The
Internet Protocol Journal

December 2009 Volume 12, Number 4

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

You can download IPJ back issues and find subscription information at:
www.cisco.com/ipj

ISSN 1944-1134

From The Editor

In our last issue we brought you Part 1 of a two-part article on *Cloud Computing*. T. Sridhar introduced various aspects of cloud computing, including the rationale, underlying models, and infrastructures. Part 2, subtitled "Infrastructure and Implementation Topics," is included in the current issue. Cloud computing has received a great deal of press in recent months and continues to be an area of rapid development. I'm confident that we will have more articles about this topic in future editions of IPJ.

With this issue we start a new series of articles under the general heading "Protocol Basics." The idea is to present a series of in-depth tutorials on numerous protocols that are used every day on the Internet and in enterprise networks. The articles will cover protocol details as well as implementation, deployment, and usage scenarios. In some cases the articles will also summarize the "lessons learned" and present "best-practice" guidelines. To start the series, we asked Bill Stallings to give us an overview of the *Secure Shell* (SSH) Protocol. We invite you to send us suggestions for other protocols that you'd like to see covered in this series.

Today's Internet is a result of many years of technological development and innovative uses of the resulting infrastructure. Of equal importance has been many *policy* choices made over the years, ranging from what protocols to use to how to allocate finite resources such as the IPv4 address space. A new book, *Protocol Politics: The Globalization of Internet Governance*, explores some of this history. The book is examined in an extended review by Tom Vest.

Let me remind you that we will no longer be automatically extending your subscription when it expires. Please take a moment to check your expiration date (printed on the back of the journal for subscribers in the United States, and on the envelope for our international subscribers). Visit the "Subscriber Services" section of our webpage at **www.cisco.com/ipj** to update or renew your subscription. You can also contact us by e-mail to **ipj@cisco.com** regarding any aspect of your subscription.

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

Cloud Computing—A Primer
Part 2: Infrastructure and Implementation Topics

by T. Sridhar

Cloud computing is an emerging area that affects IT infrastructure, network services, and applications. In Part 1[0] of this two-part article, we introduced various aspects of cloud computing, including the rationale, underlying models, and infrastructures. In Part 2 we discuss specific infrastructure aspects of cloud computing in detail, specifically:

- Network Infrastructure
- Cloud-to-Cloud and Federation Considerations
- Security

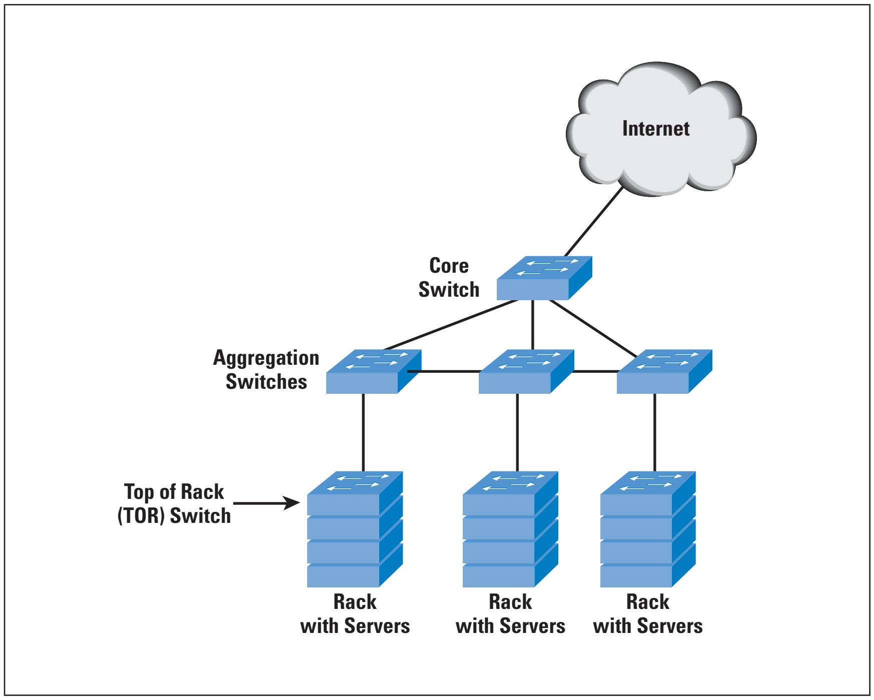
In addition, we will provide some perspective on select topics in cloud computing that have garnered interest. Remember that cloud computing is an emerging area where approaches to some of these topics are still evolving. In addition, although cloud computing is not intrinsically dependent upon virtualization, there is common agreement that virtualization (specifically, server virtualization) will be an integral part of cloud-computing solutions of the future. Consider the discussion in the following sections in this context.

Network Infrastructure

In a limited sense, the cloud can be treated as a large data center run by an external entity providing the capability for elasticity, on-demand resources, and per-usage billing. Data-center architecture often follows the common three-layer network topology of access, aggregation, and core networks with enabling networking elements (switches and routers). Consider the topology shown in Figure 4 of Part 1, reproduced here as Figure 0. The servers can be connected through a 1-Gbps link to a *Top of Rack* (TOR) switch, which in turn is connected through one or more 10-Gbps links to an aggregation *End of Row* (EOR) switch. The EOR switch is used for interserver connectivity across racks. The aggregation switches themselves are connected to core switches for connectivity outside the data center.

From a functional perspective, data-center server organization has often adopted a three-tier architecture (a specific case of an N-tier architecture). The three-tier functional architecture has a web or *Presentation Tier* on the front end, an *Application Tier* to perform the application and business-processing logic, and finally a *Database Tier* (to run the database management system), which is accessed by the Application Tier for its tasks (refer to Figure 1 on page 4).

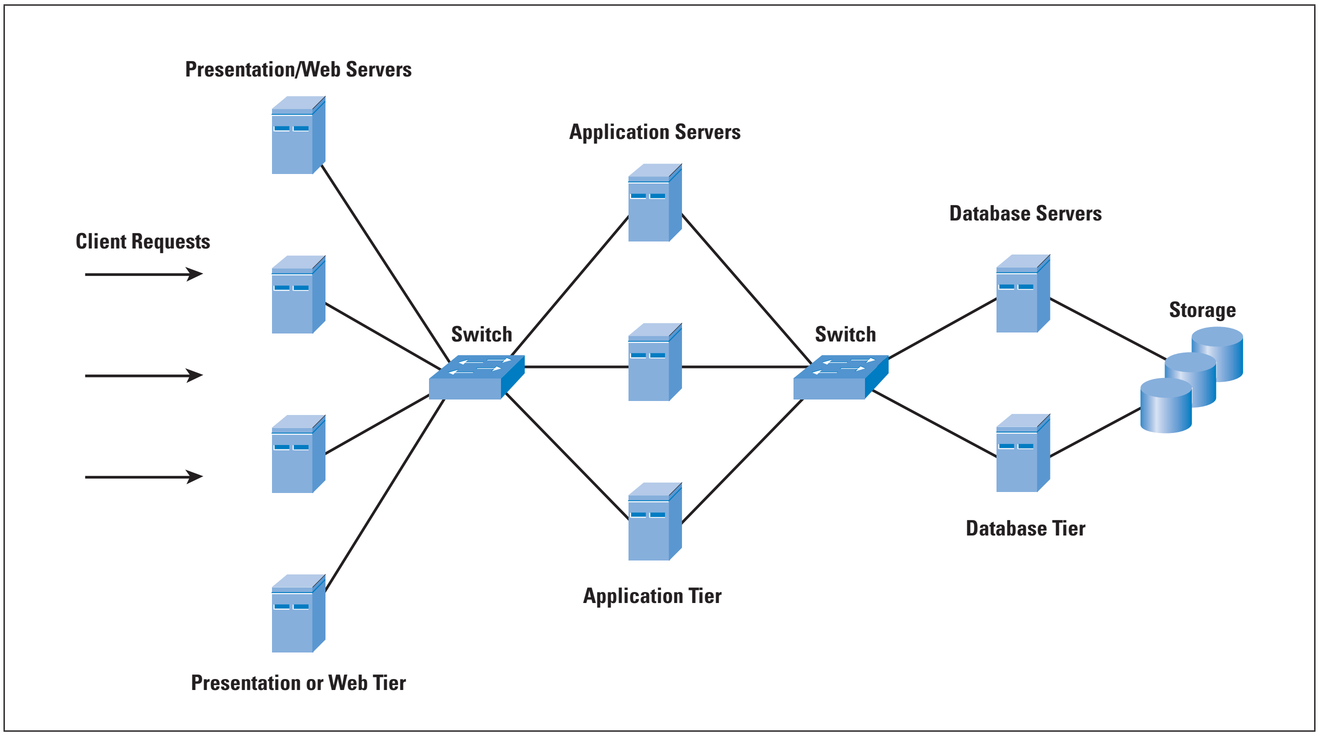
Figure 0: Example Data-Center Switch Network Architecture (from Part 1)



Although it is not necessary for each tier to be represented by its own physical servers (for example, you could have the Application and Database functions mapped into a single physical server), it is a common representation. The reason for this multitiered design is to control the connections and interactions, as well as for scaling and security. It is not uncommon for the Presentation Tier to be in a *Demilitarized Zone* (DMZ) while the other tiers are located deep inside the data center. Although all tiers could connect to storage for performing their functions, the Database Tier is the one with the maximum storage bandwidth requirements.

It follows that the server connectivity and the network topology for the cloud data centers might follow a similar organization. If you are an enterprise, you can perform the same business functions as before, but by using the external cloud. The choice of servers, software loads, and their interconnection will depend upon what you need to accomplish. In the following sections, we discuss how this design is handled in *Infrastructure as a Service* (IaaS), *Platform as a Service* (PaaS), and *Software as a Service* (SaaS).

Figure 1: Three-Tier Functional Server Architecture



Data-Center Infrastructure Extension – IaaS

If the cloud is thus seen as an extension of the existing data center, IaaS as outlined in Part 1 is a natural fit. Here, you would specify the number of servers in each tier, load the appropriate server image (web, business logic, or database manager), and "connect" them (through a menu or *Application Programming Interface* [API] provided by the IaaS provider) by specifying the links between them. You can also specify the network connectivity at this time (more on this later). For an enterprise IT administrator, this model provides the greatest degree of control and, to an extent, a familiar operating topology. The cloud provider handles the elasticity by ensuring that the number of servers and switches is adequate for you to configure and connect in the specified topology. Per-use billing and on-demand resource addition and removal are also provided by the cloud provider. Note that if you have complete control, you also are responsible for security, application usage, and resource management.

PaaS and SaaS Infrastructure

In the case of PaaS, you transfer more control to your cloud service provider. The platform used to build the service you require can scale transparently without any of your involvement other than at the time of configuration. You do not need to understand the tier connectivity, bandwidth requirements, or how it all functions under the hood.

Cloud service providers can realize this function—often with a three-tier topology similar to that for traditional data centers. However, some of them have innovated to perform parts of the function differently. For example, the database functions may rely upon a model of *scaling out* (splitting the database across multiple servers) instead of *scaling up* (increasing the capability of the machine running the database servers). Their claim is that with clouds involving large amounts of data that you can partition and work on, it is easier to scale out than scale up. According to some cloud service providers, traditional relational databases are not suitable candidates for scale-out. Hence, some cloud vendors have provided their own database models and implementations—a common one being the type known as the *Key-Value database*.

SaaS vendors have the highest degree of control among the three models. The realization of the network topology can be similar to existing data centers and scale up or down according to the number of users that are added. However, because they offer a specific set of applications to the cloud users, their server and network topology is quite straightforward.

For the following discussions, we will use IaaS as the representative cloud service model, with a primary consideration being "cloud bursting"—how an existing IT infrastructure can take advantage of the power of the cloud when it needs additional resources. Note that some of the discussion might also be relevant for internal clouds. In addition, we will assume a virtualized server infrastructure for the IaaS cloud because this infrastructure provides a greater degree of flexibility for cloud service providers (Amazon being a key example).

Virtualization and Its Demands on Switching

In Part 1, we provided the context for a virtual switch within a physical server containing multiple virtual machines. There are some addressing and control factors to consider in this model. Consider a data center with 100 servers, each with 16 virtual machines but with one physical 10-Gbps Ethernet connection to the external switch from each physical machine. If we were to carry forward the model where each physical server is replaced with its virtual equivalent but still needs to be addressable (through a *Media Access Control* [MAC] layer address and an IP address), you would need 16 MAC and IP addresses for the virtual servers that now reside "on top" of the single physical link, for a total of 1600 addresses across all servers. This problem is exacerbated when you increase the number of VMs per server. Switching between MAC addresses belonging to the virtual machines is done by the virtual switch inside the server.

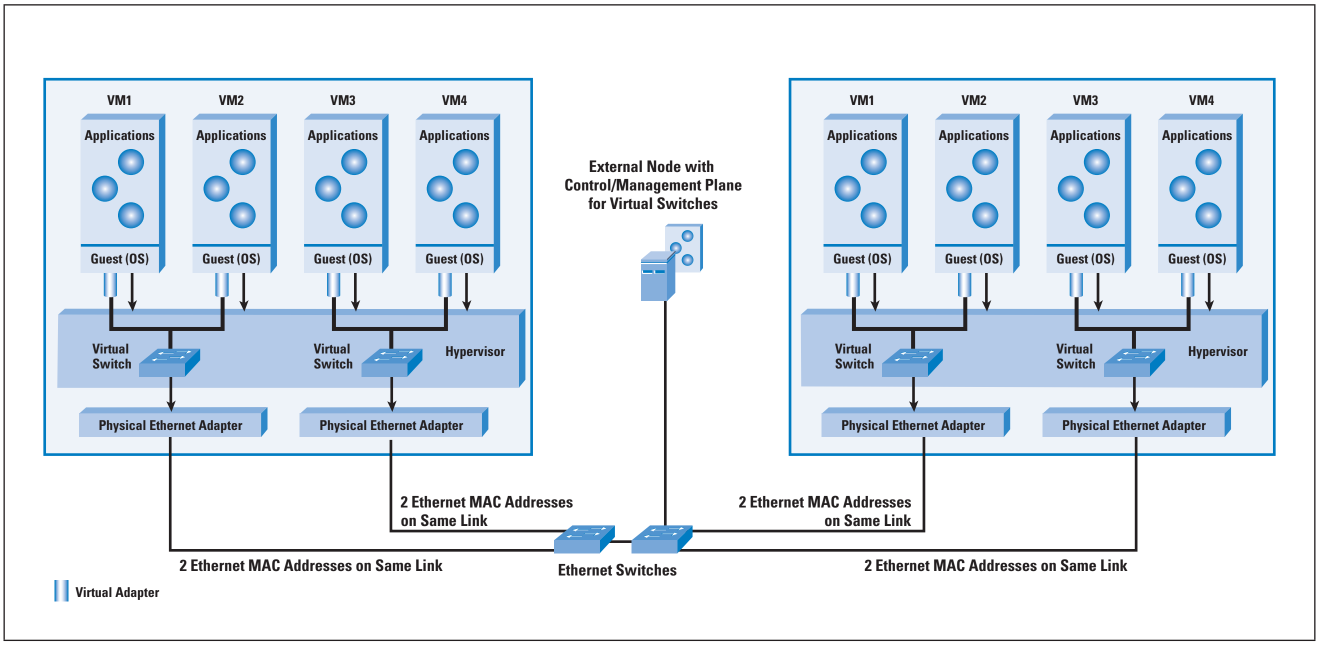
Consider the topology in Figure 2. The virtual switch treats the physical link as an uplink to the external physical switch. This intramachine *Virtual Machine* (VM) switch with an uplink to the external switch is completely in line with access and aggregation switch topologies where the access layer is subsumed inside the server. Note that each physical host can have more than one virtual switch to support greater logical segmentation. In such cases, it is common for each of the virtual switches to have its own physical uplink to the external Ethernet switch.

The virtual switch does not need to learn MAC addresses like a traditional switch—it assumes that all destination-unknown frames should be forwarded over the physical link (or uplink to the physical switch). In addition, it switches traffic between the intramachine VMs according to policy. For example, you could prohibit two VMs on the same machine from communicating with each other by configuring an access control list on the virtual switch. The VMs may all be on the same or on different VLANs. Broadcasts and intra-VLAN traffic are forwarded according to the rules for each VLAN. In effect, the virtual switch is a simple function that is used for aggregation and access control within a physical server containing VMs.

Management of these virtual switches can follow an aggregation model—where multiple virtual switches are managed through an external node (physical machine or VM), as shown in Figure 2. This external node provides the management view on behalf of the switches. Often, the external node can run control-plane protocols for Layer 2/3 functions, in effect appearing like a control or management plane with multiple data-plane instances (the virtual switches). When VMs need to be migrated to other physical servers, this separation of control- or management-plane functions permits easier migration of policy and access lists.

Virtual switches do have some disadvantages. Inter-VM traffic within the same machine is not visible to the network and cannot be subject to appropriate monitoring by network administrators. The IEEE is discussing approaches to providing external network switches the visibility into the intra-VM traffic. The options include "hair pinning," where inter-VM traffic would still be carried over to an external switch and brought back to the same physical server.

Figure 2: Virtual Switch Aggregation and Management by External Node


VM1
VM2
VM3
VM4
Applications
Guest (OS)
Virtual Switch
Hypervisor
Physical Ethernet Adapter
External Node with Control/Management Plane for Virtual Switches
2 Ethernet MAC Addresses on Same Link
Ethernet Switches
Virtual Adapter

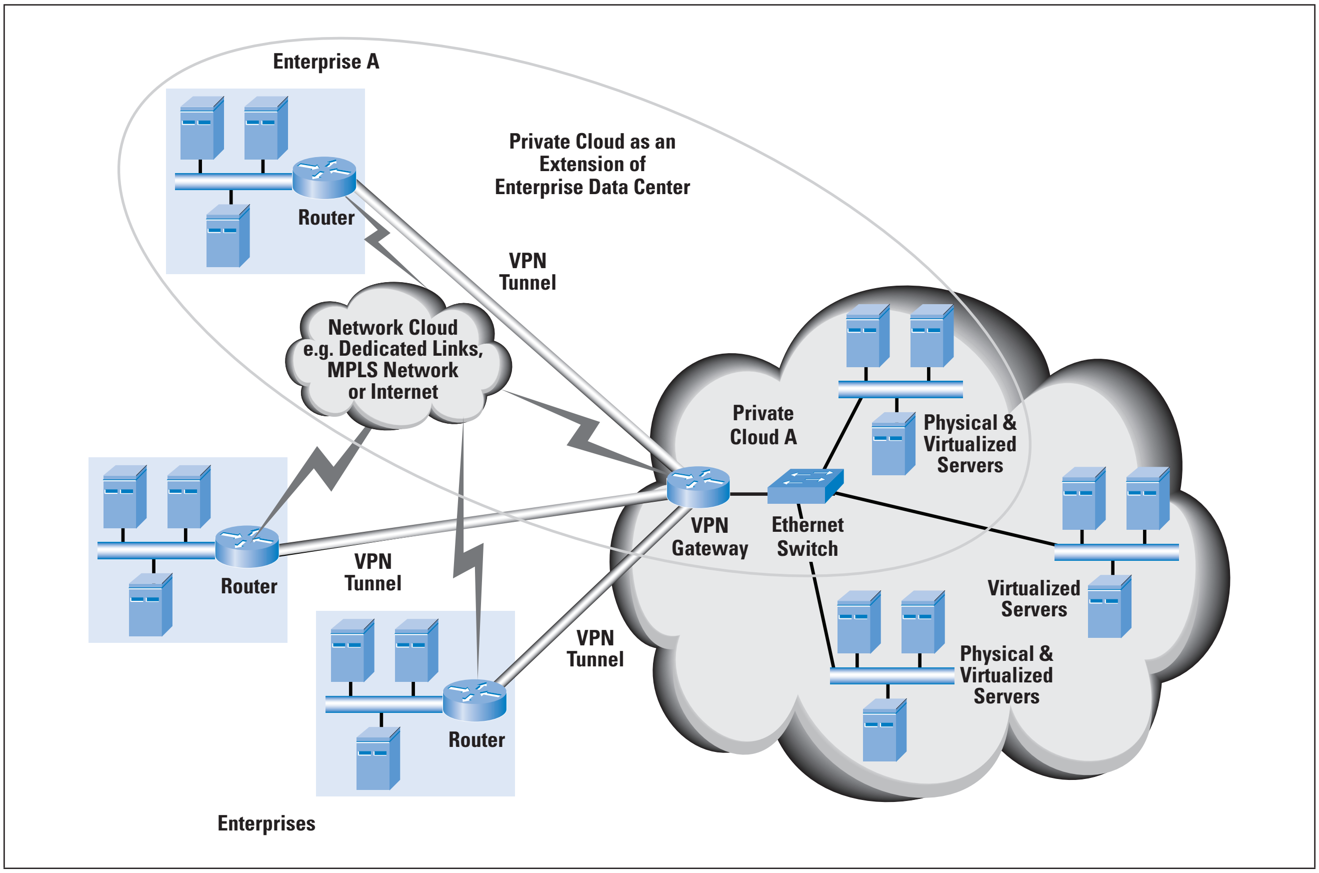
IaaS Private Clouds

Consider an IaaS cloud to which an enterprise connects to augment its server capacity for a limited period of time. Assume that the enterprise uses a `10.x.x.x` private addressing scheme for all its servers because they are internal to the enterprise. It would be ideal if the additional servers provided by the IaaS cloud were part of the same addressing scheme (the `10.x.x.x` scheme). As shown in Figure 3, the IaaS cloud service provider has partitioned a portion of its public cloud to realize a private cloud for enterprise A. The private cloud is reachable as a LAN extension to the servers in enterprise A's data center.

How is this reachability realized? A secure *Virtual Private Network* (VPN) tunnel is first established between the enterprise data center and the public cloud. This tunnel uses public IP addresses to establish the site-to-site VPN connection. The VPN gateway on the cloud service provider side uses multiple contexts—each context corresponding to a specific private cloud. Traffic from enterprise A is decrypted and forwarded over to an Ethernet switch to the private cloud for enterprise A. A server on enterprise A's internal data center sees a server on private cloud A to be on the same network.

In practice, data-center servers might be segmented into their own VLANs or IP networks according to policy and applications. The configuration and forwarding policies on the private cloud end would reflect this segmentation as well.

Figure 3: Example of Private Clouds


Enterprise A
Private Cloud as an Extension of Enterprise Data Center
Router
VPN Tunnel
Network Cloud e.g. Dedicated Links, MPLS Network or Internet
Private Cloud A
Physical & Virtualized Servers
Router
VPN Gateway
Ethernet Switch
VPN Tunnel
Virtualized Servers
VPN Tunnel
Physical & Virtualized Servers
Router
Enterprises

The following are some possible evolution scenarios for this scheme:

- *Automation of the VPN connection between the enterprise and cloud service provider:* This automation can be done through a management system responsible for the cloud bursting and server augmentation. The system sets up the VPN tunnels and configures the servers on the cloud service provider end. The management system is set up and operated by the cloud service provider.
- *Integration of the VPN functions with the site-to-site VPN network functions from service providers:* For example, service providers offer MPLS Layer 3 VPNs and Layer 2 VPNs (also known as *Virtual Private LAN Service*, or VPLS) as part of their offerings. Enterprise and cloud service providers could be set up to use these network services.
- *Cloud service providers using multiple data centers:* In such a situation, a VPLS-like service can be used to bridge the individual data centers, providing complete transparency from the enterprise side about the location of the cloud servers.

CloudNet is an example of a framework being developed by AT&T Labs and the University of Massachusetts at Amherst to address the latter two scenarios.

Layer 2 versus Layer 3 Connectivity for Cloud Networks

Enterprises and vendors follow some guidelines regarding where to use Layer 2 (switching) and Layer 3 (routing) in the network. Layer 2 is the simpler mode, where the Ethernet MAC address and *Virtual LAN* (VLAN) information are used for forwarding. The disadvantage of Layer 2 networks is scalability. When we use Layer 2 addressing and connectivity in the manner specified previously for IaaS clouds, we end up with a flat topology, which is not ideal when there are a large number of nodes. The option is to use routing and subnets—to provide segmentation for the appropriate functions at the cost of forwarding performance and network complexity.

VM migration introduces its own set of problems. The most common scenario is when a VM is migrated to a different host on the same Layer 2 topology (with the appropriate VLAN configuration). Consider the case where a VM with open *Transmission Control Protocol* (TCP) connections is migrated. If live migration is used, TCP connections will not see any downtime except for a short "hiccup." However, after the migration, IP and TCP packets destined for the VM will need to be resolved to a different MAC address or the same MAC address but now connected to a different physical switch in the network so that the connections can be continued without disruption. Proposed solutions include an unsolicited *Address Resolution Protocol* (ARP) request from the migrated VM so that the switch tables can be updated, a pseudo-MAC address for the VM that is externally managed (defined in research work being done at the University of California at San Diego), and so on.

With VPLS and similar Layer 2 approaches, VM migration can proceed as before—across the same Layer 2 network. Alternatively, it may be less complex to "freeze" the VM and move it across either a Layer 2 or Layer 3 network with the TCP connections having to be torn down by the counterpart(s) communicating with the VM. This scenario is not a desired one from an application availability consideration, but it can lower complexity.

Cloud Federation

Thus far we have considered the situation of data centers that are owned or run by the same cloud services provider. Connectivity between the data centers to provide the vision of "one cloud" is completely within the control of the cloud service provider.

There may be situations where an organization or enterprise needs to be able to work with multiple cloud providers because of migration from one cloud service to another, merger of companies working with different cloud providers, cloud providers who provide best-of-class services, and so on. Cloud interoperability and the ability to share various types of information between clouds become important in such scenarios. Although cloud service providers might see less urgency for any interoperability, enterprise customers will see a need to push them in that direction.

This broad area of cloud interoperability is sometimes known as *cloud federation*. One definition of cloud federation as proposed by Reuven Cohen of Enomaly follows:

> "Cloud federation manages consistency and access controls when two or more independent geographically distributed clouds share either authentication, files, computing resources, command and control, or access to storage resources."

The following are some of the considerations in cloud federation:

- An enterprise user wishing to access multiple cloud services would be better served if there were just a single sign-on scheme. This scheme may be implemented through an authentication server maintained by an enterprise that provides the appropriate credentials to the cloud service providers. Alternatively, a central trusted authentication server to which all the cloud services interface could be used.
- Computing and storage resources may be orchestrated through the individual enterprise or through an interoperability scheme established between the cloud providers (through a federation agreement, for example). Files may need to be transferred, services invoked, and computing resources added or removed in a useful and transparent manner. A related area is VM migration and how it can be done transparently and reliably. The *Desktop Management Task Force* (DMTF) has released a specification called the *Open Virtualization Format* (OVF) for describing a VM. It can be reasonably assumed that the payload for VM migration will be in the OVF format so that it can be interpreted across multiple vendor offerings. In effect, cloud federation has to provide transparent workload orchestration between the clouds on behalf of the enterprise user.
- Connectivity between clouds includes Layer 2 versus Layer 3 considerations and secure tunnel technologies that need to be agreed upon. Consistency and a common understanding are required irrespective of the model or technologies.
- An often-ignored concern for cloud confederation is charging or billing and reconciliation. Management and billing systems need to work together for cloud federation to be a viable option. This reality is underlined by the fact that clouds rely on per-use billing. Cloud service providers might need to look closely at telecom service provider business models for peering arrangements as a possible starting point.

Cloud federation is a relatively new area in cloud computing. It is likely that standards bodies will first need to agree upon a set of requirements before the service interfaces can be defined and subsequently realized. Provider and vendor innovation will also significantly affect this area—in fact, cloud service operators are likely to establish peering relationships and start addressing this area even before the standards bodies.

Security

As indicated in Part 1, the biggest deterrent for IT managers from venturing into cloud computing is the problem of security and loss of control. Before considering a move to a cloud service provider, enterprises need to consider some of the following security topics:

- The cloud service provider's security processes will need to be as good as or better than the processes that the enterprise uses. An audit of the vendor's processes will need to be done periodically, possibly including patches and security updates for the individual components that are used. For example, in an IaaS scenario with some preconfigured images of operating systems and applications, the cloud service provider should have the latest patches applied on the individual components.

- Infrastructure and data isolation must be assured between multiple tenants of the cloud service provider. This requirement is complicated because it is closely intertwined with the business model used by the cloud provider. For example, an IaaS provider might provide multiple tenants with VMs running on the same physical machine. Depending upon the type of work that is to be executed on the cloud, this setup may or may not be acceptable to a cloud user. In such cases, the cloud service provider should have the ability to provide separate physical servers for specific customers (and bill appropriately).

- In cases where a hypervisor and VMs are used, the hypervisor should be treated as an operating system and have the latest security patches applied to it. Security patches and updates are also essential for paravirtualized operating systems used in the VMs.

- Security functions can run as virtual appliances over hypervisors in a cloud environment. Thus it is possible for cloud users in an IaaS environment to load and configure their own firewall or other security virtual appliance to run within the cloud. The software images used for these virtual appliances need to be managed and patched similar to the way the OS, hypervisor, and other applications are managed and patched.

- Logging and audit trails for applications are important for enterprises to understand both application performance and security gaps. Cloud services providers should enable access to their application monitoring and profiling tools, where applicable.

- Authentication mechanisms ("You are who you say you are") are required at both ends of the connection—at the cloud user and cloud service provider levels. The cloud user and operator must agree upon schemes such as authentication with digital certificates and certificate authorities.

- Configuration and updates to the network infrastructure must be audited and tracked. For example, incorrect VLAN configuration on the switches can result in undesired traffic patterns between physical machines and computing resources. It would be useful to log and audit the configuration records for proper security and uptime.
- Because the cloud service is exposed to the outside world, the cloud infrastructure should support security functions such as intrusion detection and prevention, firewalling to prevent disallowed traffic, and *Denial of Service* (DoS) prevention. The cloud service is vulnerable to *Distributed Denial of Service* (DDoS) attacks—which can effectively choke its access lines, resulting in cloud users being locked out of the cloud service. Network-based DDoS prevention is a possible solution—with one of the techniques involving distribution of the cloud infrastructure to specific geographic areas and the ability to redirect cloud users in case of DDoS lockouts.

Virtualization and Security

Two options are under discussion for security in the context of virtualization. Both are useful in building out security-enabled cloud infrastructures. One option involves plug-ins to the hypervisor so that packets destined to the VMs are captured and processed by the security plug-ins. This setup enables application of security functions to the packet before it gets to the VMs. A second option is to make a specific VM handle the security functions without changing or adding to the hypervisor. The hypervisor plug-in option has the advantage of performance and initial isolation, whereas the separate VM option has the advantage of keeping the hypervisor simple and extrapolating the model that exists in physical server infrastructure. Note that these options are not mutually exclusive.

VM migration is another area where security is an important consideration. The hypervisor is responsible for the two-way communication, with the hypervisor on the destination physical machine to accomplish the migration. It is important that the connection between the source and destination hypervisors is authenticated and encrypted during the course of this migration. In addition, VM migration introduces the possibility of a DoS attack because a rogue hypervisor could overwhelm a destination machine by migrating a large number of VMs to the destination machine. Policies and logic are required at the hypervisor level to ensure that these vulnerabilities are addressed. In addition, network-based throttling might be required so that live migration does not cause congestion, which might happen if a large number of VMs need to be migrated to a destination machine at the same time.

Standards Bodies Involved in Cloud Computing

Numerous standards bodies are involved in cloud computing, addressing aspects of interoperability, virtualization migration formats, and security. Some of the organizations involved have established liaisons with the other *Standards Development Organizations* (SDOs) so that there is no duplication of effort.

The *Desktop Management Task Force* (DMTF) has specified a portable format for packaging the software to run as a VM. Known as the *Open Virtualization Format* (OVF), this package format is seeing increased use. The VM can be written onto a disk or external storage and can be moved from one physical machine to another. The DMTF has also formed a group called the *Open Cloud Standards Incubator,* which focuses on standardizing the interactions between cloud environments, including the development of resource management, packaging formats, and security.

The *Cloud Security Alliance* (CSA) is a new group formed to address security aspects of cloud computing with a focus on security assessment and management. The initial part of the effort is on developing an *Audit, Assertion, Assessment and Assurance* (API) set (A6).

The *Organization for the Advancement of Structured Information Standards* (OASIS) sees cloud computing as an extension of the *Service-Oriented Architecture* (SOA) used today in IT environments. The areas for standardization include security and policy, content format control, registry and directory standards, as well other SOA methods.

The *Storage Networking Industries Association* (SNIA) has a Cloud Storage *Technical Working Group* (TWG) that works on storage-related problems related to implementation in a cloud. The TWG has developed an interface known as the *Cloud Data Management Interface* (CDMI), which clients will use for control and configuration of the cloud.

Some Perspectives on Cloud Computing

In this section we outline and provide some perspective on cloud-computing topics that have seen interest (and some heated discussion). This list is not intended to be comprehensive but to provide a quick snapshot. Though this section has a degree of subjectivity, it is directed only to providing a broader perspective.

- *Cloud computing and SOA:* Some view cloud computing as a specific deployment case of an SOA—and this view is more popular than the one that says that cloud computing is the evolution of SOA. David Linthicum outlines that these views are complementary in that cloud-computing services will most likely be defined through SOA. IaaS provides a new variant because you can now access raw compute and storage resources as a service. Independent of the argument that “We have seen this before,” there is value to defining and invoking available services in the cloud.

- *Server virtualization schemes:* Comparisons are sometimes made based on how vendor products approach virtualization—type 1 versus type 2—and full versus paravirtualization. These approaches have pros and cons. The final decision often hinges on total costs, so it might be useful to move forward from this debate. Incidentally, vendors provide several useful tools for VM backup, recovery, fault tolerance, load management, and so on, and these tools work equally well for the various approaches to virtualization. It may be argued that these tools and features such as VM migration and the associated costs are more useful areas for comparison.

- *Other types of virtualization:* This article has deliberately omitted discussion of other types of virtualization, including desktop, application, and presentation virtualization. Some of these schemes (server-hosted desktop virtualization is one example) are affected by the cloud, specifically in the areas of network connectivity, authentication, and quality of experience. In general, any thin-client experience is affected by the cloud or data center because most of the work is done at the servers. From a cloud perspective, these types of virtualization schemes are considered to be applications that need to run reliably and consistently.

- *Data transfer and network bandwidth:* IaaS has provided a flexible model, in which you are charged based on compute power usage, storage consumed, and the duration of usage. However, there is another important factor—data needs to be sent back and forth between the cloud user and cloud service provider. Several IaaS providers charge for the amount of data transferred over the link. These charges can quickly add up if your applications are very chatty and require a lot of back-and-forth data traffic. Another concern here is the amount of time the initial upload or download can consume—for example, when you want to move a large number of your files to the IaaS provider's storage, you can tie up the link for hours. In fact, one provider has a model where cloud users can send storage media through a postal or package service for upload to the cloud provider's storage arrays.

- *WAN acceleration for the cloud:* Continuing on the previous point, chatty protocols and applications can benefit from WAN acceleration devices that can be used on both ends of a WAN link to cache and locally serve enterprise applications. These devices are not specific to the cloud—they have been used for several years for application performance improvement when a WAN link is involved. Recently, virtual network appliances for WAN acceleration are seeing deployment—here the WAN acceleration is performed by an individual VM instead of a dedicated appliance.

- *VM migration:* This article outlined some of the concerns with VM migration with respect to Layer 2 and Layer 3 topologies. Another consideration is the amount of data that needs to be moved when a VM is migrated across a network. It can potentially be in the range of gigabytes, depending upon the VM and the included operating environment.

Live migration implements this transfer in an incremental fashion so that the demand on the network is spread out. However, snapshot migration (where a VM is suspended or frozen and migrated over the network in full) can cause a surge of data on the network, leading to application performance problems for other VMs and physical machines. Throttling the amount of data that can be sent in a specific period of time, bandwidth reservation and policing at the intermediate network devices is highly desirable in such situations.

- *Management:* The current management paradigms for the cloud components are quite discrete and provide a strong level of control. For example, it is possible to log in to the *Command-Line Interface* (CLI) of a specific switch in the data center for configuration and control of the switch parameters. Similarly, it is possible to use the management console provided by the virtualization vendor to configure individual parameters for the hypervisors and VMs (for example, when to initiate VM migration to a different physical machine). Efforts are being made to unify management schemes not just through partnerships between the individual vendors but also with machine-readable interfaces (*Extensible Markup Language* [XML] being a baseline) across the multiple types of equipment and software in the cloud. Enterprise users are unlikely to accept point solutions or tools that require extensive user interaction in the long term.

- *Energy considerations:* One of the benefits of virtualization is the use of a lower number of physical servers to realize a specific function. It follows that overall energy consumption would be reduced because you have fewer servers. Although this fact may indeed be true, it would be good to characterize and monitor the effective energy savings for a specific application ("Your mileage may vary"). For example, the load on each server and the associated I/O and storage traffic may lead to higher power requirements on an individual server basis. Other considerations include the hardware infrastructure of the cloud data center because the power and cooling assumptions per rack are based on average server load.

- *Legal and regulatory considerations:* James Urquhart has compiled a set of criteria for workload migration across multiple locations, one of which is "Follow the law." Consider the case of a cloud services provider or operator that has data centers in two separate countries. The operator might use the data centers for workload migration as well as load balancing. A problem might arise if the laws in one of the countries impose limitations on what can and cannot be done at the data center. Scenarios include access to all data stored at this data center by authorities or the ability to examine all transactions on the wire at the data center. Workload migration policy statements have to be provided to cloud users so that they understand what they are signing up to. Alternatively, they might be provided the ability to set preferences for workload migration. This area is potentially worrisome, so it is important that cloud users are aware of their specific situation.

Conclusion

This article has served as a vendor-neutral primer to the area of cloud computing. In Part 1, we provided an introduction to the still-evolving area of cloud computing, including the technologies and some deployment concerns. In Part 2, we provided a more detailed look at the networking factors in the cloud, security aspects, and cloud federation. We also highlighted some areas that are seeing increased attention with cloud-computing proponents and vendors.

The area of cloud computing is very dynamic and offers scope for innovative technologies and business models. Ongoing work with respect to solutions is substantial, in the vendor research labs and product development organizations as well as in academia. It is clear that cloud computing will see significant advances and innovation in the next few years.

For Further Reading (see Part 1 for additional references)

[0] T. Sridhar, "Cloud Computing: A Primer, Part 1: Models and Technologies," *The Internet Protocol Journal,* Volume 12, No. 3, September 2009.

[1] "Building Data-Centric n-Tier Enterprise Systems," PowerVision white paper, `http://www.powervision.com/html/news/n_tier_arch.pdf`

[2] "Networking in the (Storm) Clouds," Michael Morris, `http://www.networkworld.com/community/node/43872`

[3] "Is the Relational Database Doomed?" Tony Bain, `http://www.readwriteweb.com/enterprise/2009/02/is-the-relational-database-doomed.php`

[4] "Cisco VN-Link: Virtualization-Aware Networking," `http://www.cisco.com/en/US/solutions/collateral/ns340/ns517/ns224/ns892/ns894/white_paper_c11-525307_ps9902_Products_White_Paper.html`

[5] IEEE work (in progress) on Virtual Ethernet Bridging—VEPA and VN-Tag approaches—search for "VEPA" and "VN-Tag" in the directory at: `http://www.ieee802.org/1/files/public/docs2009`

[6] "PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric," Mysore et al., `http://ccr.sigcomm.org/online/?q=node/503`

[7] "VL2: A Scalable and Flexible Data Center Network," Greenberg et al., `http://ccr.sigcomm.org/online/?q=node/502`

[8] "The Case for Enterprise-Ready Virtual Private Clouds," Wood et al., `http://www.usenix.org/event/hotcloud09/tech/full_papers/wood.pdf`

[9] "Solving the Problem of Cloud Interoperability," Reuven Cohen,
http://reuvencohen.sys-con.com/node/798504

[10] "Security Guidance for Critical Areas of Focus in Cloud Computing," Cloud Security Alliance,
http://www.cloudsecurityalliance.org/guidance/csaguide.pdf

[11] Rational Survivability Blog, Chris Hoff's blog on various topics, including cloud security,
http://www.rationalsurvivability.com/blog/

[12] "Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds" by Ristenpart, et al,
http://people.csail.mit.edu/tromer/papers/cloud-sec.pdf

[13] "Empirical Exploitation of Live Virtual Machine Migration," Oberheide et al.,
http://www.eecs.umich.edu/techreports/cse/2007/CSE-TR-539-07.pdf

[14] List of and links to cloud standards organizations,
http://cloud-standards.org/wiki/index.php?title=Main_Page/

[15] "Open Virtualization Format Specification," DMTF,
http://www.dmtf.org/standards/published_documents/DSP0243_1.0.0.pdf

[16] "SOA cloud computing relationship leaves some folks in a fog," David Linthicum,
http://www.gcn.com/Articles/2009/03/09/Guest-commentary-SOA-cloud.aspx

[17] "Is your data center ready for virtualization," Eaton white paper,
http://i.zdnet.com/whitepapers/Eaton_Is_your_data_center_ready_for_virtualization.pdf

[18] "The great paradigm shift of cloud computing is not self-service," James Urquhart, **http://news.cnet.com/8301-19413_3-10127654-240.html?tag=mncol;txt**

T. SRIDHAR received his BE in Electronics and Communications Engineering from the College of Engineering, Guindy, Anna University, Madras, India, and his Master of Science in Electrical and Computer Engineering from the University of Texas at Austin. He can be reached at **TSridhar@leitnet.com**

Protocol Basics: Secure Shell Protocol

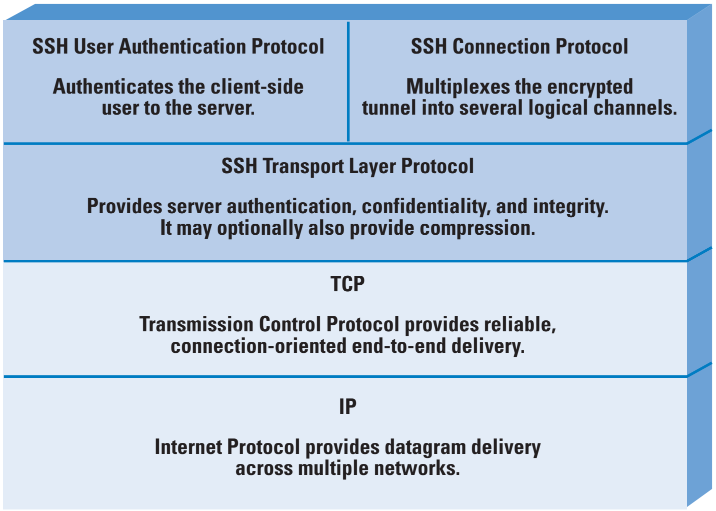
by William Stallings

Secure Shell (SSH) Protocol is a protocol for secure network communications designed to be relatively simple and inexpensive to implement. The initial version, SSH1, focused on providing a secure remote logon facility to replace Telnet and other remote logon schemes that provided no security[4]. SSH also provides a more general client-server capability and can be used to secure such network functions as file transfer and e-mail. A new version, SSH2, provides a standardized definition of SSH and improves on SSH1 in numerous ways. SSH2 is documented as a proposed standard in RFCs 4250 through 4256 [1–3], [5–8].

SSH client and server applications are widely available for most operating systems. It has become the method of choice for remote login and X tunneling and is rapidly becoming one of the most pervasive applications for encryption technology outside of embedded systems. SSH is organized as three protocols that typically run on top of TCP (Figure 1):

- *Transport Layer Protocol:* Provides server authentication, data confidentiality, and data integrity with forward secrecy (that is, if a key is compromised during one session, the knowledge does not affect the security of earlier sessions); the transport layer may optionally provide compression
- *User Authentication Protocol:* Authenticates the user to the server
- *Connection Protocol:* Multiplexes multiple logical communications channels over a single underlying SSH connection

Figure 1: SSH Protocol Stack


SSH User Authentication Protocol
Authenticates the client-side user to the server.
SSH Connection Protocol
Multiplexes the encrypted tunnel into several logical channels.
SSH Transport Layer Protocol
Provides server authentication, confidentiality, and integrity. It may optionally also provide compression.
TCP
Transmission Control Protocol provides reliable, connection-oriented end-to-end delivery.
IP
Internet Protocol provides datagram delivery across multiple networks.

Transport Layer Protocol

Server authentication occurs at the transport layer, based on the server possessing a public-private key pair. A server may have multiple host keys using multiple different asymmetric encryption algorithms. Multiple hosts may share the same host key. In any case, the server host key is used during key exchange to authenticate the identity of the host. For this authentication to be possible, the client must have presumptive knowledge of the server public host key. RFC 4251 dictates two alternative trust models that can be used:

1. The client has a local database that associates each host name (as typed by the user) with the corresponding public host key. This method requires no centrally administered infrastructure and no third-party coordination. The downside is that the database of name-to-key associations may become burdensome to maintain.

2. The host name-to-key association is certified by a trusted *Certification Authority* (CA). The client knows only the CA root key and can verify the validity of all host keys certified by accepted CAs. This alternative eases the maintenance problem, because ideally only a single CA key needs to be securely stored on the client. On the other hand, each host key must be appropriately certified by a central authority before authorization is possible.

Figure 2: SSH Transport Layer Protocol Packet Exchanges

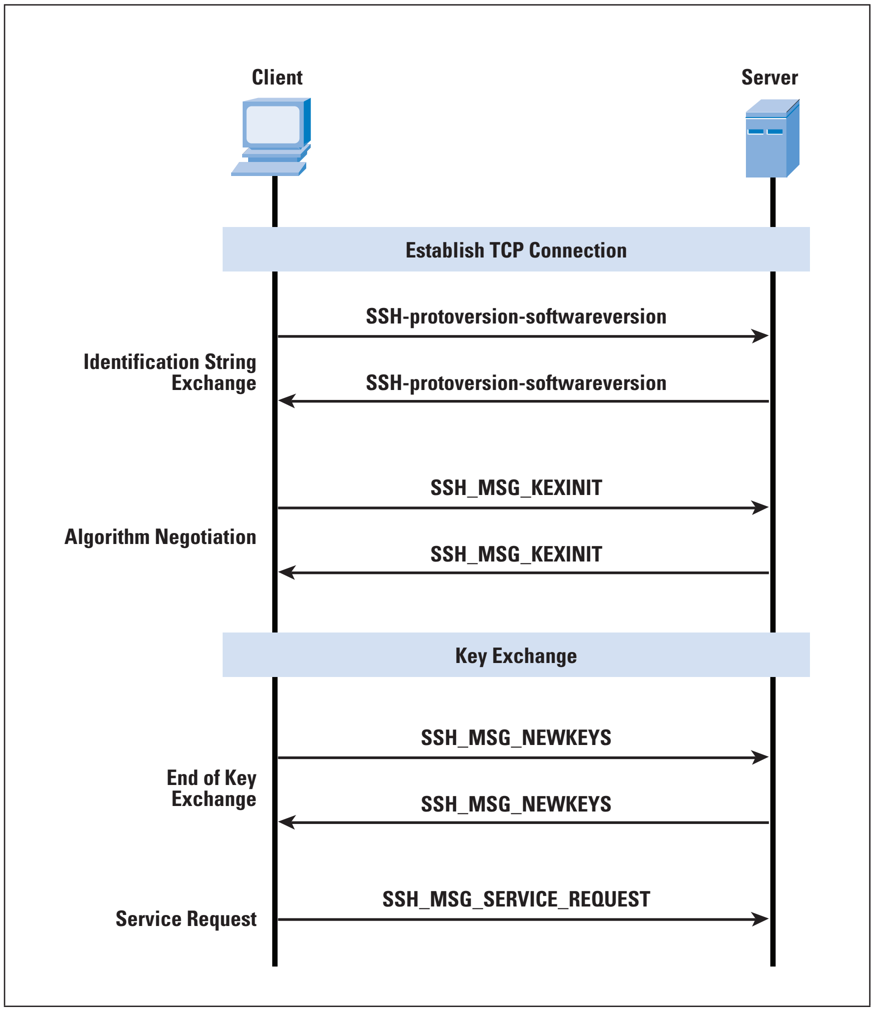
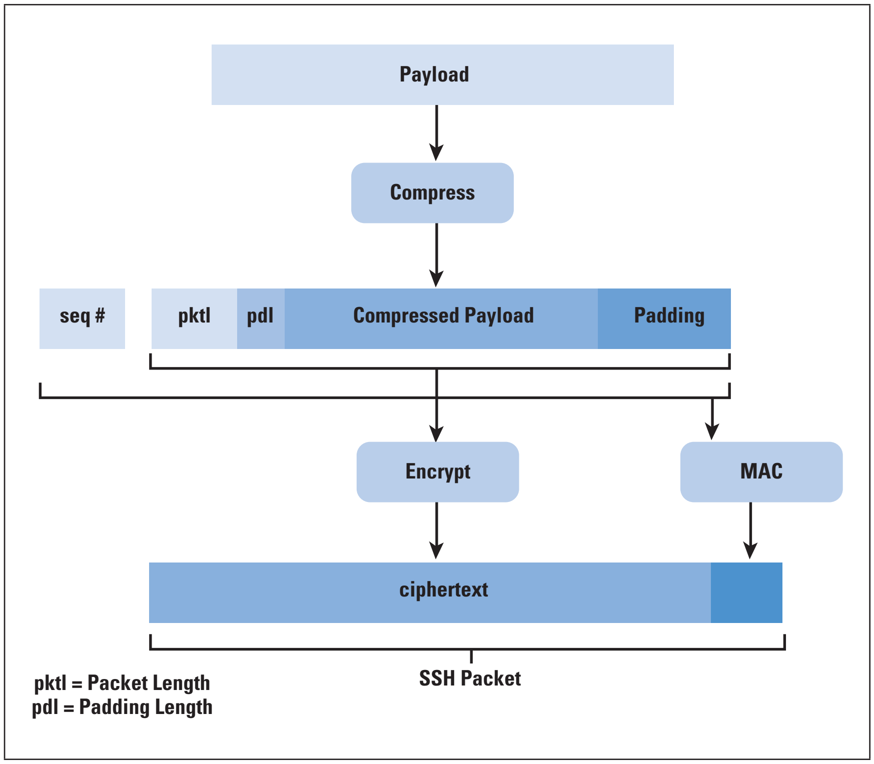


Figure 2 illustrates the sequence of events in the SSH Transport Layer Protocol. First, the client establishes a TCP connection to the server with the TCP protocol and is not part of the Transport Layer Protocol. When the connection is established, the client and server exchange data, referred to as packets, in the data field of a TCP segment. Each packet is in the following format (Figure 3):

- *Packet length:* Packet length is the length of the packet in bytes, not including the packet length and Message Authentication Code (MAC) fields.
- *Padding length:* Padding length is the length of the random padding field.
- *Payload:* Payload constitutes the useful contents of the packet. Prior to algorithm negotiation, this field is uncompressed. If compression is negotiated, then in subsequent packets this field is compressed.
- *Random padding:* After an encryption algorithm is negotiated, this field is added. It contains random bytes of padding so that that total length of the packet (excluding the MAC field) is a multiple of the cipher block size, or 8 bytes for a stream cipher.
- *Message Authentication Code* (MAC): If message authentication has been negotiated, this field contains the MAC value. The MAC value is computed over the entire packet plus a sequence number, excluding the MAC field. The sequence number is an implicit 32-bit packet sequence that is initialized to zero for the first packet and incremented for every packet. The sequence number is not included in the packet sent over the TCP connection.

Figure 3: SSH Transport Layer Protocol Packet Formation



After an encryption algorithm is negotiated, the entire packet (excluding the MAC field) is encrypted after the MAC value is calculated.

The SSH Transport Layer packet exchange consists of a sequence of steps (Figure 2). The first step, the *identification string exchange*, begins with the client sending a packet with an identification string of the form:

SSH-protoversion-softwareversion SP comments CR LF

where SP, CR, and LF are space character, carriage return, and line feed, respectively. An example of a valid string is **`SSH-2.0-billsSSH_3.6.3q3<CR><LF>`**. The server responds with its own identification string. These strings are used in the Diffie–Hellman key exchange.

Next comes *algorithm negotiation*. Each side sends an **`SSH_MSG_KEXINIT`** containing lists of supported algorithms in the order of preference to the sender. Each type of cryptographic algorithm has one list. The algorithms include key exchange, encryption, MAC algorithm, and compression algorithm. Table 1 shows the allowable options for encryption, MAC, and compression. For each category, the algorithm chosen is the first algorithm on the client's list that is also supported by the server.

Table 1: SSH Transport Layer Cryptographic Algorithms

Cipher	
3des-cbc*	Three-key Triple Digital Encryption Standard (3DES) in Cipher-Block-Chaining (CBC) mode
blowfish-cbc	Blowfish in CBC mode
twofish256-cbc	Twofish in CBC mode with a 256-bit key
twofish192-cbc	Twofish with a 192-bit key
twofish128-cbc	Twofish with a 128-bit key
aes256-cbc	Advanced Encryption Standard (AES) in CBC mode with a 256-bit key
aes192-cbc	AES with a 192-bit key
aes128-cbc**	AES with a 128-bit key
Serpent256-cbc	Serpent in CBC mode with a 256-bit key
Serpent192-cbc	Serpent with a 192-bit key
Serpent128-cbc	Serpent with a 128-bit key
arcfour	RC4 with a 128-bit key
cast128-cbc	CAST-128 in CBC mode

MAC Algorithm	
hmac-sha1*	HMAC-SHA1; Digest length = Key length = 20
hmac-sha1-96**	First 96 bits of HMAC-SHA1; Digest length = 12; Key length = 20
hmac-md5	HMAC-SHA1; Digest length = Key length = 16
hmac-md5-96	First 96 bits of HMAC-SHA1; Digest length = 12; Key length = 16

Compression Algorithm	
none*	No compression
zlib	Defined in RFCs 1950 and 1951

** = Required*
*** = Recommended*

The next step is *key exchange*. The specification allows for alternative methods of key exchange, but at present only two versions of Diffie–Hellman key exchange are specified. Both versions are defined in RFC 2409 and require only one packet in each direction. The following steps are involved in the exchange. In this, C is the client; S is the server; p is a large safe prime; g is a generator for a subgroup of GF(p); q is the order of the subgroup; **`V_S`** is the S identification string; **`V_C`** is the C identification string; **`K_S`** is the S public host key; **`I_C`** is the C **`SSH_MSG_KEXINIT`** message; and **`I_S`** is the **`S SSH_MSG_KEXINIT`** message that was exchanged before this part began. The values of p, g, and q are known to both client and server as a result of the algorithm selection negotiation. The hash function hash() is also decided during algorithm negotiation.

1. C generates a random number x ($1 < x < q$) and computes $e = g^x \bmod p$. C sends e to S.

2. S generates a random number y ($0 < y < q$) and computes $f = g^y \bmod p$. S receives e. It computes $K = e^y \bmod p$, H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K), and signature s on H with its private host key. S sends (K_S || f || s) to C. The signing operation may involve a second hashing operation.

3. C verifies that **`K_S`** really is the host key for S (for example, using certificates or a local database). C is also allowed to accept the key without verification; however, doing so will render the protocol insecure against active attacks (but may be desirable for practical reasons in the short term in many environments). C then computes $K = f^x \bmod p$, H = hash(V_C || V_S || I_C || I_S || K_S || e || f || K), and verifies the signature s on H.

As a result of these steps, the two sides now share a master key K. In addition, the server has been authenticated to the client, because the server has used its private key to sign its half of the Diffie–Hellman exchange. Finally, the hash value H serves as a session identifier for this connection. When computed, the session identifier is not changed, even if the key exchange is performed again for this connection to obtain fresh keys.

The *end of key exchange* is signaled by the exchange of **`SSH_MSG_NEWKEYS`** packets. At this point, both sides may start using the keys generated from K, as discussed subsequently.

The final step is *service request*. The client sends an **`SSH_MSG_SERVICE_REQUEST`** packet to request either the User Authentication or the Connection Protocol. Subsequent to this request, all data is exchanged as the payload of an SSH Transport Layer packet, protected by encryption and MAC.

The keys used for encryption and MAC (and any needed IVs) are generated from the shared secret key *K*, the hash value from the key exchange *H*, and the session identifier, which is equal to *H* unless there has been a subsequent key exchange after the initial key exchange. The values are computed as follows:

- Initial IV client to server: HASH(*K* || *H* || “A” || session_id)
- Initial IV server to client: HASH(*K* || *H* || “B” || session_id)
- Encryption key client to server: HASH(*K* || *H* || “C” || session_id)
- Encryption key server to client: HASH(*K* || *H* || “D” || session_id)
- Integrity key client to server: HASH(*K* || *H* || “E” || session_id)
- Integrity key server to client: HASH(*K* || *H* || “F” || session_id)

where HASH() is the hash function determined during algorithm negotiation.

User Authentication Protocol

The *User Authentication Protocol* provides the means by which the client is authenticated to the server.

Three types of messages are always used in the User Authentication Protocol. Authentication requests from the client have the format:

byte	`SSH_MSG_USERAUTH_REQUEST (50)`
string	username
string	service name
string	method name
....	method-specific fields

where *username* is the authorization identity the client is claiming, *service name* is the facility to which the client is requesting access (typically the SSH Connection Protocol), and *method name* is the authentication method being used in this request. The first byte has decimal value 50, which is interpreted as `SSH_MSG_USERAUTH_REQUEST`.

If the server either rejects the authentication request or accepts the request but requires one or more additional authentication methods, the server sends a message with the format:

byte	`SSH_MSG_USERAUTH_FAILURE (51)`
name-list	authentications that can continue
boolean	partial success

where the *name-list* is a list of methods that may productively continue the dialog. If the server accepts authentication, it sends a single-byte message, `SSH_MSG_USERAUTH_SUCCESS (52).`

The message exchange involves the following steps:

1. The client sends a **`SSH_MSG_USERAUTH_REQUEST`** with a requested method of none.
2. The server checks to determine if the username is valid. If not, the server returns **`SSH_MSG_USERAUTH_FAILURE`** with the partial success value of false. If the username is valid, the server proceeds to step 3.
3. The server returns **`SSH_MSG_USERAUTH_FAILURE`** with a list of one or more authentication methods to be used.
4. The client selects one of the acceptable authentication methods and sends a **`SSH_MSG_USERAUTH_REQUEST`** with that method name and the required method-specific fields. At this point, there may be a sequence of exchanges to perform the method.
5. If the authentication succeeds and more authentication methods are required, the server proceeds to step 3, using a partial success value of true. If the authentication fails, the server proceeds to step 3, using a partial success value of false.
6. When all required authentication methods succeed, the server sends a **`SSH_MSG_USERAUTH_SUCCESS`** message, and the Authentication Protocol is over.

The server may require one or more of the following authentication methods:

- *publickey:* The details of this method depend on the public-key algorithm chosen. In essence, the client sends a message to the server that contains the client's public key, with the message signed by the client's private key. When the server receives this message, it checks to see whether the supplied key is acceptable for authentication and, if so, it checks to see whether the signature is correct.
- *password:* The client sends a message containing a plaintext password, which is protected by encryption by the Transport Layer Protocol.
- *hostbased:* Authentication is performed on the client's host rather than the client itself. Thus, a host that supports multiple clients would provide authentication for all its clients. This method works by having the client send a signature created with the private key of the client host. Thus, rather than directly verifying the user's identity, the SSH server verifies the identity of the client host—and then believes the host when it says the user has already authenticated on the client side.

Connection Protocol

The SSH Connection Protocol runs on top of the SSH Transport Layer Protocol and assumes that a secure authentication connection is in use. That secure authentication connection, referred to as a *tunnel,* is used by the Connection Protocol to multiplex a number of logical channels.

RFC 4254, “The Secure Shell (SSH) Connection Protocol,” states that the Connection Protocol runs on top of the Transport Layer Protocol and the User Authentication Protocol. RFC 4251, “SSH Protocol Architecture,” states that the Connection Protocol runs over the User Authentication Protocol. In fact, the Connection Protocol runs over the Transport Layer Protocol, but assumes that the User Authentication Protocol has been previously invoked.

All types of communication using SSH, such as a terminal session, are supported using separate channels. Either side may open a channel. For each channel, each side associates a unique channel number, which need not be the same on both ends. Channels are flow-controlled using a window mechanism. No data may be sent to a channel until a message is received to indicate that window space is available.

The life of a channel progresses through three stages: opening a channel, data transfer, and closing a channel.

When either side wishes to open a new channel, it allocates a local number for the channel and then sends a message of the form:

byte	**`SSH_MSG_CHANNEL_OPEN`**
string	channel type
uint32	sender channel
uint32	initial window size
uint32	maximum packet size
....	channel type specific data follows

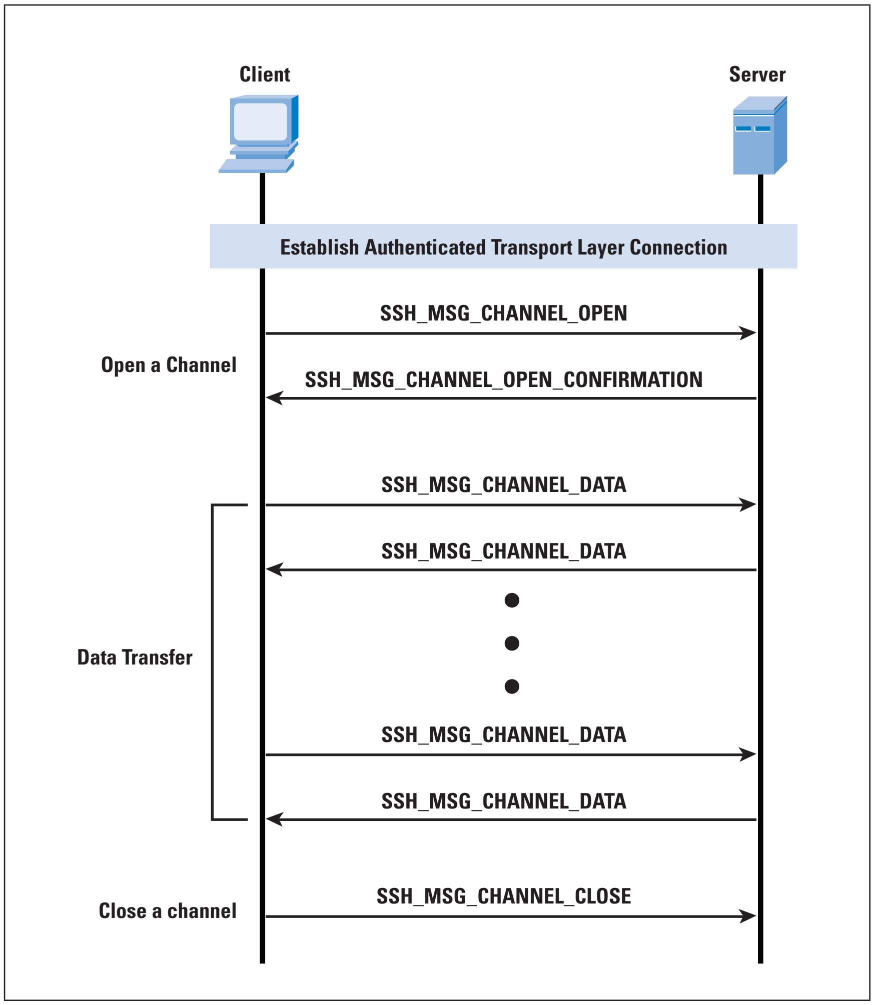
where *uint32* means unsigned 32-bit integer. The *channel type* identifies the application for this channel, as described subsequently. The *sender channel* is the local channel number. The *initial window size* specifies how many bytes of channel data can be sent to the sender of this message without adjusting the window. The *maximum packet size* specifies the maximum size of an individual data packet that can be sent to the sender. For example, one might want to use smaller packets for interactive connections to get better interactive response on slow links.

If the remote side is able to open the channel, it returns a **`SSH_MSG_CHANNEL_OPEN_CONFIRMATION`** message, which includes the sender channel number, the recipient channel number, and window and packet size values for incoming traffic. Otherwise, the remote side returns a **`SSH_MSG_CHANNEL_OPEN_FAILURE`** message with a reason code indicating the reason for failure.

After a channel is open, *data transfer* is performed using a **`SSH_MSG_CHANNEL_DATA`** message, which includes the recipient channel number and a block of data. These messages, in both directions, may continue as long as the channel is open.

When either side wishes to close a channel, is sends a **`SSH_MSG_CHANNEL_CLOSE`** message, which includes the recipient channel number. Figure 4 provides an example of Connection Protocol Exchange.

Figure 4: Example SSH Connectioin Protocol Message Exchange



Four channel types are recognized in the SSH Connection Protocol specification:

- *session:* Session refers to the remote execution of a program. The program may be a shell, an application such as file transfer or e-mail, a system command, or some built-in subsystem. When a session channel is opened, subsequent requests are used to start the remote program.
- *x11:* This channel type refers to the X Window System, a computer software system and network protocol that provides a GUI for networked computers. X allows applications to run on a network server but be displayed on a desktop machine.
- *forwarded-tcpip:* This channel type is remote port forwarding, as explained subsequently.
- *direct-tcpip:* This channel type is local port forwarding, as explained subsequently.

One of the most useful features of SSH is *port forwarding*. Port forwarding provides the ability to convert any insecure TCP connection into a secure SSH connection. It is also referred to as *SSH tunneling*. We need to know what a port is in this context. A *port* is an identifier of a user of TCP. So, any application that runs on top of TCP has a port number. Incoming TCP traffic is delivered to the appropriate application on the basis of the port number. An application may employ multiple port numbers. For example, for the *Simple Mail Transfer Protocol* (SMTP), the server side generally listens on port 25, so that an incoming SMTP request uses TCP and addresses the data to destination port 25. TCP recognizes that this address is the SMTP server address and routes the data to the SMTP server application.

Figure 5: SSH Transport Layer Packet Exchanges

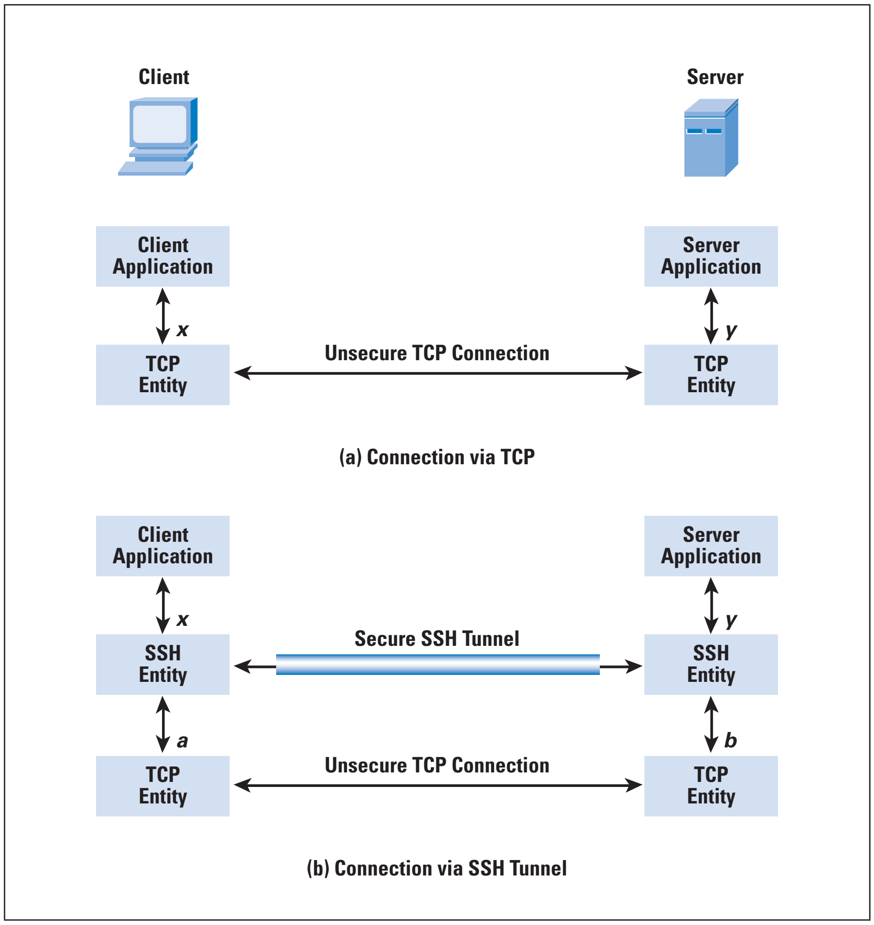


Figure 5 illustrates the basic concept behind port forwarding. We have a client application that is identified by port number *x* and a server application identified by port number *y*. At some point, the client application invokes the local TCP entity and requests a connection to the remote server on port *y*. The local TCP entity negotiates a TCP connection with the remote TCP entity, such that the connection links local port *x* to remote port *y*.

To secure this connection, SSH is configured so that the SSH Transport Layer Protocol establishes a TCP connection between the SSH client and server entities with TCP port numbers *a* and *b*, respectively. A secure SSH tunnel is established over this TCP connection. Traffic from the client at port *x* is redirected to the local SSH entity and travels through the tunnel where the remote SSH entity delivers the data to the server application on port *y*. Traffic in the other direction is similarly redirected.

SSH supports two types of port forwarding: local forwarding and remote forwarding. *Local forwarding* allows the client to set up a "hijacker" process. This process will intercept selected application-level traffic and redirect it from an unsecured TCP connection to a secure SSH tunnel. SSH is configured to listen on selected ports. SSH grabs all traffic using a selected port and sends it through an SSH tunnel. On the other end, the SSH server sends the incoming traffic to the destination port dictated by the client application.

The following example should help clarify local forwarding. Suppose you have an e-mail client on your desktop and use it to get e-mail from your mail server through the *Post Office Protocol* (POP). The assigned port number for POP3 is port 110. We can secure this traffic in the following way:

1. The SSH client sets up a connection to the remote server.
2. Select an unused local port number, say 9999, and configure SSH to accept traffic from this port destined for port 110 on the server.
3. The SSH client informs the SSH server to create a connection to the destination, in this case mailserver port 110.
4. The client takes any bits sent to local port 9999 and sends them to the server inside the encrypted SSH session. The SSH server decrypts the incoming bits and sends the plaintext to port 110.
5. In the other direction, the SSH server takes any bits received on port 110 and sends them inside the SSH session back to the client, which decrypts and sends them to the process connected to port 9999.

With *remote forwarding*, the user's SSH client acts on the server's behalf. The client receives traffic with a given destination port number, places the traffic on the correct port, and sends it to the destination the user chooses.

A typical example of remote forwarding follows: You wish to access a server at work from your home computer. Because the work server is behind a firewall, it will not accept an SSH request from your home computer. However, from work you can set up an SSH tunnel using remote forwarding.

This process involves the following steps:

1. From the work computer, set up an SSH connection to your home computer. The firewall will allow this, because it is a protected outgoing connection.
2. Configure the SSH server to listen on a local port, say 22, and to deliver data across the SSH connection addressed to remote port, say 2222.
3. You can now go to your home computer and configure SSH to accept traffic on port 2222.
4. You now have an SSH tunnel that you can use for remote logon to the work server.

Summary

SSH is one of the most commonly used cryptographic applications. It provides great flexibility and versatility for a wide variety of tasks, including remote administration, file transfer, web development, and penetration testing.

References

[1] Cusack, F. and Forssen, M. "Generic Message Exchange Authentication for the Secure Shell Protocol (SSH)," RFC 4256, January 2006.

[2] Lehtinen, S. and Lonvick, C., "The Secure Shell (SSH) Protocol Assigned Numbers," RFC 4250, January 2006.

[3] Schlyter, J. and Griffin, W. "Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints," RFC 4255, January 2006.

[4] Ylonen, T., "SSH – Secure Login Connections over the Internet," Proceedings, Sixth USENIX UNIX Security Symposium, July 1996.

[5] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Protocol Architecture," RFC 4251, January 2006.

[6] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Authentication Protocol," RFC 4252, January 2006.

[7] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Transport Layer Protocol," RFC 4253, January 2006.

[8] Ylonen, T. and Lonvick, C., "The Secure Shell (SSH) Connection Protocol," RFC 4254, January 2006.

WILLIAM STALLINGS is a consultant, lecturer, and author of more than a dozen books on data communications and computer networking. His latest book is *Cryptography and Network Security* (Prentice Hall, 2010). He maintains a computer science resource site for computer science students and professionals at **WilliamStallings.com/StudentSupport.html** and is on the editorial board of Cryptologia. He has a Ph.D. in computer science from M.I.T. He can be reached at **ws@shore.net**

Book Review

Protocol Politics

Protocol Politics: The Globalization of Internet Governance, by Laura DeNardis, MIT Press, 2009, ISBN 978-0-26204257-4.

In *Protocol Politics,* Dr. Laura DeNardis assembles a variety of stories gleaned from official and unofficial *Internet Engineering Task Force* (IETF) records and firsthand accounts, and supplements them with primer-level descriptions of successive generations of Internet addressing and routing protocols to create a broadly accessible overview of the factors that have shaped the present and evolving state of these most central features of Internet technology.

The author, a former enterprise networking consultant and technology analyst, joined the Yale Law School Information Society Project as a Post-Doctoral Fellow in 2006, and became the Executive Director of the program in late 2008. DeNardis approaches the challenge of organizing these disparate materials by adopting an interpretive framework that highlights the role of power—interpersonal as opposed to electrical—as both the primary input and most important output or consequence of the definition, selection, and implementation of Internet protocols.

The book knits together a wealth of important historical information that has to-date remained largely neglected outside of the technical community. Although DeNardis' choice of framing is perfectly legitimate—and in fact quite common within the academic disciplines that delve into the influence of institutions on industries, economies, and society—in this case it leads her to overreach a bit, and arguably to draw a few prominent conclusions that are not well-supported by the balance of available historical evidence.

Organization of the Book

DeNardis employs this interpretive framework across six densely written chapters, the first four of which directly address the significance of power in a different functional context of relevance to the evolution of Internet addressing and routing. The introductory chapter investigates the significance of scarcity and its effect on protocol resource management and *Internet Governance*. Here she devotes considerable space to detailing the critical importance of IP addresses as the single element among Internet protocols that is both indispensable and nonsubstitutable. DeNardis' insightful overview of the general characteristics of IP addresses is somewhat marred by her mixing together of some basic, intrinsic functional properties of addressing (for example, *identifier* and *locator* functions) with various necessary but extrinsic correlates or consequences of those functional properties (for example, universality, external observability), or with contingent features of current IP address usage conventions (for example, indifference to underlying technologies).

In addition, despite the ostensible focus on scarcity in the chapter, no reference is made to that other, equally essential and quantity-constrained feature of the Internet service landscape—that is, the inherently limited, occasionally overtaxed carrying capacity of Internet routing subsystems, particularly the collectively provisioned inter-domain routing system. Overall, *Protocol Politics* provides almost no exposure to the technical, operational, and economic constraints that define the routing environment, much less to the constraints that those factors impose on number resource distribution arrangements. Chapter One closes with an overview of the priorities that justify and define the sphere of Internet Governance which anticipates many of the concluding observations in the book's final chapter on "Opening Internet Governance." Both chapters acknowledge "technical expertise" only as a source of institutional or political legitimacy, without according any special significance to the *content* of such expertise, or why it matters at all. Readers of *Protocol Politics* may thus come away with insufficient appreciation of the fact that before Code can become Law (or anything else), it first must be running code—*and that not every wish is translatable into running code.*[1]

Piercing the Fog of Protocol War

In the three chapters that follow, DeNardis presents her observations about how power shapes and flows from the definition and selection of Internet protocols. Chapter Two covers the first half of this proposition, focusing on the events that followed the December 1990 IETF meeting where, DeNardis suggests, the twin challenges that would shape the development of Internet addressing intersected with the chief institutional impediment that would ultimately reveal the true political nature of Internet standards development.

The first challenge that she identifies is the foreseeable inadequacy of IPv4 as the exclusive addressing resource pool for a rapidly growing and globalizing Internet. In keeping with the overall theme of the book, the second challenge that DeNardis chooses to highlight is the implicitly political challenge of accommodating greater international participation in the U.S.-centric Internet technical coordination and decision-making bodies. Against this backdrop, DeNardis introduces the other chief protagonist in her story, the *International Organization for Standardization* (ISO), which backed the rival *Open Systems Interconnection* (OSI) family of protocols as an alternative, non-TCP/IP-based foundation for the ongoing, global proliferation of data networking. DeNardis details the convoluted, multidimensional deliberations that followed that 1990 IETF meeting, which eventually culminated in 1994 in the formal recognition of IPv6 as "The Next-Generation Internet Protocol."

Chapter Three goes on to explore the implications of both IPv4 and IPv6 for important civil liberties—especially privacy—and how such considerations did and did not, *but hypothetically might have,* influenced the choice and form of the most important features of TCP/IP.

Chapter Four rounds out the central thesis of the book by illustrating how various national-level considerations—especially government-directed foreign and domestic economic policies—have resulted in an increasingly diverse global pattern of IPv6 adoption.

DeNardis' detailed account of the complexities surrounding the *IP Next-Generation* (IPng) debate and its aftermath incorporates a diverse mix of sources, from pointed remarks made on various mailing lists, to conference presentations and official *Internet Architecture Board* (IAB) meeting minutes, and represents a major feat of historical scholarship. That said, her presentation of "relevant historical facts" from the 1990–1994 period is by no means complete, nor is her interpretation of the facts that she does cover or the conclusions that she draws from them immune to criticism. For example, in puzzling over possible hidden forces behind the selection of IPv6, DeNardis states that:

> "If anything, there was market pressure to adopt an OSI rather than TCP/IP-based protocol. The ISO alternative had the political backing of most Western European governments (sic) influential technology companies, and users invested in OSI protocols, and was even congruent with OSI directives of the United States. The selection of IPv6…" (p. 61)

Although these facts may be beyond dispute, they do not represent the full picture. To give one illustration, in 1989, almost 2 years before the date that DeNardis marks as the start of the IETF's lone struggle against the combined forces of Europe, influential carriers and hardware manufacturers, and the U.S. government, an indigenous movement of European network operators emerged and began self-organizing to facilitate the exchange of TCP/IP-based traffic, contact information, and operational tips, and to discuss best practices in areas of networking where individual network-level decisions could have far-reaching effects on internetwork performance.

That organization would go on to become *Réseaux IP Européens Network Coordination Centre* (RIPE NCC), the first independent, transnational registry for Internet Protocol number resources, and the institution that would provide the organizational template for the *Regional Internet Registries* (RIRs) that subsequently sprang up in Asia (APNIC, 1993), North America (ARIN, 1997), Latin America (LACNIC, 2002), and Africa (AFRINIC, 2004). These facts point to a level of active indigenous European support for TCP/IP-based networking that would seem to be at odds with any suggestion of a continent united in support of OSI against a less-attractive standard being pushed by an insular foreign organization.

Thus, regardless of whether DeNardis' concerns about institutions and power relations are well-founded, her intuitions about the division of contestants in the great protocol power struggle clearly are not.[2]

Market Contrast

Another question that DeNardis raises, obliquely but repeatedly, relates to the possibility of "free markets" as an alternative mechanism for defining, selecting, and distributing Internet protocols and the virtual resources that they create.

In no less than a dozen separate passages scattered across each of the chapters in the book, DeNardis sharply contrasts a range of IETF and RIR institutional processes to the workings of the "free market." For example, she observes that the value of IP addresses is unknown because they have never been exchanged in free markets (p. 16); that Internet addresses have never been exchanged in free markets (pp. 23, 190); that the privacy potential of Internet technologies is enhanced by selection pressures from free markets (p. 74); that the IETF refused to countenance an IPng protocol selection made by free markets (p. 51); that the selection of IPv6 happened outside the realm of free markets (p. 69); that widespread adoption of IPv6 is impeded by the absence of a free market for protocols (p. 137); that IETF philosophy holds that it would be inappropriate to exchange protocol resources in free markets (pp. 163, 183–184); that the *Internet Assigned Numbers Authority* (IANA) refused to relinquish IP addresses to free markets (pp. 163, 164); that traditional opposition to the exchange of protocol resources in free markets fortified and centralized the IETF's institutional control (p. 184); and that exchanging IPv4 in free markets has pragmatic appeal, if only as a temporary stopgap (p. 228), although such exchanges might have unintended consequences (p. 229).

Given this frequency of repetition, it is impossible to avoid forming a strong impression of DeNardis' underlying opinion about the intrinsic merits of "free markets" as compared to the seemingly market-antithetical goals and practices of the IETF and the other TCP/IP-centric standards-setting and technical coordination bodies. However, even if one stipulates that "free markets" would by definition represent a superior alternative to the enumerated protocol design and distribution mechanisms, DeNardis never provides any clear indication of where a model for such "free markets" might be found—whether in Europe, the United States, or anywhere else, now or anytime in the past.

Even her own description of that fateful moment in networking history when IPv6 was selected clearly suggests that the alternative to the IETF process that ultimately prevailed was itself neither "free" nor especially market-like:

> "… congruent with OSI directives of the United States. The selection of IPv6, an expansion of the prevailing IPv4 protocol *over such a politically sanctioned OSI alternative* solidified and extended the position of the Internet's traditional standards-setting establishment as the entity responsible for the Internet's architectural direction." (p. 61, emphasis added).

Arguably, the non-inclusion of a pure “free market” example is not merely a coincidence, but rather reflects a more fundamental problem inherent in the concept itself. Further, if one grants that the market mechanism that is *most free* is the one that fosters the broadest participation in those activities that make markets attractive—including openness to participation, exercise of individual choice, competition, accelerated innovation, and wealth creation—then one might interpret the two-plus orders-of-magnitude growth in the number of independent network services providers operating on both sides of the Atlantic since that time as a solid indicator that markets have not suffered too badly from the 1994 decision to extend the lifetime of TCP/IP through IPv6.

Clearly the looming inflection point in IP addressing will provide many irresistible opportunities to revisit that choice in the days ahead. Meanwhile, the question of whether the embrace of an OSI-friendlier IPng by the IETF would have been sufficient to offset the varied negative externalities that might have accompanied such a choice must forever remain unanswered. Would an IETF endorsement have trumped the as-yet incomplete state of OSI standards, as well as OSI’s tighter associations with non-standards-based operating systems, proprietary hardware platforms, and the connection-oriented networking technologies favored by then Internet-averse incumbent *Public Switched Telephone Network* (PSTN) operators? Would that choice alone have created or been likely to foster a freer market, or to have led to a more enthusiastic, widespread embrace of a different post-IPv4 addressing format—or alternately would it have led to the appearance of books like *Protocol Politics,* albeit written from the opposite perspective, and possibly a decade sooner? Contrary to the popular adage, hindsight is not 20/20, any more than is our vision of where to go from here.[3]

Beyond the Clash of Idealizations

Writing a book review is an inherently risky undertaking, one that is vulnerable to many of the same human biases and errors that have unquestionably informed both the selection and development of various technical standards, just as they have influenced the embrace, rejection, or modification of various market arrangements throughout history.

Even when people (book reviewers, for example) recognize that real-world decisions and their consequences tend to be irreducibly complex—or perhaps precisely because they recognize that complexity—they nevertheless tend to gravitate toward explanatory frameworks and cognitive models that promise to invest their perceptions and choices with the kind of absolute certitude that is very rarely found outside of the physical world (and only infrequently found there).

The problem, of course, is that many such explanatory frameworks can be found to fit quite nicely with the same set of human experiences, even though some of those models may be mutually orthogonal, and some may be quite mutually and actively antagonistic. In this sense, the juxtaposition of pure, frictionless "free markets" alongside the idea of absolutely pure scientific or technical decision making divorced from all other human considerations, while well-calibrated to inflame passions, represents less a contrast of opposites than a rather less illuminating pairing of two deeply unrealistic ideal types. Distilling a book as rich and informative as *Protocol Politics* down to one possible review-sized essence is much easier to accomplish from just such a privileged vantage point, and no doubt this particular review suffers from the all-too-predictable effects described herein. However, with that caveat firmly established, a few more things about *Protocol Politics* deserve to be mentioned here.

First, *Protocol Politics* is an important book. It is the well-written and informative, and is the first to be written for a general audience that draws on the right historical sources (or at least most of the right ones that remain accessible) to cover this critical period in the development of the Internet's core addressing and routing protocols. Even those who are least likely to be sympathetic to its findings are likely to find *Protocol Politics* to be a thoughtful and engaging read.

Second, IPJ readers and other technologists should not dismiss the inherently political, power-oriented framework that DeNardis employs in *Protocol Politics*. In general, the most honest and effective response to an assertion of *systemic* political or institutional bias is not to claim an equally absolute, otherworldy detachment from the affairs of man, but rather to remind the critic that in a world where all institutions are regarded as manifestations of somebody's will to power, specific targeted criticisms based *solely* on that fact lose all coherence. Would-be institutional critics who espouse such views thus have no choice but to make a positive argument as to which arrangement, among all of the equally power-tainted institutional arrangements that are possible, should be regarded as the preferable outcome, for whom, and why. Judged in this light, this reviewer feels that "the IETF way" still stands up pretty well, foibles and all. There is always room for improvement, but just as in matters of code, a concrete proposal for improvement is worth a thousand critiques of the past.

Finally, the careful reader may notice a pattern within this review, one composed of points highlighted here even though they may not be equally central to the story presented in *Protocol Politics* (for example, about the role of technical expertise in Internet governance, the dynamic limitations of routing system carrying capacity, the possibility of free market alternatives to current Internet address distribution arrangements, and so on).

Each of these points merits special attention because taken together they help to illuminate the existence of an identical set of critiques that have reappeared periodically in the course of another, much older (actually, centuries-old) debate that parallels the as-yet unresolved debates outlined by DeNardis in *Protocol Politics*.

In both instances, the question at issue involves the relative merits of nonmarket, technical expert-based systems as a means of managing resources that are uniquely central to economic growth, and for mitigating the systemic risks that can threaten that growth. In that other debate, arguments in favor of pure free market solutions have generally been dismissed as extreme and unrealistic for more than a century, ever since the last real-world implementation of such a system finally succumbed to its own chronic instabilities and was replaced by a nonmarket coordination arrangement. More recently, however, a resurgence of extreme turmoil in that parallel industry has undermined belief in expert management, if not in the underlying "hard realities" that were supposed to constitute the managers' technical domain of expertise. In turn this turmoil has sparked renewed interest in the long-marginalized pure free market proposals, as well as in alternative remedies involving much tighter industry control by nonmarket authorities.

How the current chapter in either of these parallel stories will play out remains to be written. However, those who are eager to anticipate the kind of language that is likely to play a central role in both outcomes will find that a close reading of *Protocol Politics* provides a wealth of possibilities to consider, and more than a few to keep one up at night.

—Tom Vest, Consultant
`tvest@eyeconomics.com`

References

[1] DeNardis makes several references to the idea that *Code is Law,* which was first articulated by Larry Lessig in *Code and Other Laws of Cyberspace* (1999) [Editor's note: *Code* was reviewed in IPJ Volume 11, No. 3]. Here the phrase is juxtaposed with David Clark's famous paean to "rough consensus and running code," which DeNardis describes as an "articulation of the IETF's core philosophy" (p. 47), and amended with a paraphrasing of an early (c. 1992) observation made by Marshall Rose about a common problem encountered when attempting to implement code to satisfy a non-operationally developed standard. The original staying was, "The problems of the real world are remarkably resilient to administrative fiat."

[2] Several formerly obscure insights on the events of this period were recently illuminated by RIPE co-founders Rob Blokzijl and Daniel Karrenberg, during RIPE's 20th Anniversary Commemoration at the RIPE 58 meeting in Amsterdam (May 2009). Some of these are available at: **http://www.ripe.net/ripe/meetings/ripe-58/content/presentations/Blokzijl-RIPE-20-years.pdf**

and

http://www.ripe.net/ripe/meetings/ripe-58/content/presentations/the-origins-of-ripe.pdf

[3] Those wishing to investigate these questions further may benefit substantially from yet another unique historical resource that has recently been made available online. Thanks to the Charles Babbage Institute and the Institute of Technology at the University of Minnesota, the entire ten-year archive of *ConneXions—The Interoperability Report* (1987–1996) is now available online at: **http://www.cbi.umn.edu/hostedpublications/Connexions/index.html**

In keeping with its mandate to track the interoperability of emerging network technologies, *ConneXions* published more than sixty substantive articles on OSI and GOSIP during the period leading up to and following the IPng debates recounted in *Protocol Politics*.

Fragments

Lorenzo Colitti (L) and Erik Kline
Photo: Matsuzaki Yoshinobu

Colitti and Kline Receive First Itojun Service Award

The first *Itojun Service Award* was presented at the recent IETF meeting in Hiroshima, Japan to Lorenzo Colitti and Erik Kline of Google for their outstanding contributions to the development and deployment of IPv6.

The award honours the memory of Dr. Jun-ichiro "Itojun" Hagino, who passed away in 2007, aged just 37. Established by the friends of Itojun and administered by the *Internet Society* (ISOC), the award recognises and commemorates the extraordinary dedication exercised by itojun over the course of IPv6 development.

"The sustained efforts of Lorenzo and Erik have tangibly increased the availability of Web-based services that use IPv6, reflecting the Itojun Service Award's focus on pragmatic contributions in the spirit of serving the global Internet's continued evolution," said Jun Murai of the Itojun Service Award committee and Director of the WIDE Project. "The award aims to recognize how important both the development of IPv6 and related protocols and efforts to advance their deployment are to ensuring the Internet continues to serve as a platform for innovation around the world."

The award, expected to be presented annually, includes a presentation crystal, a US$3,000 honorarium and a travel grant.

Lorenzo Colitti, Network Engineer at Google said, "This is a great honour. Itojun is a legend in the IPv6 community, and the Internet is indebted to him. Without his foundational work, none of what we achieved with IPv6 would be possible—we stand on the shoulders of giants. Itojun has been a source of inspiration, and I regret never being able to meet him, to show him our work, and show him that we too shared his vision of bringing IPv6 to the users of the Internet."

Erik Kline, IPv6 Software Engineer at Google said, "It's humbling to be sharing the Itojun Service Award, having achieved by comparison only a small fraction of the impact of his widely influential body of work. For me personally, Google's IPv6 efforts are not just for the Internet and its future, but also a way to honor his vision, dedication, and passion."

More information on the Itojun Service Award is available at: `http://www.isoc.org/itojun`

ISOC Donation to Support Evolution of W3C Organization

ISOC and the *World Wide Web Consortium* (W3C) recently announced a donation from ISOC for the purpose of advancing the evolution of W3C as an organization that creates open Web standards. Citing strongly aligned views on the value of an open global Internet and support for the current Internet governance and management model, ISOC pledged to support W3C efforts to implement a more agile, inclusive, and flexible organizational structure.

"ISOC and W3C have worked together for years in a number of areas, and have deeply shared values about the Internet's development," said Lynn St. Amour, President and CEO of ISOC. "Our support to the W3C in their transition efforts demonstrates our commitment to ensuring the Internet continues to be a global platform for innovation. What's at stake is the Internet's openness, which is a critical enabler of new products and services to billions of users worldwide."

"ISOC and W3C have a long history of cooperation and the Internet ecosystem has benefited from our shared yet independent voices," said Tim Berners-Lee, W3C Director. "The W3C staff, Members, and community continue to work on making W3C more relevant and valuable to the Web and Internet communities. ISOC support will allow W3C to evolve its structure to ensure we continue to forge solid working relationships with the increasing numbers of developers and users, worldwide."

The two organizations will continue to operate independently, and will maintain their long-standing, informal collaboration. ISOC's pledge of support is for three years, with both organizations working to ensure progress. A FAQ with additional information is available on both the ISOC site and the W3C site, see **`http://www.isoc.org`** and **`http://www.w3.org`**

DNSSEC Deployment in the Root Zone

In December 2009, ICANN and VeriSign began to deploy DNSSEC across the root server system and launched a website that provides information about DNSSEC for the root zone. The website is a repository for the documentation relating to the deployment of DNSSEC, and it includes information such as technical status updates and the full timetable for the deployment osf DNSSEC.
See: **`http://www.root-dnssec.org/`**

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

The Internet Protocol Journal, Cisco Systems
170 West Tasman Drive
San Jose, CA 95134-1706
USA

ADDRESS SERVICE REQUESTED

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

Printed in the USA on recycled paper.