

# Intel's Internet Connectivity: Evolution, Technical Architecture, and Future Directions

Cindy Bickerstaff, Intel® Online Services, Intel Corporation  
Sally Hambridge, Intel Online Services, Intel Corporation  
Lynne Marchi, Information Technology, Intel Corporation  
Tod Oace, Intel Online Services, Intel Corporation  
Stacy Purcell, Information Technology, Intel Corporation  
Jeff Sedayao, Intel Online Services, Intel Corporation  
Charles Smothers, Information Technology, Intel Corporation  
Ken True, Intel Online Services, Intel Corporation

Index words: the Internet, connectivity, firewalls, security, performance, measurement

## ABSTRACT

This paper describes Intel's Internet connectivity architecture, including the business drivers that led to its creation and the technology developed to build and maintain it. Intel's first Internet connection was a 2400 bit per second modem used to pick up and deliver e-mail for a small community of engineers and researchers. With the advent of the Mosaic Web browser, Intel needed to develop a scalable, high speed, highly available Internet connectivity architecture that is secure, available, provides good performance, and minimizes the impact of Internet usage on Intel's internal network.

An architecture was developed and implemented that provided multihomed Internet gateways at major Intel sites. A screened subnet firewall architecture provides defense and depth. Several fail over modes were implemented to maintain connectivity even if a number of gateway components failed. To manage this distributed gateway implementation, we borrowed methods from Intel's manufacturing world such as Copy Exactly! and certain measurement and experimental methodologies and statistical techniques. We modelled our set of distributed gateways as a single system and drove configuration through a Makefile.

Our architecture has proven highly successful. It supports Internet access for tens of thousands of Intel employees' Internet access, and Intel does \$1 billion a month in e-Commerce through this infrastructure. Current challenges include maintaining performance

while dealing with growth and security threats. Future uses of the Internets are dial-in modem replacement, intercompany connectivity, virtual private networking, and streaming media.

## INTRODUCTION

Intel initially connected to what would become the Internet in 1986 [1]. The first connection took place using a 2400 bit per second modem that was used to pick up and deliver e-mail for a small community of engineers and researchers within the company. Connectivity eventually evolved into a direct leased line into the Internet. With the advent of the Mosaic Web browser, Internet usage became mainstream. As we foresaw that the Internet would become as important a tool to business as the telephone, Intel's Information Technology group faced a number of challenges in providing Internet services:

- provide Internet services with good performance
- scale service to deal with increasing numbers of servers, services, and users
- make Internet services like Web access and e-mail highly available and robust
- ensure that our Internet gateways are secure
- minimize any impact of Internet traffic across Intel's internal network

To meet these challenges, we implemented an Internet connectivity architecture that provides Internet gateways at major Intel sites. Connecting to at least two Internet Service Providers (ISPs) at each site increased availability and performance. We implemented a screened subnet firewall architecture to provide defensive in-depth security, and we made our gateway capable of several different fail over modes in order to make it as highly available as possible.

The major challenge of a distributed Internet gateway system is managing it. We had to ensure that gateways were operating correctly, the Internet service was good, and consistent security policies were being enforced. Moreover, the staff maintaining Intel's Internet gateways was fairly small, less than ten people. To manage the Internet gateway system, Intel's IT took a number of concepts from Intel's manufacturing arm, such as Copy Exactly! [2], a technique to simplify deployment and maintenance of systems and configurations. We also used measurement and statistical methodologies developed and honed in wafer fabrication plants to measure and monitor Internet performance [3]. In addition, we simplified our maintenance by modelling our distributed gateway system as a single computer and the router and server configurations as software modules to be compiled and installed [4].

Our architecture has proven highly successful. It supports tens of thousands of Intel employees' Internet access, and Intel does \$1 Billion a month in e-Commerce through this infrastructure. We have managed to scale almost all of the components of the architecture, from bandwidth and ISPs to servers and users. Policies and techniques developed while implementing this architecture have gone into Internet Standards [5] and industry white papers [6], and many of the techniques developed are being implemented in Intel's new business unit, Intel® Online Services.

The constant challenge that we have seen is keeping up with increasing usage of the services we offer and the number of new services that users demand. Another key challenge is dealing with the ever increasing and changing security threats. In the future, we see the Internet used increasingly for dial-in modem replacement, intercompany links, and for virtual private networking. We foresee a time when every site will have its own Internet connection, although we see challenges to this vision in non-US locations where bandwidth costs are problematic [7].

This paper describes how we designed and implemented Intel's Internet connectivity architecture. It covers the early drivers for connectivity, the requirements and design issues for a connectivity architecture, and the

technologies and policies needed to implement and maintain the architecture that we built.

## **EARLY DRIVERS FOR INTERNET ACCESS**

Intel first connected into what would become the Internet in 1986 [1] via an organization called CSNET. The primary driver for connectivity was the exchange of e-mail between Intel engineering organizations and consortia outside of Intel, such as the Semiconductor Research Corporation and Sematech. The Internet environment (then the ARPANET) was much different during that time period. It was designed primarily as a research environment, and there were explicit restrictions on commercial use. Direct connections through leased lines were expensive and not easy to come by, so Intel exchanged mail through a system called PhoneNET. Intel dialed up a server, dropped off outgoing e-mail, and picked up incoming e-mail. All this happened through a 2400 bit per second modem.

The power and utility of being able to communicate with people in other organizations electronically made e-mail use grow exponentially. This was despite the fact that Intel didn't offer classes or documentation on how to use the Internet and did not even advertise the fact that we were connected!

By the late 1980's, large archives of freely distributed and highly useful software began to appear. There was increasing demand for access to these archives, for which direct connectivity to the Internet was necessary. As Internet mail use at Intel grew exponentially, the dial-up charges began to approach the costs of a leased line. At that point, Intel obtained one 56 Kilobit leased line to the Internet for general use by employees. The Internet then was largely used by engineering personnel and systems administrators. Systems with Internet access were scattered haphazardly across the company and managed by a variety of groups.

The Mosaic browser and the World Wide Web brought the Internet into mainstream use at Intel. Before then, the Internet was largely text-based. It was mainly used for e-mail, FTP (file transfer), and remote login. The menuing system that preceded the Web, Gopher, was mostly textual. The Web's use of the in-line graphics increased Internet connectivity bandwidth requirements. As the Internet became commercialized and the Web began to take off, Intel needed to seriously revamp the way it connected to the Internet. At that point, we knew Internet connectivity and services were as critical as the phone. The state of Internet services at the time was haphazard and not secure. Servers used to provide Internet access

and provide Internet services like DNS were scattered across the internal network and inconsistently managed.

## CONNECTIVITY ARCHITECTURE AND IMPLEMENTATION

We knew we had to move to an Internet connectivity architecture that supported Internet service as a robust, secure service used by Intel employees, customers, and suppliers. This section talks about how we designed and implemented an architecture that met that goal. First we discuss key requirements and design tradeoffs. We then describe the connectivity and security design, followed by a section on how we maintain our large distributed system. Finally we discuss how we ensure that we receive quality connectivity service, and how we crafted policies to guide Internet usage.

### Key Requirements and Design Tradeoffs

With the growth of the World Wide Web and the increasing use of the Internet by all employees, we decided to view the Internet as a key utility, as basic and necessary a business tool as the telephone. Like telephony service, access to web sites and electronic mail exchange with other companies would be used by everyone, not only by a small community of engineers and scientists. With that as a model, several requirements immediately came to mind:

*High Availability and Reliability.* Internet services had to be as available and reliable as the telephone.

*Security and Accountability.* The service could not leave Intel vulnerable to intruders and could not be vulnerable to denial of service attacks. The Internet architecture also had to allow Intel to keep track of employee use so that data were available when deciding on growth. Also, Intel needed to be able to track possible misuse by employees.

*Good Performance.* Our architecture needed to perform well so employees could be productive in their use of the Internet. Good performance is also related to security: a system that is not performing well enough to be usable will be worked around, usually in a not secure way.

*Maintainability.* The staff maintaining our Internet connectivity was small and was not going to grow rapidly. Whatever architecture was implemented needed to be easily maintained remotely by a small staff. When something went wrong, we needed an architecture that could deal with it and continue to work.

*Scalability.* In the early years of Internet connectivity, we saw exponential growth in traffic to the Internet year after year. Whatever we designed had to be able to scale in almost every way we could imagine. We knew that the number of employees using this infrastructure and the

time they spent using this infrastructure would constantly increase. Our architecture needed to be able to grow quickly to meet this demand—from increases in bandwidth to the number of servers providing service.

*Impact on Intel's internal Network.* Intel has a number of critical internal applications, such as chip design and semiconductor manufacturing, as well as its internal e-mail system. Use of the Internet should not impact these critical uses of Intel's extensive internal network.

### Critical Decisions

A number of critical decisions had to be made at this point:

- Should the gateways be centralized or distributed.
- Should Intel connect to one ISP or several. If more than one, how many.
- How should the network be laid out so that it is highly available, secure, and yet scalable.

*Centralized or distributed gateways.* There are several tradeoffs between having a single gateway with centralized services and having several distributed gateways. Having a single gateway is much easier to manage. It is also better from a security standpoint, as there is only one gateway to watch and control. A single gateway is also much cheaper to provision and maintain.

Having several gateways is better from an availability standpoint, especially if you can fail over traffic from one gateway to another. Also, if traffic is directed to the closest of several Internet gateways, the amount of traffic is minimized on the Intel internal network. It is also easier to deal with traffic growth when there are several Internet gateways. If one of the gateways becomes congested, only the users of that gateway are affected. Moreover, traffic can be moved to different gateways making it unnecessary to upgrade gateway resources.

For Intel, the availability requirements and the need to keep Internet traffic off of the internal network drove us to have distributed Internet gateways. Since gateways are not free, we decided to have Internet gateways only at major Intel sites—sites with thousands of employees. This minimizes traffic on our internal network while keeping the number of gateways manageable. Two key concerns with having multiple gateways are manageability and security. These are discussed later.

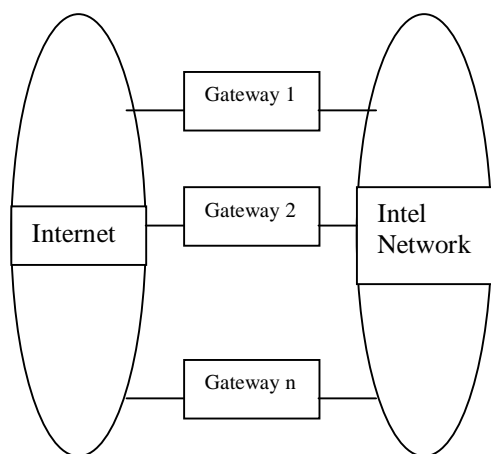
*ISP Connectivity—one vs. many.* Another design choice was ISP connectivity. The tradeoffs between having more than one ISP at all the gateways are between manageability, performance, and availability. Having one ISP is clearly easier to manage, but having only one at all of our gateways leaves us vulnerable to a problem at that

ISP that propagates across the entire Internet and brings all gateways down. We have seen this kind of problem occur on a number of occasions. Having multiple ISPs is harder to manage from a troubleshooting and a vendor perspective, but it provides higher availability. Multiple ISPs perform better because a packet to and from our gateways could travel between Intel and other Internet sites more often without having to transit a peering point. Consequently, we decided on two ISPs. More would provide better performance, but two are manageable and cost effective and still meet performance and availability goals.

*Network layout.* The network layout has to be secure, scalable, highly available, and perform well. We needed a network that would still work and still be secure if a single component failed. To achieve that goal, we made each of our Internet gateways capable of handling traffic for another gateway. To implement defense in depth, we used the screened subnet firewall design [8]. To make sure that the design was scalable, we designed all of our ISPs and firewall complexes to land on a specific Ethernet segment so that additions and changes would have minimal impact.

### Internet Connectivity Architecture

Let's look at the network layout in more detail. At the highest level of abstraction, our Internet connectivity architecture looks like that shown in Figure 1:

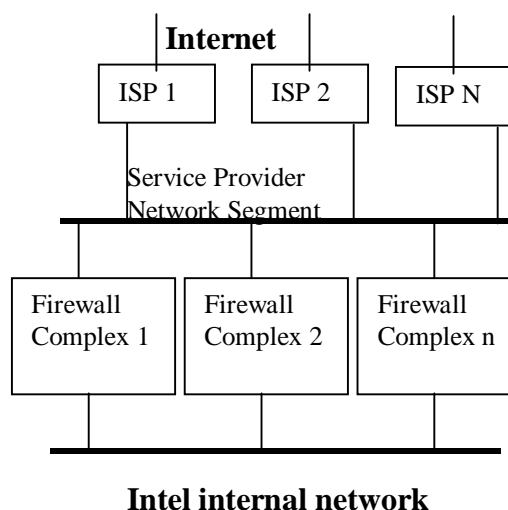


**Figure 1: Gateway connectivity between Intel and the Internet**

There are a number of Internet gateways between Intel and the Internet, each located at a major Intel facility. Systems within Intel are configured to use the Internet gateway closest to them (typically on the same site) when

they want to utilize services such as Web access. Going to the closest gateway minimizes the impact that Internet traffic has on Intel's internal network. Also, if one gateway goes down, traffic can be automatically rerouted to another gateway. This feature increases the availability of Internet service. If necessary, the architecture can be scaled by either adding gateways or increasing the bandwidth between a gateway and the Internet.

At the next level of abstraction, each gateway looks like that shown in Figure 2:



**Figure 2: Architecture of an individual gateway**

Each gateway consists of one or more firewall complexes connected to a service provider segment. Firewall complexes are groups of systems that have some common purpose, such as Intel's Web site or systems that provide basic Internet services. Most gateways will have one complex just for basic services, while some gateways at Intel have as many as four firewall complexes.

Two or more ISPs connect to this segment. This network layout has a number of advantages. With traffic flowing through the system, an ISP can be added, deleted, or upgraded. For ISPs that manage the router on their customer's premises, the service provider makes an excellent point of demarcation between the ISP and Intel. Also, each firewall complex can have its own individual routing policies with the ISPs. This allows us to tune performance for a firewall complex by adjusting routing.

This design can be scaled in a number of ways. The bandwidth can be increased by bringing in an ISP in parallel to the current connection and then cutting over.

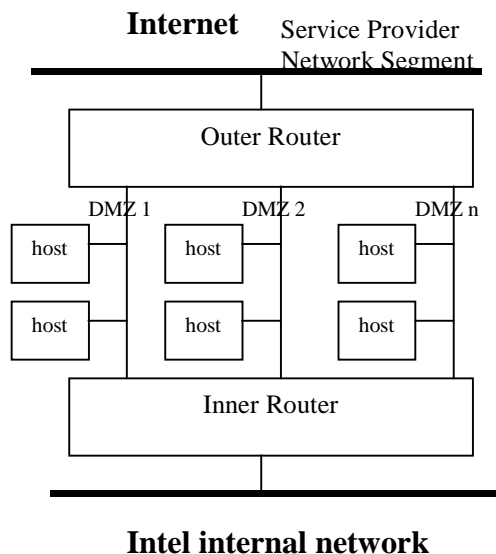
Additional ISPs can be brought in if necessary or the number of firewall complexes can be increased.

This design contains a number of high availability features. If any single ISP goes down, traffic can be rerouted through another ISP. If the service provider segment goes down, then traffic can be routed through Intel to another Internet gateway. Firewall complexes are independent of each other. If one complex has a problem, that problem is usually isolated to that complex.

### Security Architecture of a Firewall Complex

At the next level of abstraction is a firewall complex. We used a "defense in depth" approach in designing the architecture of our firewall complexes. Defense in depth relies heavily on the use of multiple components in the firewall to reduce the risk of an intrusion. An attacker must compromise more than one firewall component before gaining access to Intel's network—there is no single point of failure. While this approach does not guarantee the security of our internal network, it definitely makes it harder and more time consuming for an attacker.

A firewall complex is designed using the screened subnet architecture [8] shown in Figure 3.



**Figure 3: Basic design of a firewall complex**

The firewall components are the outer router, the Demilitarized Zones (DMZs) and the inner router. The outer and inner routers are responsible for controlling both incoming and outgoing traffic going to the various DMZ network segments. The outer router allows communication between the Internet and the hosts in the DMZ while blocking direct communication with Intel's internal systems. In addition, the outer router blocks

specific well known attacks. The inner router heavily filters communication between Intel's internal systems and the DMZ servers. Access by DMZ systems to Intel's internal network is restricted because the DMZ systems are exposed to attack from the Internet and cannot be trusted.

The method used by the routers to control traffic is called packet filtering. Each packet received by the router is compared against a set of predefined rules. These rules or access lists determine whether or not each packet is allowed to pass through the firewall and continue its journey towards the destination system. An access list consists of a source IP address a destination IP address, and a port number that denotes the service being requested.

The systems that sit in the DMZ are called bastion hosts. Each bastion host performs a specific set of functions. For example, one server provides proxy services, while another server acts as an SMTP mail relay and DNS server. This separation of services limits the damage that can be caused by a break-in, since the SMTP/DNS server does not run the proxy services and vice versa. In addition, the underlying operating system on all of the bastion hosts has been made more secure: unnecessary services have either been disabled or removed. Special programs have been installed that limit access or provide additional security. For example, the "sudo" program defines which administrators can execute privileged commands, and the Secure Shell program provides administrators with an encrypted tunnel so that an intruder cannot pick up passwords. We also run a program daily that compares the files on each of the bastion hosts with a set of master files. Differences are noted and sent to our system administrators for further investigation.

This design can be scaled in a number of ways. To deal with increased demand, more servers can be added to a DMZ segment. If more specialized services are needed, such as streaming media or authentication services, DMZ segments can be added.

### Services Architecture

Initially, the only service provided by Intel's Internet connection was electronic mail for Intel employees. Today, Intel's Internet connections provide services for not only Intel employees, but also for Intel customers and business associates. These services allow employee access to the World Wide Web, FTP, Usenet news, and streaming audio and video. Intel customers can access Intel's Web site and can download product specifications and software. Business associates can place orders for products on our e-Commerce servers. Understandably,

the complexity of the infrastructure to support these additional services has grown.

A single mail relay server was Intel's first general purpose presence on the Internet. Later, to provide additional services such as FTP and telnet to remote hosts, several more computers were granted access to the Internet connection. Users who logged into these servers could freely access all the Internet had to offer. But even before the explosive growth of the Internet brought about by the World Wide Web, this service model did not scale well. Providing an account for Internet access to every employee is clearly not scalable.

To deal with this problem and to avoid incidents where a user might mistakenly or maliciously subvert the security of an Internet access server, the user accounts were removed and replaced with several special software agents called proxies. Each proxy provides access to a specific type of Internet service. Intel employees can access the Internet by employing these proxies. Only specialized servers within DMZs can talk directly to the Internet. This improves security and it also minimizes traffic on Intel's internal network. DMZ servers such as Web servers and DNS servers that provide services to people on the Internet talk directly to the Internet without hitting Intel's internal networks. For certain services such as e-mail and DNS, we spread servers over multiple sites and gateways. This spreads out the load and makes the services available in case entire Internet gateways go down.

### **Maintaining Multiple Distributed Internet Gateways**

Distributed Internet gateways make their management a key challenge. Having Internet gateways located throughout the world, each with its own complex configurations, posed a significant maintenance challenge to the small staff responsible for managing Intel's Internet gateways. Configuring each system by hand one at a time would be time consuming and prone to error. Because break-ins are often carried out by crackers who exploit misconfigured devices, system and network administrators need to make sure that configurations are consistent and correct. Responding to changes in the environment would be increasingly difficult if we had to manage each system individually. We needed to create a system to help us manage all our firewall devices. Therefore, we designed, built, and currently use a system and methodologies that manage all of our router access control lists and all of our bastion host configurations. Our techniques, discussed in the following sections, greatly reduce the likelihood of a misconfiguration.

### **Copy Exactly!**

The first key method we use was taken from Intel's manufacturing sector. "Copy Exactly!" (CE!) [2] means that each gateway is copied exactly from the previous one as much as absolutely possible. We specifically wanted to avoid engineering a solution for each gateway. In manufacturing, new factories ramp faster and more productively if a standard design is copied exactly rather than reengineered. CE! has proven to be the only way to ensure that our configurations are implemented consistently across systems. With regard to the hardware, each firewall is composed of identical sets of routers. Each bastion host is based on a small set of approved platforms. Our maintenance system ensures that the software configurations are all CE!

CE! posed an interesting challenge for writing software and configurations that are distributed into our architecture. The code on each of the Internet gateways has to be identical; yet, that code would have to operate on systems particular to each gateway. To deal with this problem, we developed a program called *which\_dmz* that tells the code what gateway it is in and can return data specific to that gateway. If a code needs specific gateway information it can call *which\_dmz*. This way, we can distribute identical code and scripts to all gateways and have the code use gateway-specific information.

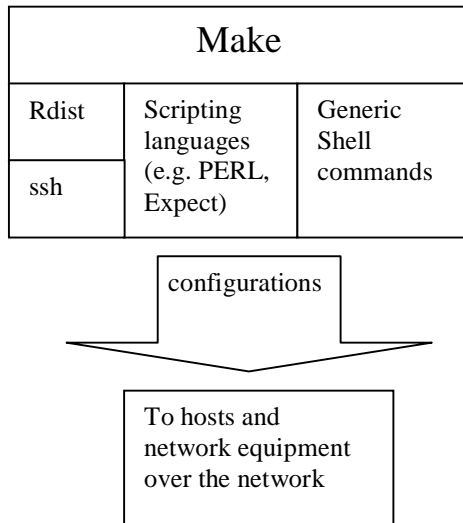
### **Modelling the Distributed Gateways as A Single System**

We model all of our Internet gateways as a single large system. In our model, all the configurations and software of the gateway components are considered source code. This code is compiled into an "executable" form that is then loaded into the system and run. Certain configuration files have dependencies: if particular configuration files change, then other files must change also. Just as programmers use *Make* [9] to manage compilation and installation of source code, we use *Make* to drive and manage the configuration of our connectivity architecture [4].

A single master set of host and network configurations is kept in a central repository. From this set, the configuration files for all the devices, whether hosts or routers, can be derived. Once a new configuration has been created, it can be "beta tested" by pushing out the changes to a single device. Once the configuration is tested, the remaining servers can be updated. As an added benefit, we can compare the configuration of each device in the firewall against the central copy to see if it has been tampered with (which may indicate a firewall device has been broken into). The central computer makes use of encryption and authentication to gain access to each device during maintenance. This prevents a

hacker from gaining access by impersonating the central computer or by eavesdropping on passwords that may be used in an update process.

The architecture of the *Make*-driven update process is shown in Figure 4.



**Figure 4: Architecture of the *Make*-driven update process**

*Make* controls all of the operations of configuration maintenance. It can activate *rdist*, a file distribution program, run programs written in various scripting languages, or execute any generic shell command to build and distribute the proper configurations and software, which are sent out over the network to network equipment and server hosts. *rdist* runs over SSH, the secure shell protocol. We discuss how *Make* is used to configure network equipment and systems in the next two sections.

### Maintaining Router Configurations

When Intel's Internet connectivity was just a single ISP connection through a single set of routers, router maintenance was simple. Once we began to have multiple gateways and once we also began to expect the traffic to fail over from one Internet gateway to another, we realized we would have to synchronize the router access lists that control what kind of packets can pass through our gateways. A change in the permissions at one gateway needed to propagate to other gateways. Since we stored access list entries in files, we needed to have a change in one file trigger changes in other configuration files. Since *Make* is excellent at managing the relationships and dependencies between files, it seemed to be a natural choice to manage the collection of router access lists.

Any gateway has to fail over to any other gateway. In order to do this, we use variables for the access control list (ACL) numbers that identify what access list an individual access list entry is part of. Macro definitions are used to define these variables to substitute the correct ACL number for the appropriate router. In this fashion, an access list entry on one router could appear in a different access list on another in such a way that traffic that regularly went out of one router could pass through another. Installation of access lists is driven through a router access list Makefile. *Make* is used to call Expect scripts that automatically load new versions of access lists.

### Managing System Configurations

*Make* is used to manage the system configurations of all the servers, wherever they may be in the world. Since *Make* is so versatile, we use it to drive the remote distribution program *rdist*. In turn, *rdist* has been configured to run over Secure Shell to prevent eavesdropping by potential crackers and to prevent attacks where intruders masquerade as friendly systems in order to penetrate defenses.

Our key system configurations are organized into separate source trees, each with its own Makefile, the configuration for *make*, and its own Distfile, the configuration for *rdist*. There are separate source trees, containing all the executables and configurations necessary for a given server subsystem like the operating system, our IMCS software, sendmail, and other significant collections of software.

Once new software is installed into a tree, installing it on all of the servers is as easy as typing "make install." Typing "make check" shows what's waiting to be installed. Every morning we receive a report from a script that does a "make check" in each tree. This report is an extremely useful tool for keeping track of changes that are being pilot tested on one machine and will need to be installed on the rest of the machines. "make check" reports on what files have changed and what needs to be done in order to synchronize a given system with the definitive build.

By using *Make* we can easily insert hooks to massage or generate certain data files before they are distributed to the servers. For example, the sendmail tree's Makefile can automatically generate a sendmail.cw file that contains a list of all of the DNS domain names from the DNS tree.

By using *rdist*, we can easily customize which files get installed where and on which servers. We use *rdist* to automatically run commands when certain files are installed or updated. For example, when the mail alias

file on a server is updated, a program called "newaliases" needs to be run to rebuild the file into an indexed file format. *rdist* can automatically run the "newaliases" command once an alias file is updated.

Our use of *Make* and *rdist* for server management closely parallels the "Copy Exactly!" methodology discussed earlier.

## MEASURING THE QUALITY OF INTERNET ACCESS

With such a significant investment in Internet connectivity, we needed to be able to measure its quality. We therefore developed the Internet Measurement and Control System (IMCS) [3]. This system measures key indices of Internet performance such as packet loss, packet delay, Web page download rate and Web page error retrieval rates, and raw traffic volumes. When these key indices exceed predefined limits, the Network Operation Center (NOC) personnel are alerted and start working through a script to debug the potential problem. We modelled this system on the statistical process control systems used in Intel manufacturing plants, where when certain metrics are out of control, action is taken.

Early in the design of our measurement systems for the Intel firewalls, we decided that the most useful architecture was a distributed one that allowed measurement systems (such as IMCS and public domain systems such as MRTG [10]) to collect their data on a system in the DMZ and publish the results as HTML pages on Web servers on the measurement system. By using Perl-based CGI scripts on the measurement systems, it was possible to drill down to individual measurement values using GET-mode URLs. The fact that the measurements were available on the Web allowed us to make some simple Perl automation scripts that would routinely fetch current measurement thumbnail graphs from all our measurement systems and display them on a browser anywhere inside of Intel. The side-by-side comparison of key metrics across firewall complexes allowed quick isolation of common faults to individual firewalls, or conversely, allowed us to see Internet-wide events in near real-time.

The common tools (ping, traceroute, whois, etc.) and controlled access to the tools' current state of interfaces on firewall routers are made available by a combination of JavaScript-enabled HTML pages and CGI scripts written in Perl and Expect. This allows us to allow NOC personnel access to tools without giving them direct access to network equipment, thus avoiding potential security exposures.

## INTERNET USAGE POLICIES

We felt that we could not offer Internet access unless some guidelines covering behavior on the Internet were in place. We examined many Internet guides and wrote up a set of guidelines, which became the Intel Internet guidelines from 1993 to 1996 [11].

In 1996, Intel decided they needed one comprehensive guideline to cover employees' use of electronic mail, computers in general, and the Internet. Several of the Internet connectivity staff, as well as representatives from marketing and legal drafted the current set of guidelines in use today.

## RESULTS

Our experiences with the connectivity architecture has generally been positive. In this section, we discuss how well the architecture has scaled, describe some spinoffs of the technology and the policies employed to maintain the architecture, and then outline some of the key challenges we have faced.

### Scaling to Meet Demand

Our gateway architecture is quite scalable. From a service standpoint, it provides Internet services for more than 60,000 employees. The architecture is robust enough to support \$1 billion per month in electronic commerce. We have scaled the components of the architecture in almost every possible direction. To deal with increasing use of services in an Internet gateway, we have increased the number of servers. To deal with different services in a firewall complex, we have increased the number of DMZs and created new firewall complexes. To deal with additional performance requirements, we added additional ISPs and additional bandwidth. To deal with acquisitions and new users, we even added additional Internet gateways.

Despite this growth just mentioned, our architecture is maintained by a small staff. At the present moment, a staff of five maintains the configurations of six Internet gateways and ten firewall complexes, which includes more than 30 routers and about 30 servers around the world. We are able to maintain thousands of lines of router access list entries. Copy Exactly!, modelling Internet connectivity at Intel as a single system, and maintaining configurations using *Make* has made this possible. The availability of the design makes the architecture functional despite the failures of individual components.

### Connectivity Architecture, Tools, and Policy

The architecture, tools, and policies developed for Intel's Internet connectivity have found other uses. The

connectivity architecture for Intel's use of the Internet has been adapted into the first two data centers of Intel's new Web hosting business, Intel® Online Services (IOS). Many of the tools originating from within Intel's IT have also been implemented within IOS, including the Make update methodology and IMCS. Our experience with measuring Internet performance lead us to contribute to the Cross Industry Working Team (XIWT) white paper on measuring Internet performance [6]. XIWT is a consortium of a diverse group of industries working to improve information infrastructure.

The Internet usage policy that we described in [11] is one of the more interesting spinoffs of our work. The paper became widely distributed in a number of collections of Internet usage and security policies. At about the same time that the paper was published, the User Services Area of the Internet Engineering Task Force began looking for people and material to create a set of Internet usage guidelines to publish as an informational RFC. The work we did on Internet usage policy became part of RFC 1855 [5], Netiquette Guidelines, published in 1995.

## KEY CHALLENGES

We experienced a number of key challenges as we implemented our Internet connectivity architecture. Our make-driven maintenance process, while powerful, requires tremendous discipline to use. While software and configuration changes can be rolled out uniformly across all of the Internet gateways, it is just as easy to roll out a bad configuration change as a good one. It also takes discipline to maintain Copy Exactly! and distribute changes from the central repository. Occasionally configuration changes are made on individual servers and routers and then lost when other changes are distributed from the central configuration repository.

While our architecture is designed to withstand the failure of a single physical component, the failure of a software component is much harder to deal with. Because of CE!, we deploy the same version of software and configuration files to each type of network equipment and server. If there is a failure in that software, then all the equipment with that version fails. We have seen this happen most often when there are scaling issues—where software or configurations cannot handle the demands made on them. The way to avoid this problem is to test new software or configurations under significant load. Unfortunately, not all load conditions can be simulated in tests.

Security continues to be a challenge. New threats and vulnerabilities crop up regularly and must be dealt with. Among the more difficult challenges are applications that users want to utilize across the Internet that are not well designed for firewalled environments. We have had to

refuse a number of requests for services that cannot be easily proxied or do significant engineering to make those services secure.

Scaling to meet demand for Internet services continues to be a challenge. As the Internet becomes used more and more, servers and network equipment can become heavily taxed. Dealing with large numbers of firewall rule requests is also very taxing. Another demanding aspect is that changes or major projects often happen on short notice (also known as Internet time).

Our IMCS system relies heavily on active measurements, i.e., measurements that generate nonvalue-added traffic on the Internet and our infrastructure in order to obtain performance data. That nonvalue-added traffic causes a number of problems. We have received some complaints from the targets of our measurements about the measurements that we do. In one case, our measurements added significantly to a target site's bandwidth costs, prompting them to complain vigorously. We would like to measure everything we can. However, active measurements add traffic and thus costs to our infrastructure and to the infrastructure of the targets that we measure.

International Internet gateways are another challenge. Bandwidth costs are often much greater internationally than in the US, so it is not always clear that installation of an Internet gateway will reduce costs and provide better end-user performance. As noted by Cukier [7], the performance going from a country to the United States can often be better and cheaper than performance within a region such as Europe or Asia. Since more bandwidth is being put between the US and Asia and the US and Europe than is being installed locally within those regions, this trend is likely to continue, making it difficult to justify the cost of international Internet gateways.

## FUTURE PLANS AND EXPECTED TRENDS

In general, we see ever increasing use of the Internet for more and more functions. Two particular trends we see are the use of Virtual Private Networking and streaming media.

Intel currently utilizes high-cost, high-maintenance legacy technology to provide connectivity between our sites, to employees when they are away from the office, and to our business associates. A new connectivity model, called Virtual Private Networking (VPN), can be utilized to lower costs and to reduce the time required to establish a connection to a new site or business associate.

With VPN, we are able to utilize the Internet bandwidth to enable employees who are out of the office to connect back in to Intel and have access to our internal resources

using their own ISPs. ISPs typically offer much wider and cheaper coverage for dial-up connectivity than Intel. Using the triple DES encryption technology provided by protocols such as IPSEC, we can ensure that Intel's intellectual property is safe as it traverses the Internet.

The majority of Intel's Wide Area Network is composed of costly privately leased lines. When we need to connect to a new site, we are often delayed due to the time required to order and install these circuits. Using VPN technologies, we will also be able to leverage the Internet connectivity that is already installed at our major sites to create an encrypted, secure link between them through the Internet. We will be able to reduce the setup time of two to three months to one day or in some cases the connection can be established within hours of receiving the request.

Connecting business associates presents a slightly different set of requirements because we only want to share a subset of information. By building an environment that limits accessibility to specific resources, we can utilize VPN technology to reduce the cost and time required to connect and implement the connection in a similar fashion to WAN replacement technology mentioned above.

We also see increasing use of streaming media such as video and audio. These type of media will substantially increase bandwidth requirements. Because these media are jitter sensitive, service-level agreements between Intel and its ISP become more and more critical.

We see passive measurements becoming much more important in the future. Mining data sources such as Web server logs and router flow statistics can yield valuable performance data without adding probe traffic. As the number of gateways increases, active measurements become more and more of a burden on the measured sites.

## CONCLUSION

As the Internet evolved from a network linking a community of researchers and engineers to the mainstream medium that it is today, Intel's Internet connectivity evolved from a modem used for e-mail to the highly available, multiple service distributed system that it is today. Our design and implementation of an Internet connectivity architecture has enabled Intel to handle \$1 Billion in e-Commerce per month and has spun off technology, methods, and policies used in other Intel businesses and across the Internet. Anticipating ever increasing use of the Internet, our architecture is ready to deal with new challenges.

## ACKNOWLEDGMENTS

We acknowledge Suzanne Johnson, whose vision helped land our first Internet dial-up connection those many years ago. We also acknowledge and thank the many people who have helped so much with deploying and maintaining the architecture, including the CPS team, ITSP, the OSC, the WAN team, and countless folks in site IT organizations.

## REFERENCES

- [1] Johnson, Suzanne M., "*Internet Affects the Corporation: Experiences from Eight Years of Connectivity*," INET 95, Honolulu, HI, June 1995.
- [2] McDonald, Chris J., "*The Evolution of Intel's Copy EXACTLY! Technology Transfer Method*," Intel Technology Journal, [http://developer.intel.com/technology/itj/q41998/article/Oace, Tod, Sedayao, Jeff, Wong, Clinton, 'Don't Just es/art\\_2.htm](http://developer.intel.com/technology/itj/q41998/article/Oace, Tod, Sedayao, Jeff, Wong, Clinton, 'Don't Just es/art_2.htm).
- [3] Bickerstaff, Cindy, True, Ken, Smothers, Charles, "*Talk About The Weather - Manage it! A System for Measuring, Monitoring, and Managing Internet Performance and Connectivity*," 1<sup>st</sup> Usenix Conference on Network Administration, Santa Clara, CA, April 1999.
- [4] Hambridge, Sally L., Oace, Tod, Sedayao, Jeff, and Smothers, Charles, "*Just Type Make! Managing Firewall Using Make and Other Publicly Available Utilities*," 1<sup>st</sup> Usenix Conference on Network Administration, Santa Clara, CA, April 1999.
- [5] Hambridge, Sally. "Netiquette Guidelines," RFC 1855, October, 1995.
- [6] Cross Industries Working Team White Paper, "Customer View of Internet Service Performance: Method, Methodology, and Metrics," "[Customer View of Internet Service Performance: Measurement Methodology and Metrics](#)," September 1998.
- [7] Cukier, Kenneth Neil, "*Bandwidth Colonialism? The Implications of Internet Infrastructure on International E-Commerce*," INET 99, San Jose, CA, June 1999.
- [8] Chapman, Brent D. and Zwicky, Elizabeth D., "*Building Internet Firewalls*," O'Reilly & Associates, Inc., Sebastopol, CA, pp. 58, 66.
- [9] Oram, Andrew and Talbot, Steve, "*Managing Projects with Make*," O'Reilly and Associates, Inc., Sebastopol, CA 1991.
- [10] Oetiker, Tobias. <http://www.mrtg.org/>.

[11] Hambridge, Sally and Sedayao, Jeff, "*Horses and Barn Doors: Evolution of Corporate Guidelines for Internet Usage*," Proceedings of the Seventh Systems Administration Conference, LISA '93, Monterey, CA, November 1993.

## AUTHORS' BIOGRAPHIES

**Cindy Bickerstaff** is a 20-year Intel veteran with the first 15 in silicon process development especially in metrology and process control. She received "The Five Year (or so) SPC Marathon Award" in 1988 for driving SPC implementation in California Technology Development. She received an Intel Achievement Award in 1989 for "formulating, developing, and implementing the first statistically based process control strategy for lithography at Intel". After her 10th silicon technology development cycle she decided to try something new and moved into the Internet group in 1995. Since then, Cindy has been mapping many of the measurement and control methods she worked on in silicon processing into the Internet knowledge domain. Her email address is [cindy.bickerstaff@intel.com](mailto:cindy.bickerstaff@intel.com).

**Sally Hambridge** began working on the Internet in 1980 when she joined the Information Sciences Institute in Marina del Rey, the home of the IANA and the RFC-Editor. She joined Intel in 1984 after a brief sojourn at Atari. She saw the advent of Internet connectivity at Intel, and has worked in the Internet group since 1992. Since 1996 she has been responsible for router access control lists and for routing in the Internet connections. She joined Intel Online® Services in August 1999. Her email address is [sally.l.hambridge@intel.com](mailto:sally.l.hambridge@intel.com).

**Lynne Marchi** received her B.S. degree in computer science from California State University, Sacramento. She began work at Intel in 1992 as a co-op and then joined the Corporate Information Security group in 1993. In 1996, she joined Internet Connectivity Engineering where she has focused on the secure implementation of new firewalls. Her email address is [lynne\\_c\\_marchi@intel.com](mailto:lynne_c_marchi@intel.com)

**Tod Oace** is a UNIX and networking systems engineer. Over the past 20 years he has worked with a wide range of computer platforms from micro to mainframe, and is proficient in several programming languages and networking protocols. He pioneered the use of BSD UNIX for several enterprise applications since joining Intel in 1991. He received division recognition awards for the redesign and support of an in-house created Intranet SMTP/cc:Mail gateway, and for the co-creation of an Intranet LED readerboard display control system. He also designed the first large-scale network for the [www.intel.com](http://www.intel.com) server complex. Tod is now working in

the Firewall Engineering group of Intel Online Services. His email address is [tod.r.oace@intel.com](mailto:tod.r.oace@intel.com).

**Stacy Purcell** graduated from the Georgia Institute of Technology with a B.S. degree in computer science. He joined IT in 1994 where he has worked as a Front Line Manager, a Network Engineer specializing in the design, maintenance, and troubleshooting of LANs for the Folsom site and in new acquisitions. He currently works in the ICE team focusing on Intel's six Internet firewalls and plans to expand that number to 24 by the end of the year. His email address is [stacy.p.purcell@intel.com](mailto:stacy.p.purcell@intel.com).

**Jeff Sedayao** is a network engineer in Intel Online Services. Between 1987 and 1999, he architected and ran Intel's Internet connectivity. His primary interests are in Internet performance, security, and policy implementation. He also serves as Intel's representative to the Cross Industries Working Team's Internet Performance Team and has participated in the IETF's IP Performance Metrics Workgroup. His email address is [jeff.sedayao@intel.com](mailto:jeff.sedayao@intel.com).

**Charles Smothers** is a Senior Systems Programmer for Information Technology. He joined Intel in 1982, when he wrote 8085 assembly code for testing the 27128 EPROM. With the Memory Components Division, Low Yield Analysis team he automated several of the tasks used in the visual inspection and defect categorization process. In 1990, he joined the MCD Design Engineering group as a UNIX systems administrator. Today, he specializes in Internet Proxy servers. His email address is [charles.smothers@intel.com](mailto:charles.smothers@intel.com).

**Ken True** is Manager of Internet Firewall Engineering for Intel Online Services, Inc. He joined Intel in 1981 and has provided technical management and individual contributions in the areas of Mainframe Systems Programming, WAN Network Engineering, PC Technical Services, LAN Software Distribution (OfficeLAN), Intranet/Internet Engineering, and Internet Connectivity Engineering. He and his organization are responsible for the architecture, engineering, and security of the IOS firewall systems worldwide. His email address is [ken.true@intel.com](mailto:ken.true@intel.com).

Copyright © Intel Corporation 2000. Legal notices at <http://www.intel.com/tradmarx.htm>.