security Architectures* (Cisco Press, 2004). Sean has presented to or consulted with thousands of enterprise customers around the world on designing secure networks. Before Cisco, Sean held various positions in IT and security consulting during his 14 years in networking. E-mail: **`sconvery@idengines.com`**

Geographic Implications of DNS Infrastructure Distribution

by Steve Gibbard, Packet Clearing House

The past several years have seen significant efforts to keep local Internet communications local in places far from the well-connected core of the Internet. Although considerable work remains to be done, Internet traffic now stays local in many places where it once would have traveled to other continents, lowering costs while improving performance and reliability. Data sent directly between users in those areas no longer leaves the region. Applications and services have become more localized as well, not only lowering costs but keeping those services available at times when the region's connectivity to the outside world has been disrupted. I discussed the need for localization in a previous paper, "Internet Mini-Cores: Local connectivity in the Internet's spur regions."[1] What follows here is a more specific look at a particular application, the *Domain Name System* (DNS).

Most Internet applications depend on the DNS, which maps human-readable domain names to the *Internet Protocol* (IP) addresses computers understand. Two Internet hosts may have connectivity to each other but be unable to communicate because no DNS server can be reached. This article examines the placement of DNS servers for root and top-level domains and the implications of that placement on the reliability of the services these servers provide in different parts of the world. It is not a "how-to" guide to the construction of DNS infrastructure and does not contain recommendations on DNS policy; it is rather a look at the placement of DNS infrastructure as currently constructed.

Although it is possible to access Internet resources without the DNS by entering numeric IP addresses directly, this type of access is not generally done. IP addresses, such as **`209.131.36.158`**, are difficult to remember, are generally unpublished outside the DNS, and often change without notice. Local caching of DNS information can mask temporary problems with DNS data for commonly accessed domain names, but caches are emptied when caching resolvers are restarted, data in caches expires, and nothing is cached until the first time it is accessed by a local user.

It should be noted that information about DNS deployment is changing rapidly. Several organizations are working on new DNS deployment. Information in this article can be considered current, to the best of my knowledge, as of May 2006.

DNS Hierarchy

The DNS is a hierarchy of domains within domains. The levels of the hierarchy are separated by dots. At the top of the hierarchy is the *root,* usually invisible but sometimes represented as a trailing dot. Using **`www.yahoo.com`** as an example, the **`com`** domain is contained within the root. **`Com`** contains **`yahoo`**, and **`yahoo`** contains **`www`**. Domains in the position **`com`** takes in this example are known as *Top-Level Domains,* or TLDs; they are the first level in the root domain. Domains in the position of **`yahoo`** are known as *Second-Level Domains*. In this example, **`www`** occupies the third level, and so forth.

The information that makes up the Domain Name System is stored on DNS *servers*. That information is divided into *zones,* which for our purposes are synonymous with domains. Each zone is stored on a set of *authoritative servers,* which are queried when users or applications attempt to access a service on the Internet. In the simplest case, a domain name query works like the following:

A *caching resolver* (so named because it caches information it receives) that has not yet cached any DNS zone data receives a query for **`www.yahoo.com`**. Because its cache is empty, it uses the *hints* distributed with the DNS software to contact one of the root servers and asks, "Where is **`www.yahoo.com`**?" The root server replies with a list of servers for **`.com`**. The caching server then asks one of the **`.com`** servers, "Where is **`www.yahoo.com`**?" and gets a response that directs it to servers for **`yahoo.com`**. It asks the same question of those servers and finally gets an answer to the question it was asking.

Generally several servers can answer questions about any domain, but if all the servers for any single level are broken or unreachable, the query fails and the service the user is looking for is unreachable. It is therefore important that the DNS be reliable, and that the servers for each zone throughout the hierarchy be reachable from anywhere the servers they point at are being used.

Root Servers

Without root servers, none of the DNS works. As of this writing, 117 root servers exist worldwide, operated by 12 different organizations.[2] Root servers are added frequently, so the number may be significantly greater by the time this article is in circulation.

Because of protocol limitations, the root servers can use only 13 IP addresses. Each root-server operator is responsible for one or two of those addresses. Using a technique called *anycast,* which allows servers in separate locations to share a *single* IP address, six of those operators operate multiple servers using the same IP address[3], meaning that only 13 of them are visible at the same time from any single location, but those 13 should in most cases include the topologically closest one.[4]

The distribution of root servers is rather uneven. North America and Europe have similar numbers: 38 in North America and 35 in Europe. The 35 in Europe are distributed fairly evenly, with the largest concentrations (four each) in London and Amsterdam, Europe's two largest Internet hubs. North America has 8 in Washington, D.C., 8 in the San Francisco Bay Area, and 5 in Los Angeles. In the United States, all cities that host root servers are on the coasts except Atlanta and Chicago. All seven of the remaining IP addresses represented by only a single server, known as *unicast roots,* are in the Washington, D.C., San Francisco, and Los Angeles areas.

Australia has two root servers in Brisbane, one in Perth, and one in Sydney. New Zealand has two, one in Wellington and one in Auckland. Singapore and the wealthier parts of East Asia are well-covered, and there are two root servers in Jakarta and one each in Bangkok and Kuala Lumpur. A year ago, there were none in the vast expanse between Bangkok and Dubai, but three have recently been added in India, along with others in Dhaka and Karachi. Mainland China and the former USSR each have two. There are three in Africa: two in Johannesburg and one in Nairobi. Another will be installed in Nairobi shortly, but most of the rest of Africa lacks direct connectivity to Johannesburg or Nairobi and must cross satellite or intercontinental fiber links to reach the nearest root servers. All four of the root servers in South America are in Brazil and Chile, with two in Sao Paulo and one each in Brasilia and Santiago de Chile.

With some exceptions, root-server density tends to correlate strongly with per-capita income. This fact is not surprising—it is true for other forms of infrastructure as well—but it means that those with the greatest dependence on external infrastructure are those least able to pay for external connectivity.

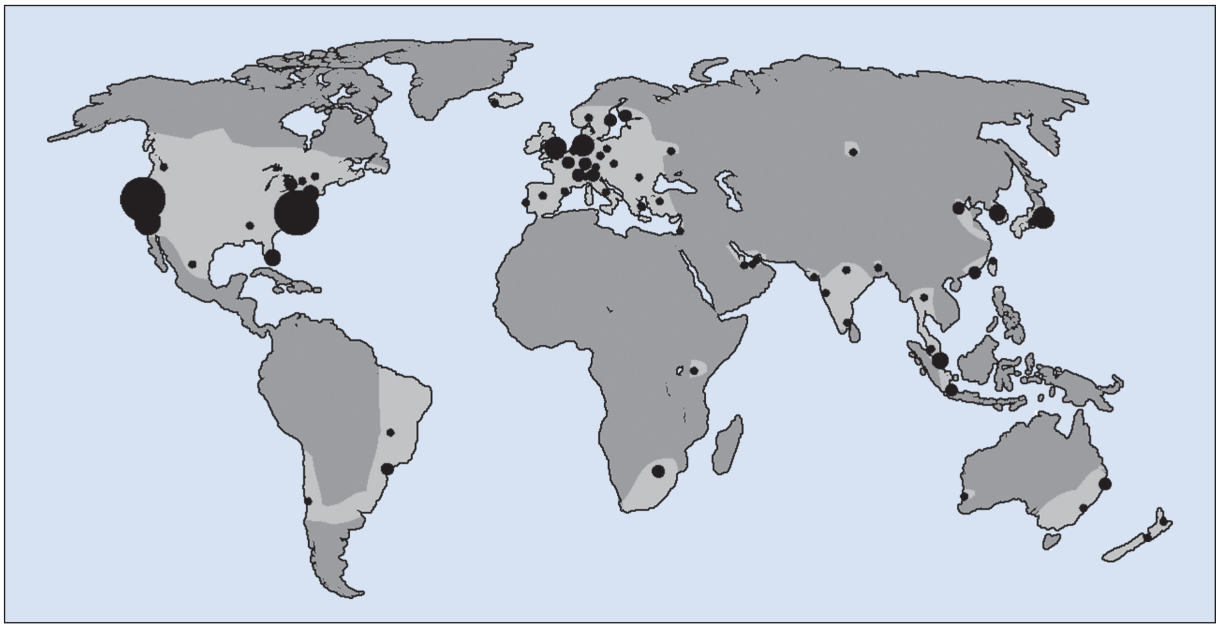
Root-Server Placement

In areas that have local root servers, finding the name servers for a top-level domain should be fairly reliable. In areas without local root servers, the ability to query the root servers is dependent on other long-distance infrastructure. In some places this infrastructure is well-developed, so this problem is not a significant one. Elsewhere long-distance infrastructure is slow, expensive, and unreliable, consisting of satellite links or a single fiber connection that may take several days to fix if it breaks.

Sri Lanka, for example, is connected to the rest of the world by a single fiber connection, which was cut in 2004 by a ship that dragged anchor in the Colombo harbor.[5] Although Sri Lanka has an exchange point that should have allowed connectivity to local Internet services, news reports said that Sri Lanka's "Internet and long-distance phone service" had been cut off. I have not received a good account of what Internet connectivity looked like from anyone in Sri Lanka at the time, but it is likely that even local Internet connections would not have worked without a local root server.

Sri Lanka is not an isolated case. The dots in Figure 1 show the locations of all root servers. The light grey areas are regions in which multiple fiber paths are available to root servers. The remainder of the world can reach root servers only by a single fiber path or by satellite. Large areas of the world are poorly covered.

Figure 1: Root Server Locations and Areas of Redundant Connectivity



Root-Server Expansion

Four of the 12 root-server operators are presently working to install root servers in areas that lack them. Although the 117 root servers currently in operation are a big improvement over the 13 that were in operation three years ago, many regions still do not have any. Those root-server operators are installing servers wherever they can get the funding to do so.

Funding is generally provided either by grants, especially from the *Asia Pacific Network Information Centre* (APNIC) in the Asia-Pacific region, by local governments or *Internet Service Provider* (ISP) associations. Because the addition of new anycast copies of root servers is relatively easy given sufficient funding, the main limitation preventing the installation of root servers in new locations is lack of funding.

One question probably best addressed in a more central manner is whether it makes sense to have many copies of one or two root-server IP addresses in some regions or whether it would be better to have more of a mix of root-server IP addresses. Currently, only 6 of the 13 root-server addresses are anycasted, only 4 are anycasted in large numbers, and 2 of those focus on specific regions, meaning that in many of the more remote parts of the world the only nearby root servers are *Internet Systems Consortium* (ISC)'s "F" and Autonomica's "I" roots, and some places have several of one of those closer than the next one of the other.

Because some DNS resolvers have their own mechanisms for finding the closest server and for handling failures of types that do not include route withdrawals, having multiple IP addresses nearby seems like a good thing. A more complex question is whether it would be worthwhile to anycast all 13 of them widely, or if there is some smaller number that would be sufficient to have nearby. Previous research on this topic has assumed a limit of 13 root servers, producing conclusions that are not applicable to the modern Internet.[6]

This article should not be seen as a criticism of the places with large numbers of root servers. Although the U.S. distribution looks strange, with the San Francisco Bay and D.C. area clusters perhaps excessive, it comes close to following the Internet topology in the United States. Indeed, the U.S. concentration may be appropriate to handle server load. Western Europe's dense but relatively even distribution of root servers through the region appears to be an optimal distribution, because most populated areas have multiple root servers nearby. Likewise, Jakarta is one of the very few cities in the developing world to have more than one, and that provides local redundancy that much of the developing world lacks. If root-server deployment were funded from a single global budget, the distribution across the world's regions would look very unfair. But because Internet infrastructure is mostly funded locally, Jakarta and Western Europe are examples other regions could emulate.

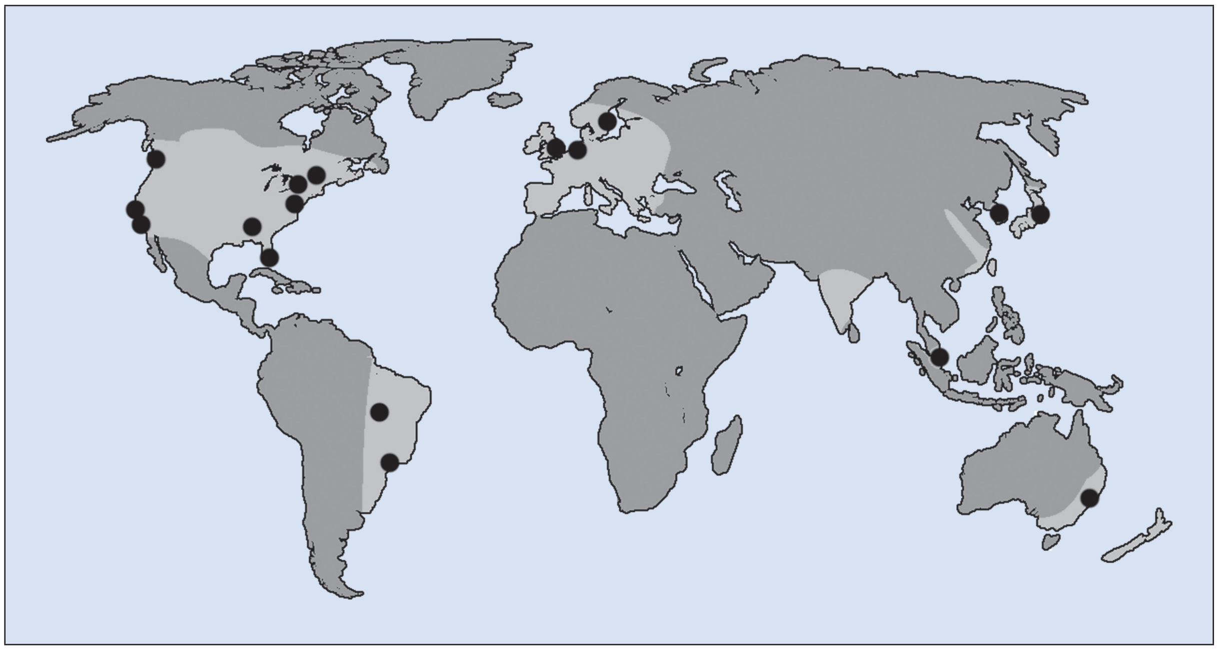
TLD Distribution

Use of the DNS also requires access to TLD servers. To access something in the **`.com`** domain, a user's local DNS resolver must be able to reach the **`.com`** servers. This statement is true for any TLD, whether it is a *generic TLD* (gTLD), such as **`.com`**, **`.net`**, and **`.org`**, or a *country code TLD* (ccTLD). Unlike the root, it is not necessary that all TLDs be reliable from all locations; if a TLD is not used to name local resources in a region, having local access to that TLD will not help if that the region gets cut off from the rest of the world.

gTLD Distribution

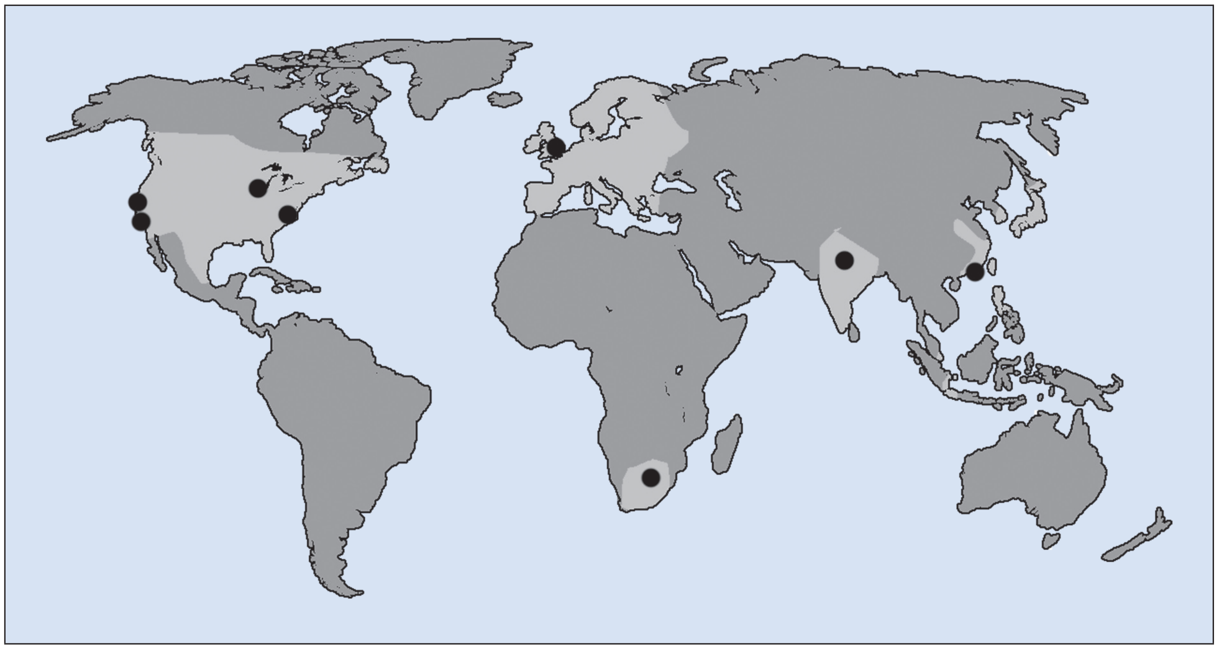
Of the gTLDs, **`.com`** is by far the largest. It is well-connected to the Internet core, the area with well-meshed internal connectivity mainly comprising North America, Western Europe, East Asia, and Singapore. (See Figure 2.) The **`.com`** servers are located in Australia, Brazil, Japan, South Korea, the Netherlands, Sweden, the United Kingdom, and the U.S. states of California, Florida, Georgia, Virginia, and Washington. The **`.com`** servers are well-connected to areas well-connected to those regions but poorly connected to Africa, South Asia, and parts of South America.

Figure 2: Server Locations for .com and .net and Areas of Redundant Connectivity



UltraDNS, the operator of **`.org`**, **`.info`**, **`.mobi`**, and **`.coop`**, among others, is also somewhat well-connected to the Internet core, although not to the extent the **`.com`** servers are. It has publicly accessible servers in four metropolitan areas in the United States as well as in London and Hong Kong. It has a couple of noncore locations, in Delhi and Johannesburg. UltraDNS also has servers in other locations, accessible only to the resolvers of certain large ISPs. Because those servers are not available to the general public in their regions, they are omitted from discussion here. (See Figure 3.)

Figure 3: Server Locations for .org, .info, and .mobi and Areas of Redundant Connectivity



Other gTLDs do not do considerably better. Table 1 shows the locations of all the gTLDs.

Table 1: Locations of TLD Servers

gTLD	Locations by Country or U.S. State
`.aero`	Switzerland, Germany, India, Hong Kong, United Kingdom, and the following states in the United States: California, Illinois, and Virginia
`.biz`	Australia, Hong Kong, Netherlands, New Zealand, Singapore, United Kingdom, and the following states in the United States: California, Florida, Georgia, New York, Virginia, and Washington
`.com`	Australia, Brazil, Canada, Japan, South Korea, Netherlands, Sweden, Singapore, United Kingdom, and the following states in the United States: California, Florida, Georgia, Virginia, and Washington
`.coop`	United Kingdom and the following states in the United States: California, Illinois, and Massachusetts
`.edu`	Netherlands, Singapore, and the following states in the United States: California, Florida, Georgia, and Virginia
`.gov`	Canada, Germany, and the following states in the United States: California, Florida, New Jersey, Pennsylvania, and Texas
`.info`	India, Hong Kong, South Africa, United Kingdom, and the following states in the United States: California, Illinois, and Virginia
`.int`	Netherlands, United Kingdom, and California in the United States
`.jobs`	Netherlands, Singapore, and the following states in the United States: California, Florida, Georgia, and Virginia
`.mil`	The following states in the United States: California, Maryland, Virginia, and other unknown locations
`.mobi`	India, Hong Kong, South Africa, United Kingdom, and the following states in the United States: California, Illinois, and Virginia
`.museum`	Sweden and California in the United States
`.name`	Singapore, United Kingdom, and the following states in the United States: California, Florida, Georgia, Virginia, and Washington
`.net`	Australia, Brazil, Canada, Japan, South Korea, Netherlands, Sweden, Singapore, United Kingdom, and the following states in the United States: California, Florida, Georgia, Virginia, and Washington
`.org`	India, Hong Kong, South Africa, United Kingdom, and the following states in the United States: California, Illinois, and Virginia
`.pro`	Canada and the following states in the United States: Illinois and Texas
`.travel`	Australia, Hong Kong, Netherlands, New Zealand, Singapore, United Kingdom, and the following states in the United States: California, Florida, Georgia, New York, Virginia, and Washington

Although gTLDs are typically marketed for their applicability to specific types of organization, or in the case of **`.com`** because it is the only domain many people have heard of, geography should also be considered in selecting domains. Most of the gTLDs have reasonable coverage throughout the Internet core region, but there are exceptions. The **`.int`** and **`.museum`** domains are hosted only in North America and Europe, and **`.pro`** is hosted only in North America.

Outside the Internet core there is little gTLD presence. Only **`.biz`**, **`.travel`**, **`.com`**, and **`.net`** are present in Australia and New Zealand. South Africa and India have **`.aero`**, **`.info`**, **`.mobi`**, and **`.org`**, making them the only gTLDs hosted in either Africa or the South Asian region. South America hosts only **`.com`** and **`.net`**, with servers in two cities in Brazil. Taken together, these are the only Southern Hemisphere gTLD locations as of this writing, and no gTLD has any presence in parts of the world without external fiber-optic connectivity, although that may be changing.

Where gTLDs should be hosted, and with what scope, are somewhat open questions. Should these domains address resources anywhere, or should their scope be local? This question is really one for the *Internet Corporation for Assigned Names and Numbers* (ICANN), or for the gTLD sponsors or registries, and beyond the scope of this article. Verisign, the company that administers **`.com`** and **`.net`**, points out that database replication with the amount of changes in the **`.com`** zone is a significant problem over slow network links.

ccTLD Distribution

Questions about where ccTLDs, the top-level domains assigned to individual countries, ought to work seem more straightforward. Working effectively in their own countries seems like the top priority, with connectivity to the Internet core and to other regions with which people in the country communicate regularly being somewhat lower priorities. Just over two-thirds of ccTLDs are hosted in their own countries; refer to Figure 4 for the bigger countries, and the online appendices for the full list. Although the third of ccTLDs not hosted in their own countries include some marketed more for international use than global use—Cocos Island's **`.cc`**, Tonga's **`.to`**, Turkmenistan's **`.tm`**, and Tuvalu's **`.tv`**, among others—those are very much the exception.

Indonesia has local access to the root and to its ccTLD (**`.id`**). Pakistan has a root server, but no local access to its ccTLD (**`.pk`**). Let's compare what happens when someone in Indonesia does a lookup on an **`.id`** domain name with what happens when someone in Pakistan does a lookup of a name in the **`.pk`** domain.

In Indonesia, the query goes to a root DNS server at the Indonesian Internet Exchange in Jakarta, where it is answered with the locations of the **`.id`** servers, several of which are also in Indonesia. The query then goes to the local **`.id`** server and is answered locally, whereupon the user can start sending traffic to the host he or she was trying to connect to, which is presumably also local. The traffic need not leave Indonesia, and if all the parties involved are in Jakarta it need not leave town.

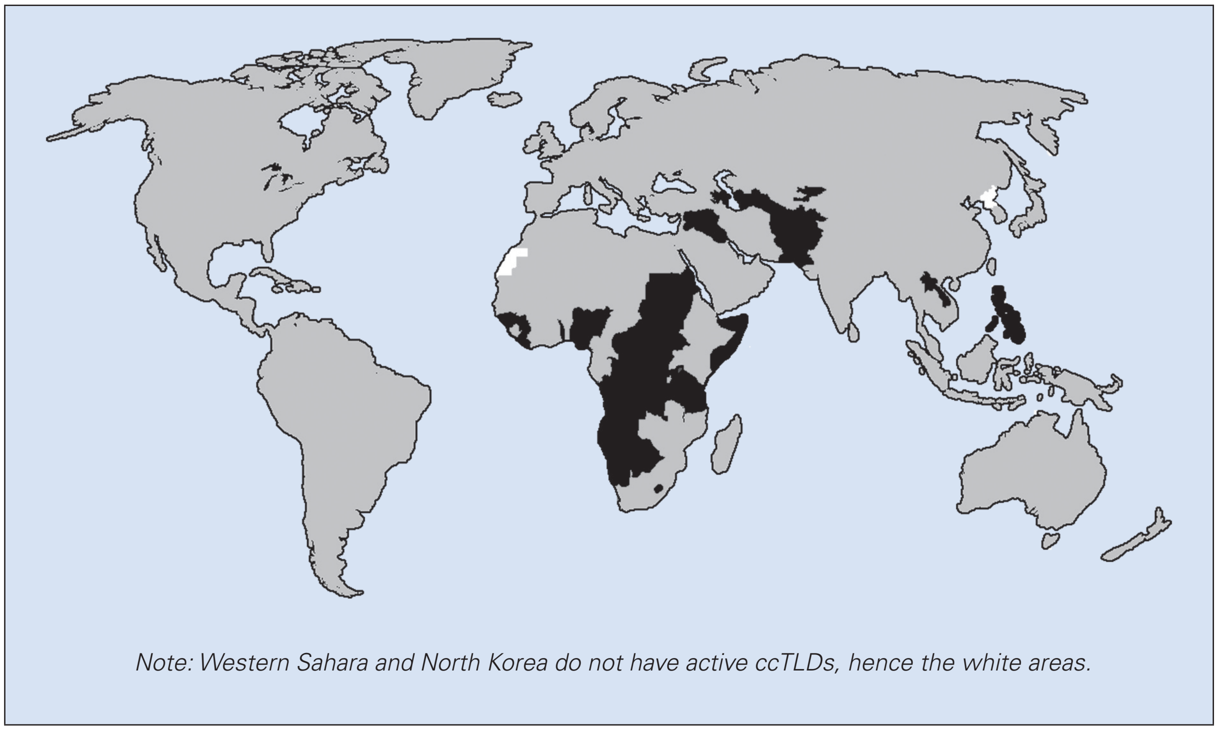
The Pakistani case is quite different. Until early 2006, there were no root servers in Pakistan, nor were there local servers for the **`.pk`** domain. There is now a root server in Karachi, but lacking servers for any TLDs it is of limited utility. DNS resolvers start out querying the local root server, but the response directs them to servers for the **`.pk`** domain, all located in the United States, at least 10 time zones away. They then send their lookup packets across the single fiber connection all the way to the United States and wait for the response. At best, this process is slow. If that fiber connection goes down, or if there is any other problem between Pakistan and these U.S. servers, local communications in Pakistan are crippled.

The situation with traditional gTLDs (**`.com`**, **`.net`**, and **`.org`**) in Indonesia and Pakistan is somewhat different. In Indonesia, local root servers provide addresses for the **`.com`**, **`.net`**, and **`.org`** servers. The **`.com`** and **`.net`** lookups can be handled in Singapore, 18 milliseconds away. Theoretically, **`.org`** lookups can be handled in Hong Kong, but *traceroutes* indicate **`.org`** queries being answered in California instead. Thus, in Indonesia, **`.id`** is hosted mostly locally, **`.com`** and **`.net`** are nearby, and **`.org`** is considerably farther away. In Pakistan, in contrast, **`.pk`** queries and **`.org`** queries are answered from the United States, more than 200 milliseconds away, while **`.com`** and **`.net`** are answered from Singapore, 80 milliseconds away. For Pakistani users of all TLDs, there are single points of failure, but **`.com`** and **`.net`** do appear to be somewhat better connected than **`.pk`**.

In Nairobi, Kenya, there are local copies of a root server and the local ccTLD (**`.ke`**). All external connectivity is by satellite, and most ISPs have only a single satellite link. Two Internet users in Nairobi wanting to communicate can do a lookup on the local root server to find the servers for **`.ke`** and can do a lookup on a local **`.ke`** server to find the servers for a subdomain of **`.ke`**. Assuming the subdomain being used is served locally, they can do a local lookup for a host within that subdomain and then send data across the local exchange point. Thus the two users in the same town can send data back and forth without having to send any data elsewhere.

According to Verisign, Nairobi will soon have servers for **`.com`** and **`.net`** as well. In contrast, to use the **`.org`** domain they can again obtain addresses of the **`.org`** server from their local root server, but the lookup of the **`.org`** domain must go over a satellite link to Europe in order to be answered by a server in London. If the satellite link is up, this process adds half a second of latency to the query. If the satellite link is down, whatever local resource they are trying to connect to is out of reach.

Figure 4: Countries that Host Their Own ccTLDs in Grey; Those that Do Not in Black


Note: Western Sahara and North Korea do not have active ccTLDs, hence the white areas.

There is also a concern about ccTLDs not served from the global core; if their region or upstream provider is cut off from the Internet outside their region, the rest of the world is unable to see that ccTLD. (See Table 2). This situation may or may not be of concern; if all Internet resources within that ccTLD become unreachable in the same outage, the DNS portion of the outage may have no additional effect. However, if there is anything in that ccTLD that is not in the ccTLD's region, or if people or systems outside prefer to get a DNS response for an unreachable IP address rather than no DNS response at all, it may be of concern. Indeed, having servers that are well-connected to "the Internet as a whole" is a recommendation of RFC 2182, though the RFC does not consider the case of large portions of the Internet not being well-connected to each other.

Table 2: TLDs Not Served in the "Internet Core" Region

TLD	Country	Location of DNS Servers
BB	Barbados	Barbados
BD	Bangladesh	Bangladesh
BH	Bahrain	Bahrain
CN	China	China
EC	Ecuador	Ecuador
GF	French Guiana	French Guiana and Guadeloupe
JM	Jamaica	Jamaica
KG	Kyrgyzstan	Kazakhstan
KW	Kuwait	Kuwait
MP	Northern Mariana Islands	Guam
MQ	Martinique	Guadeloupe and Martinique
PA	Panama	Brazil, Chile, Costa Rica, and Panama
PF	French Polynesia	French Polynesia
QA	Qatar	Qatar
SR	Suriname	Suriname
TJ	Tajikistan	Tajikistan
ZM	Zambia	South Africa and Zambia

Lack of Exchange Points and Local Peering

In the "Internet Mini-Cores" article[1], I noted that local hosting of critical infrastructure is moot if there is not either a local exchange point or a monopoly transit provider in the region. If data needed in a poorly connected region must leave the area and return to reach the user requesting it, the communication has double the latency, and possibly double the reliability problems, that it would have if it were hosted somewhere in the core. For the specific examples used in this article, I have mostly chosen areas that do have exchange points. I have not analyzed the underlying local infrastructure in all countries.

Methodology

The addresses of DNS servers for a TLD are available through several means: by looking at the root zone, by doing *digs* for the name servers, and by looking in the *Internet Assigned Numbers Authority* (IANA) *whois* data, among others. I did lookups against an anycast root server on my own network, because that seemed easiest to automate. My script then did a lookup for the address of each name server, stripped off the last octet, and produced a list of TLDs hosted in each /24 subnet.

There are 635 /24s containing name servers for TLDs; 142 of them host multiple TLDs; the rest host just one. I assumed that all DNS servers in a given /24 were likely to be in the same or nearby locations. This situation appears not to be the case for the UUNet name servers, and there are probably a few other exceptions that will show up as errors in my data.

I looked at a few automated geolocation systems to attempt to attach locations to the DNS servers, but none of them appeared to be producing accurate information. Instead, I guessed at the locations of the 600 subnets, using *traceroutes* from a variety of locations, paying attention to DNS, latency, and the results of whois queries for address space along the way. I also asked lots of questions of DNS operators and others and am particularly grateful to several anycast DNS operators, whose locations would not have all been found by my *traceroutes*. Some of my guesses are likely incorrect, and corrections are appreciated.

I may be missing some information about the UltraDNS TLD servers, because UltraDNS has locations it regards as confidential. This information about UltraDNS servers is from Afilias's **`.net`** application, *traceroutes* from a variety of locations, and UltraDNS.[7]

Locations of root servers are easier to find; they are listed at **`http://www.root-servers.org`**. Some supplemental information about **`j.root-servers.net`** was supplied by Verisign. If there are operational root servers not included on **`www.root-servers.org`** other than the J-roots, I did not count them.

The full lists of locations of all TLDs and TLD servers are in the appendices to this article, at:
`http://www.pch.net/resources/papers/infrastructure-distribution/dns-distribution-appendices.pdf`.

References

[1] Steve Gibbard, "Internet Mini-Cores: Local connectivity in the Internet's spur regions" (2005):
`http://www.pch.net/resources/papers/Gibbard-mini-cores.pdf`

[2] Root Server Technical Operations Association:
`http://www.root-servers.org`

[3] Joe Abley, "Hierarchical anycast for global service distribution," ISC Tech Notes (2003):
`http://www.isc.org/index.pl?/pubs/tn/index.pl?tn=isc-tn-2003-1.html`

[4] Bradley Huffaker, "Two days in the life of three DNS root servers" (2006):
`http://www.caida.org/publications/presentations/2006/brad_wide0611_anycast_analysis`

[5] Tim Richardson, "Ship's anchor cuts cable to Sri Lanka," *The Register,* August 24, 2004:
`http://www.theregister.co.uk/2004/08/24/sri_lanka_anchor`

[6] Tony Lee, Bradley Huffaker, Marina Fomenkov, and kc claffy, "On the problem of optimization of DNS root servers' placement" (2003):
`http://www.caida.org/publications/papers/2003/dns-placement/`

[7] Afilias, ".NET Application Form":
`http://www.icann.org/tlds/net-rfp/applications/afilias.htm`

STEVE GIBBARD is a Network Architect for the nonprofit organization, Packet Clearing House (**`www.pch.net`**), based in Berkeley, California. He runs an anycast DNS network that hosts the top-level domains for several countries and several of the "I" root anycast DNS servers, maintains PCH's network of route collectors and route servers at exchange points around the world, and researches the interconnection of Internet networks. In addition, Steve carries out network architecture and peering work as a consultant for several ISPs in the San Francisco Bay Area and elsewhere. Steve is a former Senior Network Engineer at Cable & Wireless, and has held network engineering positions at Digital Island and World Wide Net. E-mail: **`scg@pch.net`**

Writing Internet Drafts and RFCs Using XML

by Marshall T. Rose, Dover Beach Consulting, Inc. and Carl Malamud, Public Resource, Inc.

What is the work product of the *Internet Engineering Task Force* (IETF)? Some cynical observers might suggest "many fine lunches or dinners," but we argue that those niceties are merely the means to an end. The goal of the IETF is to provide open standards for the Internet community, and those standards are memorialized as written documents called *Request For Comments* (RFCs).

In general, two organizations control the publication of documents as RFCs:

- The *Internet Engineering Steering Group* (IESG) determines which documents are suitable for publication as RFCs—typically by chartering working groups, reviewing their progress (through reading the work-in-progress *Internet Drafts*)—and ultimately approving their documents for publication.
- The RFC Editor strives for "quality, clarity, and consistency of style and format," and has developed a particular editorial style. The latest RFC that documents this style, RFC 2223[1], is about a decade old. A somewhat more current version can be found in a text file maintained by the RFC Editor.[4]

For a more detailed discussion of the interaction between these two organizations, consult RFC 3932[2].

As an organization, the IETF excels at "eating its own dog food," including its work product: just as a protocol specification describes interactions on the wire but does not dictate the programming language used for implementation, so too, the IETF has not really cared which document preparation tools are used. The IESG worries about technical quality, and the RFC Editor worries about stylistic consistency (and, to be fair, technical quality as well). This policy works because of the careful choices made by the early Internet community, and in particular the RFC Editor, with respect to the "final form" footprint of the documents. (A discussion of these design decisions is far beyond the scope of this short article—for now, we note that it is hard to argue with success.)

An unfortunate side effect of this focus on stylistic consistency is that, for many years, the RFC Editor has had to recode documents for consistent formatting. Internally, the RFC Editor used *nroff*[5] for this purpose, and sophisticated authors wishing to minimize RFC Editor "downtime" tended to use the same *nroff* boilerplate. The *nroff* text-formatting program has many strengths, but it can also be fairly viewed as a textual "assembly language," with the result that authors spent a lot of time dealing with low-level formatting concerns.

In some limited cases, the high degree of formatting-specific expertise is warranted, but for the vast majority of documents, the high entry cost is not.

From Assembly Language to Markup

In early 1999 we were working at a startup company, and we needed a way to organize, search, and retrieve information from documents. We decided to use a markup language for this purpose. We also decided to use the RFC series as one of the testing grounds for the technology, because this series was one we were familiar with. Although today everyone knows what the *Extensible Markup Language* (XML) is, then there were only two widely known markup languages for authoring: SGML and HTML.

The "SG" in SGML is an abbreviation for *Standard Generalized* and not *Simple Generic*. SGML is used for the formatting of a great many books; further, it is used in large projects with long lifetimes. Although truly excellent from an "enumerate every possibility" standpoint, it has a very high cost of entry, making it difficult to use for anything other than specialized applications.

In contrast, the *Hypertext Markup Language* (HTML) embodies elegance of design, but (in the absence of *Cascading Style Sheets* [CSS]), is a presentation language, not unlike *nroff* in many respects. In other words, we needed something with the structural richness of SGML and the elegant simplicity of HTML. The newly invented XML seemed to meet the requirements.

This process led us to develop a language based on XML, which captured high-level RFC constructs (for example, authorship information) and largely ignored presentational concerns. The result is called the *2629 format*[3] (also known as the "xml2rfc format," named after the initial processor for this language).

The Advantages of Markup

To understand the advantages of this approach, let's look at one example: references. Like most archival series, the RFC Editor has a very rigorous, yet unstructured, syntax for citations. Although this consistency is good for readers of RFCs, achieving consistency of references using tools such as *nroff* was often the hardest part of creating a new document. With the 2629 format, the **`<reference>`** element contains a small number of subordinate elements that capture all the semantics of the reference. The XML processor takes this information and produces a properly formatted document.

Further, because this information is structured, it is possible to develop automated bibliographic databases for a wide range of data sources. In fact, using the XML "include" mechanism, a document author usually includes just a pointer to the reference, and lets the processor do all the complicated work.

A second advantage is that processors can produce different kinds of output. Some people prefer to view their documents in HTML rather than the canonical textual format. Julian Reschke has written a library of XSLT files that convert to various HTML formats (Strict, Transitional, XHTML, and so on). For example, references are hyperlinked in line, allowing for easy traversal of citations. Still others prefer the *Portable Document Format* (PDF) for printing. By using one of Julian's XSLT scripts and the truly excellent Prince[6] XML/CSS processor, the result is high-quality, printer-ready output.

However, the primary advantage is that the "high-level" approach allows the author to focus more on content and less on format: a processor can enforce the vast majority of the esoterica associated with the RFC Editorial style, including:

- Inserting required boilerplate (and in particular, the desired revision of the boilerplate)
- Checking for mandatory sections such as "Security Considerations" or "Normative References"
- Generating a specialized table of contents, etc.

To Infinity and Beyond

After publishing RFC 2629, an unexpected result occurred: people outside the IETF started using the 2629 format for their projects. Most credit for this side effect goes to the universality of the canonical textual format. However, some authors are using the 2629 format when writing books (they convert the 2629 format to SGML, which is sent to the publisher), business plans, and software documentation—and even to create a new series of non-IETF technical documents. The constituency here seems to revolve around having a simple yet structured way to author documents.

For the last few years, a large number of XML editing programs have been deployed, and many of these support the 2629 format. These editors offer two advantages: first, they provide a natural paradigm for editing nested content; and, second, sophisticated editors can be integrated into an automated work flow. (Having said that, the authors still use *Emacs* and *vi* for their XML editors.)

A good example of the use of XML editors is a "plug-in" for the *XMLMind* Editor[7]. This plug-in, written by Bill Fenner, provides a variety of services to the author, such as graphical editing of sections, templates for common constructs, and validation of references.

Over the last 10 years, the 2629 format has evolved in true IETF fashion, based on running code and a rough consensus. Originally created by the authors for our own convenience, we have been more than pleased to see this format used first by an informal community of developers and writers, and more recently by the IETF secretariat, tools team, and administrative entity and by the RFC Editor.

Today, many people use a common high-level markup language for writing RFCs. The next step in this natural evolution will be making the repository of XML-tagged RFCs available to those involved in document distribution, so that RFC repositories will be able to take advantage of the meta-data in the creation of search engine, alternative formats, and any other value-added constructs that would be of use to the community. (At present the RFC Editor prefers input in the 2629 format, but ultimately runs a processor that generates *nroff* for "tweaking"—in the near future, we hope that the xml2rfc textual output can be tuned to avoid this final step.)

To find out more, go to the xml2rfc Website[8] or visit the official directory of IETF authoring tools[9].

References

[1] Postel, J. and J. Reynolds, "Instructions to RFC Authors," RFC 2223, October 1997.

[2] Alvestrand, H., "The IESG and RFC Editor Documents: Procedures," BCP 92, RFC 3932, October 2004.

[3] Rose, M., "Writing I-Ds and RFCs using XML," RFC 2629, June 1999.

[4] Reynolds, J. and R. Braden, "Instructions to Request for Comments (RFC) Authors," August 2004. **ftp://ftp.rfc-editor.org/in-notes/rfc-editor/instructions2authors.txt**

[5] Ossanna, J., "Nroff/Troff User's Manual," UNIX Programmer's Manual – Volume 2 (Bell Laboratories), 1979. **http://en.wikipedia.org/wiki/Nroff**

[6] **http://princexml.com/**

[7] **http://www.xmlmind.com/xmleditor/**

[8] **http://xml.resource.org**

[9] **http://tools.ietf.org/inventory/author-tools.shtml**

CARL MALAMUD is the co-founder of Public Resource, a nonprofit public-benefit engineering firm. *Exploring the Internet* was published in 1992 as a book, but today would be called a blog. "Geek of the Week" was published in 1993 as an audio file available for download with FTP, but today would be called a podcast.
E-mail: **carl@media.org**

MARSHALL T. ROSE is Principal of Dover Beach Consulting, Inc. He has authored 9 books, 74 RFCs, and 4 patents. With respect to his work on the 2629 format, he claims "self defense." E-mail: **mrose@dbc.mtview.ca.us**

Fragments

ICANN Board Rejects .xxx Domain Application

On March 30th, 2007 the Board of the *Internet Corporation for Assigned Names and Numbers* (ICANN) voted to reject the **`.xxx`** *sponsored Top Level Domain* (sTLD) application from ICM Registry, Inc.

"This decision was the result of very careful scrutiny and consideration of all the arguments. That consideration has led a majority of the Board to believe that the proposal should be rejected," said Dr Vint Cerf, Chairman of ICANN. "I thank my fellow Board members and the community for their input," Dr Cerf said.

A copy of the resolution from the Board meeting is available at:

`http://www.icann.org/minutes/minutes/resolutions-30mar07.htm`

A transcript of the Board meeting is also available at:

`http://icann.org/meetings/lisbon/transcript-board-30mar07.htm`

ISOC Fellowship to the IETF

The *Internet Engineering Task Force* (IETF) is the world's premier Internet standards setting organization. It operates as a large, open international community of network designers, operators, vendor experts, researchers, and other interested technologists. While much of the IETF's work takes place over mailing lists, the in-person experience promotes a stronger understanding of the standardization process, encourages active involvement in IETF work, and facilitates personal networking with others that have similar technical interests.

Presently, there is limited participation at the IETF by technologists from developing countries. There are, however, many talented individuals in developing regions that have an interest in and follow IETF work and would benefit from the opportunities that attending an IETF meeting presents. As such, the main purposes of the Internet Society (ISOC)'s *IETF Fellowship Program* are to:

- Raise global awareness about the IETF and its work
- Foster greater understanding of and participation in the work of the IETF by technologists from the developing world
- Provide an opportunity for networking with individuals from around the world with similar technical interests
- Identify and foster potential future leaders from developing regions
- Demonstrate the Internet community's commitment to fostering greater global participation in Internet Forums such as the IETF

ISOC successfully piloted the IETF Fellowship program at the 66th IETF meeting in Montreal in June 2006. Two individuals from Africa participated in this first pilot. Three individuals from the Pacific and Latin America participated in a second pilot phase at the 67th IETF meeting in San Diego in November 2006. All found the experience highly beneficial. Based on the success of the pilots, ISOC decided to formalize the program beginning in 2007.

The ISOC Fellowship pays for the Fellow's IETF meeting registration and social event fees, a round-trip economy class airfare to the meeting, hotel accommodation, and a small stipend to offset incidental expenses.

The program provides fellowships for up to five individuals per IETF meeting. ISOC will be putting out a call for candidates, including through ISOC chapters, at least 3 months before an IETF meeting. A small selection committee comprised of individuals knowledgeable about the IETF will evaluate the applicants against selection criteria and make their fellowship recommendations.

Fellowship recipients will have an obligation to present or otherwise share their experiences at the IETF meeting they attend with their local community and to provide feedback on their experience to ISOC so that the program can be continuously improved. An *ISOC Fellowship Alumni Network* will be established to extend the fellows IETF experience and relationship-building opportunities after the meeting.

For further information on the specifics of the program and how to apply for an ISOC Fellowship see:

`http://www.isoc.org/educpillar/fellowship/application.shtml`

BGP: The Movie

Statistics on Internet resources have been animated to provide a high-level overview of the consumption and use of IPv4 addresses and AS numbers since 1983. The animated video also clearly shows the effect of *Classless Interdomain Routing* (CIDR) and *Regional Internet Registries* (RIR) allocation policies on consumption rates and routing. This animation was developed by *Asia Pacific Network Information Centre* (APNIC) staff members, Geoff Huston and George Michaelson. You can download the 58MB movie from:

`http://www.apnic.net/news/hot-topics/docs/bgp-movie.mpg`

Internet Governance Articles and References

APNIC is also maintaining a collection of articles and references on Internet governance to help the community understand the issues and stay abreast of developments. You can find these at:

`http://www.apnic.net/news/hot-topics/internet-gov/index.html`

Call for Papers

The Internet Protocol Journal (IPJ) is published quarterly by Cisco Systems. The journal 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. The journal carries tutorial articles ("What is...?"), as well as implementation/operation articles ("How to..."). It provides readers with technology and standardization updates for all levels of the protocol stack and serves as a forum for discussion of all aspects of internetworking.

Topics include, but are not limited to:

- Access and infrastructure technologies such as: ISDN, Gigabit Ethernet, SONET, ATM, xDSL, cable, fiber optics, satellite, wireless, and dial systems
- 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, network 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, and regulatory topics such as: copyright, content control, content liability, settlement charges, "modem tax," and trademark disputes in the context of internetworking

In addition to feature-length articles, IPJ will contain standardization updates, overviews of leading and bleeding-edge technologies, book reviews, announcements, opinion columns, and letters to the Editor.

Cisco will pay a stipend of US$1000 for published, feature-length articles. Author guidelines are available from Ole Jacobsen, the Editor and Publisher of IPJ, reachable via e-mail at **`ole@cisco.com`**

The Internet Protocol Journal is published quarterly by the Chief Technology Office, Cisco Systems, Inc.
www.cisco.com
Tel: +1 408 526-4000
E-mail: ipj@cisco.com

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive, M/S SJ-7/3
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

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