The Internet Protocol Journal

December 2012 Volume 15, Number 4

A Quarterly Technical Publication for Internet and Intranet Professionals

In This Issue

From The Editor

Accurate timekeeping has long been an engineering challenge if not obsession in some circles. Take for example the iconic Swiss *chronometer* watch or the pendulum-controlled clock mechanism in London's Palace of Westminster, often referred to as "Big Ben." Such mechanical systems—accurate as they may be—are no match for the clocks we use in telecommunication and computer networks. In our last issue, Geoff Huston described the glitches encountered last June when a *Leap Second* was applied to *Coordinated Universal Time* (UTC). In this issue he explains the operation of the *Network Time Protocol* (NTP). The article is another installment in our series "Protocol Basics."

It is difficult to believe that it has been more than 25 years since the first publication of Douglas Comer's book series *Internetworking With TCP/IP*. Volume 1 of this series will soon be available in its sixth edition, and we asked the author to write an article about *Packet Classification* based on material in the book.

The recent *World Conference on International Telecommunications* (WCIT) did not have the outcome with respect to the Internet that many had hoped for. We plan to publish an analysis of this event in our next issue. This time—in our "Fragments" section—we have some reactions from the *Number Resource Organization* (NRO) and the Internet Society, as well as pointers to further information about WCIT.

January 1, 2013, marked the 30th anniversary of the *Transmission Control Protocol/Internet Protocol* (TCP/IP). A transition from the earlier *Network Control Program* (NCP) took place on January 1, 1983, also known as "Flag Day." Such an instant technology change would have been desirable for the transition from IPv4 to IPv6, but sadly this isn't possible. Instead we are happy to honor those who dedicate their careers to IPv6 deployment with an *Itojun Service Award*. See page 25 for more details.

On page 30 you will find some frequently asked questions about subscriptions to this journal. If you have other questions or comments, please contact us at `ipj@cisco.com`

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

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

ISSN 1944-1134

Protocol Basics: The Network Time Protocol

by Geoff Huston, APNIC

Back at the end of June 2012[0] there was a brief IT hiccup as the world adjusted the *Coordinated Universal Time* (UTC) standard by adding an extra second to the last minute of the 31st of June. Normally such an adjustment would pass unnoticed by all but a small dedicated collection of time keepers, but this time the story spread out into the popular media as numerous Linux systems hiccupped over this additional second, and they supported some high-profile services, including a major air carrier's reservation and ticketing backend system. The entire topic of time, time standards, and the difficulty of keeping a highly stable and regular clock standard in sync with a slightly wobbly rotating Earth has been a longstanding debate in the *International Telecommunication Union Radiocommunication Sector* (ITU-R) standards body that oversees this coordinated time standard. However, I am not sure that anyone would argue that the challenges of synchronizing a strict time signal with a less than perfectly rotating planet is sufficient reason to discard the concept of a coordinated time standard and just let each computer system drift away on its own concept of time. These days we have become used to a world that operates on a consistent time standard, and we have become used to our computers operating at sub-second accuracy. But how do they do so? In this article I will look at how a consistent time standard is spread across the Internet, and examine the operation of the *Network Time Protocol* (NTP).

Some communications protocols in the IP protocol suite are quite recent, whereas others have a long and rich history that extends back to the start of the Internet. The ARPANET switched over to use the TCP/IP protocol suite in January 1983, and by 1985 NTP was in operation on the network. Indeed it has been asserted that NTP is the longest running, continuously operating, distributed application on the Internet[1].

The objective of NTP is simple: to allow a client to synchronize its clock with UTC time, and to do so with a high degree of accuracy and a high degree of stability. Within the scope of a WAN, NTP will provide an accuracy of small numbers of milliseconds. As the network scope gets finer, the accuracy of NTP can increase, allowing for sub-millisecond accuracy on LANs and sub-microsecond accuracy when using a precision time source such as a *Global Positioning System* (GPS) receiver or a caesium oscillator.

If a collection of clients all use NTP, then this set of clients can operate with a synchronized clock signal. A shared data model, where the modification time of the data is of critical importance, is one example of the use of NTP in a networked context.

(I have relied on NTP timer accuracy at the microsecond level when trying to combine numerous discrete data sources, such as a web log on a server combined with a *Domain Name System* (DNS) query log from DNS resolvers and a packet trace.)

NTP, Time, and Timekeeping

To consider NTP, it is necessary to consider the topic of timekeeping itself. It is useful to introduce some timekeeping terms at this juncture:

Stability	How well a clock can maintain a constant frequency
Accuracy	How well the frequency and absolute value of the clock compares with a standard reference time
Precision	How well the accuracy of a clock can be maintained within a particular timekeeping system
Offset	The time difference in the absolute time of two clocks
Skew	The variation of offset over time (first-order derivative of offset over time)
Drift	The variation of skew over time (second-order derivative of offset over time)

NTP is designed to allow a computer to be aware of three critical metrics for timekeeping: the *offset* of the local clock to a selected reference clock, the *round-trip delay* of the network path between the local computer and the selected reference clock server, and the *dispersion* of the local clock, which is a measure of the maximum error of the local clock relative to the reference clock. Each of these components is maintained separately in NTP. They provide not only precision measurements of offset and delay, to allow the local clock to be adjusted to synchronize with a reference clock signal, but also definitive maximum error bounds of the synchronization process, so that the user interface can determine not only the time, but the quality of the time as well.

Universal Time Standards

It would be reasonable to expect that the time is just the time, but that is not the case. The Universal Time reference standard has several versions, but these two standards are of interest to network timekeeping.

UT1 is the principal form of Universal Time. Although conceptually it is *Mean Solar Time* at 0° longitude, precise measurements of the Sun are difficult. Hence, it is computed from observations of distant quasars using long baseline interferometry, laser ranging of the Moon and artificial satellites, as well as the determination of GPS satellite orbits. UT1 is the same everywhere on Earth, and is proportional to the rotation angle of the Earth with respect to distant quasars, specifically the *International Celestial Reference Frame* (ICRF), neglecting some small adjustments.

The observations allow the determination of a measure of the Earth's angle with respect to the ICRF, called the *Earth Rotation Angle* (ERA), which serves as a modern replacement for *Greenwich Mean Sidereal Time*). UT1 is required to follow the relationship

$$\text{ERA} = 2\pi(0.7790572732640 + 1.00273781191135448\,\text{Tu}) \text{ radians}$$
$$\text{where Tu} = (\text{Julian UT1 date} - 2451545.0)$$

Coordinated Universal Time (UTC) is an atomic timescale that approximates UT1. It is the international standard on which civil time is based. It ticks SI seconds, in step with *International Atomic Time* (TAI). It usually has 86,400 SI seconds per day, but is kept within 0.9 seconds of UT1 by the introduction of occasional intercalary leap seconds. As of 2012 these leaps have always been positive, with a day of 86,401 seconds.[9]

NTP uses UTC, as distinct from the *Greenwich Mean Time* (GMT), as the reference clock standard. UTC uses the TAI time standard, based on the measurement of 1 second as 9,192,631,770 periods of the radiation emitted by a caesium-133 atom in the transition between the two hyperfine levels of its ground state, implying that, like UTC itself, NTP has to incorporate leap second adjustments from time to time.

NTP is an "absolute" time protocol, so that local time zones—and conversion of the absolute time to a calendar date and time with reference to a particular location on the Earth's surface—are not an intrinsic part of the NTP protocol. This conversion from UTC to the wall-clock time, namely the local date and time, is left to the local host.

Servers and Clients

NTP uses the concepts of *server* and *client*. A server is a source of time information, and a client is a system that is attempting to synchronize its clock to a server.

Servers can be either a *primary server* or a *secondary server.* A primary server (sometimes also referred to as a *stratum 1* server using terminology borrowed from the time reference architecture of the telephone network) is a server that receives a UTC time signal directly from an authoritative clock source, such as a configured atomic clock or—very commonly these days—a GPS signal source. A s*econdary server* receives its time signal from one or more *upstream servers,* and distributes its time signal to one of more *downstream servers* and *clients*. Secondary servers can be thought of as clock signal repeaters, and their role is to relieve the client query load from the primary servers while still being able to provide their clients with a clock signal of comparable quality to that of the primary servers. The secondary servers need to be arranged in a strict hierarchy in terms of upstream and downstream, and the stratum terminology is often used to assist in this process.

As noted previously, a stratum 1 server receives its time signal from a UTC reference source. A stratum 2 server receives its time signal from a stratum 1 server, a stratum 3 server from stratum 2 servers, and so on. A stratum *n* server can peer with many stratum *n* – 1 servers in order to maintain a reference clock signal. This stratum framework is used to avoid synchronization loops within a set of time servers.

Clients peer with servers in order to synchronize their internal clocks to the NTP time signal.

The NTP Protocol

At its most basic, the NTP protocol is a clock request transaction, where a client requests the current time from a server, passing its own time with the request. The server adds its time to the data packet and passes the packet back to the client. When the client receives the packet, the client can derive two essential pieces of information: the reference time at the server and the elapsed time, as measured by the local clock, for a signal to pass from the client to the server and back again. Repeated iterations of this procedure allow the local client to remove the effects of network jitter and thereby gain a stable value for the delay between the local clock and the reference clock standard at the server. This value can then be used to adjust the local clock so that it is synchronized with the server. Further iterations of this protocol exchange can allow the local client to continuously correct the local clock to address local clock skew.

NTP operates over the *User Datagram Protocol* (UDP). An NTP server listens for client NTP packets on port 123. The NTP server is stateless and responds to each received client NTP packet in a simple transactional manner by adding fields to the received packet and passing the packet back to the original sender, without reference to preceding NTP transactions.

Upon receipt of a client NTP packet, the receiver time-stamps receipt of the packet as soon as possible within the packet assembly logic of the server. The packet is then passed to the NTP server process. This process interchanges the IP Header Address and Port fields in the packet, overwrites numerous fields in the NTP packet with local clock values, time-stamps the egress of the packet, recalculates the checksum, and sends the packet back to the client.

The NTP packets sent by the client to the server and the responses from the server to the client use a common format, as shown in Figure 1.

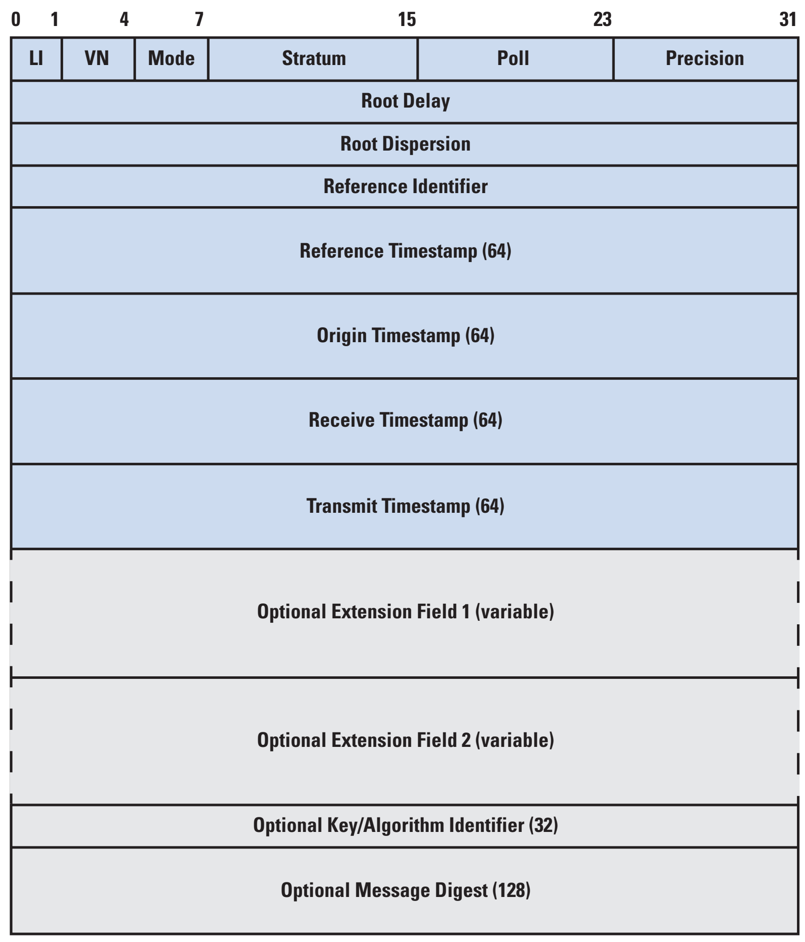


Figure 1: NTP Message Format

The header fields of the NTP message are as follows:

LI	Leap Indicator (2 bits) This field indicates whether the last minute of the current day is to have a leap second applied. The field values follow: 0: No leap second adjustment 1: Last minute of the day has 61 seconds 2: Last minute of the day has 59 seconds 3: Clock is unsynchronized
VN	NTP Version Number (3 bits) (current version is 4).

Mode	NTP packet mode (3 bits)
	The values of the Mode field follow:
	0: Reserved
	1: Symmetric active
	2: Symmetric passive
	3: Client
	4: Server
	5: Broadcast
	6: NTP control message
	7: Reserved for private use
Stratum	Stratum level of the time source (8 bits)
	The values of the Stratum field follow:
	0: Unspecified or invalid
	1: Primary server
	2–15: Secondary server
	16: Unsynchronized
	17–255: Reserved
Poll	Poll interval (8-bit signed integer)
	The $\log_2$ value of the maximum interval between successive NTP messages, in seconds.
Precision	Clock precision (8-bit signed integer)
	The precision of the system clock, in $\log_2$ seconds.
Root Delay	The total round-trip delay from the server to the primary reference sourced. The value is a 32-bit signed fixed-point number in units of seconds, with the fraction point between bits 15 and 16. This field is significant only in server messages.
Root Dispersion	The maximum error due to clock frequency tolerance. The value is a 32-bit signed fixed-point number in units of seconds, with the fraction point between bits 15 and 16. This field is significant only in server messages.
Reference Identifier	For stratum 1 servers this value is a four-character ASCII code that describes the external reference source (refer to Figure 2). For secondary servers this value is the 32-bit IPv4 address of the synchronization source, or the first 32 bits of the *Message Digest Algorithm 5* (MD5) hash of the IPv6 address of the synchronization source.

Figure 2: Reference Identifier Codes (from RFC 4330)

Code	External Reference Source
LOCL	uncalibrated local clock
CESM	calibrated Cesium clock
RBDM	calibrated Rubidium clock
PPS	calibrated quartz clock or other pulse-per-second source
IRIG	Inter-Range Instrumentation Group
ACTS	NIST telephone modem service
USNO	USNO telephone modem service
PTB	PTB (Germany) telephone modem service
TDF	Allouis (France) Radio 164 kHz
DCF	Mainflingen (Germany) Radio 77.5 kHz
MSF	Rugby (UK) Radio 60 kHz
WWV	Ft. Collins (US) Radio 2.5, 5, 10, 15, 20 MHz
WWVB	Boulder (US) Radio 60 kHz
WWVH	Kauai Hawaii (US) Radio 2.5, 5, 10, 15 MHz
CHU	Ottawa (Canada) Radio 3330, 7335, 14670 kHz
LORC	LORAN-C radionavigation system
OMEG	OMEGA radionavigation system
GPS	Global Positioning Service

The next four fields use a 64-bit time-stamp value. This value is an unsigned 32-bit seconds value, and a 32-bit fractional part. In this notation the value 2.5 would be represented by the 64-bit string:

```
0000|0000|0000|0000|0000|0000|0000|0010.|1000|0000|0000|0000|0000|0000|0000|0000
```

The unit of time is in seconds, and the epoch is 1 January 1900, meaning that the NTP time will cycle in the year 2036 (two years before the 32-bit Unix time cycle event in 2038).

The smallest time fraction that can be represented in this format is 232 picoseconds.

Reference Timestamp	This field is the time the system clock was last set or corrected, in 64-bit time-stamp format.
Originate Timestamp	This value is the time at which the request departed the client for the server, in 64-bit time-stamp format.
Receive Timestamp	This value is the time at which the client request arrived at the server in 64-bit time-stamp format.
Transmit Timestamp	This value is the time at which the server reply departed the server, in 64-bit time-stamp format.

The basic operation of the protocol is that a client sends a packet to a server and records the time the packet left the client in the *Origin Timestamp* field (T1). The server records the time the packet was received (T2). A response packet is then assembled with the original Origin Timestamp and the *Receive Timestamp* equal to the packet receive time, and then the *Transmit Timestamp* is set to the time that the message is passed back toward the client (T3). The client then records the time the packet arrived (T4), giving the client four time measurements, as shown in Figure 3.

Figure 3: NTP Transaction Timestamps (from RFC 4330)

Timestamp Name	ID	When Generated
Originate Timestamp	T1	time request sent by client
Receive Timestamp	T2	time request received by server
Transmit Timestamp	T3	time reply sent by server
Destination Timestamp	T4	time reply received by client

These four parameters are passed into the client timekeeping function to drive the clock synchronization function, which we will look at in the next section.

The optional Key and Message Digest fields allow a client and a server to share a secret 128-bit key, and use this shared secret to generate a 128-bit MD5 hash of the key and the NTP message fields. This construct allows a client to detect attempts to inject false responses from a man-in the-middle attack.

The final part of this overview of the protocol operation is the polling frequency algorithm. A NTP client will send a message at regular intervals to a NTP server. This regular interval is commonly set to be 16 seconds. If the server is unreachable, NTP will back off from this polling rate, doubling the back-off time at each unsuccessful poll attempt to a minimum poll rate of 1 poll attempt every 36 hours. When NTP is attempting to resynchronize with a server, it will increase its polling frequency and send a burst of eight packets spaced at 2-second intervals.

When the client clock is operating within a sufficient small offset from the server clock, NTP lengthens the polling interval and sends the eight-packet burst every 4 to 8 minutes (or 256 to 512 seconds).

Timekeeping on the Client

The next part of the operation of NTP is how an NTP process on a client uses the information generated by the periodic polls to a server to moderate the local clock.

From an NTP poll transaction, the client can estimate the delay between the client and the server. Using the time fields described in Figure 3, the transmission delay can be calculated as the total time from transmission of the poll to reception of the response minus the recorded time for the server to process the poll and generate a response:

$$\delta = (T4 - T1) - (T3 - T2)$$

The offset of the client clock from the server clock can also be estimated by the following:

$$\Theta = \tfrac{1}{2} [(T2 - T1) + (T3 - T4)]$$

It should be noted that this calculation assumes that the network path delay from the client to the server is the same as the path delay from the server to the client.

NTP uses the minimum of the last eight delay measurements as δ_0. The selected offset, Θ_0, is one measured at the lowest delay. The values (Θ_0,δ_0) become the NTP update value.

When a client is configured with a single server, the client clock is adjusted by a slew operation to bring the offset with the server clock to zero, as long as the server offset value is within an acceptable range.

When a client is configured with numerous servers, the client will use a selection algorithm to select the preferred server to synchronize against from among the candidate servers. Clustering of the time signals is performed to reject outlier servers, and then the algorithm selects the server with the lowest stratum with minimal offset and jitter values. The algorithm used by NTP to perform this operation is *Marzullo's Algorithm*[2].

When NTP is configured on a client, it attempts to keep the client clock synchronized against the reference time standard. To do this task NTP conventionally adjusts the local time by small offsets (larger offsets may cause side effects on running applications, as has been found when processing leap seconds). This small adjustment is undertaken by an *adjtime()* system call, which slews the clock by altering the frequency of the software clock until the time correction is achieved. Slewing the clock is a slow process for large time offsets; a typical slew rate is 0.5 ms per second.

Obviously this informal description has taken a rather complex algorithm and some rather detailed math formulas without addressing the details. If you are interested in how NTP operates at a more detailed level, consult the references that follow, which will take you far deeper into the algorithms and the underlying models of clock selection and synchronization than I have done here.

Conclusion

NTP is in essence an extremely simple stateless transaction protocol that provides a quite surprising outcome. From a regular exchange of simple clock readings between a client and a server, it is possible for the client to train its clock to maintain a high degree of precision despite the possibility of potential problems in the stability and accuracy of the local clock and despite the fact that this time synchronization is occurring over network paths that impose a noise element in the form of jitter in the packet exchange between client and server. Much of today's distributed Internet service infrastructure relies on a common time base, and this base is provided by the common use of the Network Time Protocol.

References and Further Reading

[0] Geoff Huston, "Leaping Seconds," *The Internet Protocol Journal,* Volume 15, No. 3, September 2012.

[1] David L. Mills, "A Brief History of NTP Time: Confessions of an Internet Timekeeper," ACM SIGCOMM, *Computer Communication Review,* Vol. 33, No. 2, pp. 9–12, April 2003, `http://www.eecis.udel.edu/~mills/database/papers/history.pdf`

[2] K. A. Marzullo, "Maintaining the Time in a Distributed System: An Example of a Loosely-Coupled Distributed Service," Ph.D. dissertation, Stanford University, Department of Electrical Engineering, February 1984, `http://en.wikipedia.org/wiki/Marzullo%27s_algorithm`

[3] David L. Mills, "NTP Architecture, Protocol and Algorithms," University of Delaware, `www.eecis.udel.edu/~mills/database/brief/arch/arch.ppt`

[4] Jack Burbank, William Kasch, and David Mills, "Network Time Protocol Version 4: Protocol and Algorithms Specification," RFC 5905, June 2010.

[5] David L. Mills, "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI," RFC 4330, January 2006.

[6] `http://www.ntp.org`

[7] `http://www.eecis.udel.edu/~mills/ntp.html`

[8] David Mills, *Computer Network Time Synchronization: the Network Time Protocol on Earth and in Space,* Second Edition, CRC Press, 2011.

[9] `http://en.wikipedia.org/wiki/Universal_Time`

Disclaimer

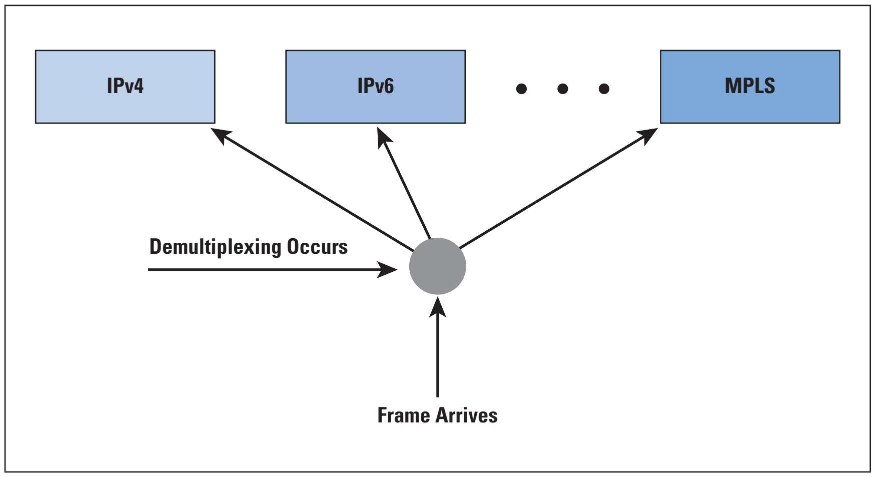
GEOFF HUSTON, B.Sc., M.Sc., is the Chief Scientist at APNIC, the Regional Internet Registry serving the Asia Pacific region. He has been closely involved with the development of the Internet for many years, particularly within Australia, where he was responsible for the initial build of the Internet within the Australian academic and research sector. He is author of numerous Internet-related books, and was a member of the Internet Architecture Board from 1999 until 2005. He served on the Board of Trustees of the Internet Society from 1992 until 2001.
E-mail: `gih@apnic.net`

Packet Classification: A Faster, More General Alternative to Demultiplexing

by Douglas Comer, Purdue University

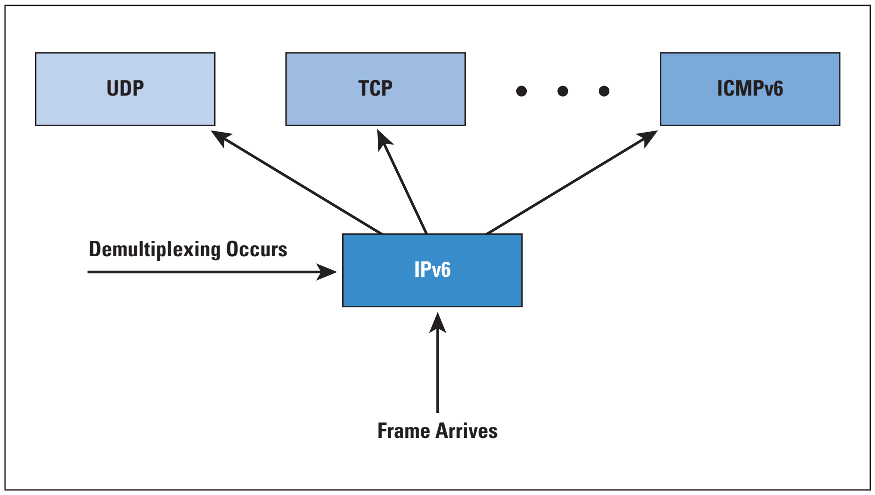
Traditional packet-processing systems use an approach known as *demultiplexing* to handle incoming packets (refer to [1] for details). When a packet arrives, protocol software uses the contents of a *Type Field* in a protocol header to decide how to process the payload in the packet. For example, the Type field in a frame is used to select a Layer 3 module to handle the frame, as Figure 1 illustrates.

Figure 1: Frame Demultiplexing



Demultiplexing is repeated at each level of the protocol stack. For example, IPv6 uses the *Next Header* field to select the correct transport layer protocol module, as Figure 2 illustrates.

Figure 2: Demultiplexing at Layer 3



Modern, high-speed network systems take an entirely different view of packet processing. In place of demultiplexing, they use a technique known as *classification*[2]. Instead of assuming that a packet proceeds through a protocol stack one layer at a time, they allow processing to cross layers. (In addition to being used by companies such as Cisco and Juniper, classification has been used in Linux[3] and with network processors by companies such as Intel and Netronome[4].)

Packet classification is especially pertinent to three key network technologies. First, Ethernet switches use classification instead of demultiplexing when they choose how to forward packets. Second, a router that sends incoming packets over *Multiprotocol Label Switching* (MPLS) tunnels uses classification to choose the appropriate tunnel. Third, classification provides the basis for *Software-Defined Networking* (SDN) and the *OpenFlow* protocol.

Motivation for Classification

To understand the motivation for classification, consider a network system that has protocol software arranged in a traditional layered stack. Packet processing relies on demultiplexing at each layer of the protocol stack. When a frame arrives, protocol software looks at the Type field to learn about the contents of the frame payload. If the frame carries an IP datagram, the payload is sent to the IP protocol module for processing. IP uses the destination address to select a next-hop address. If the datagram is in *transit* (that is, passing through the router on its way to a destination), IP forwards the datagram by sending it back out one of the interfaces. A datagram reaches TCP only if the datagram is destined for the router itself. TCP then uses the protocol port numbers in the TCP segment to further demultiplex the incoming datagram among multiple application programs.

To understand why traditional layering does not solve all problems, consider MPLS processing. In particular, consider a router at the border between a traditional internet and an MPLS core. Such a router must accept packets that arrive from the traditional internet and choose an MPLS path over which to send the packet. Why is layering pertinent to path selection? In many cases, network managers use transport layer protocol port numbers when choosing a path. For example, suppose a manager wants to send all web traffic down a specific MPLS path. All the web traffic will use TCP port 80, meaning that the selection must examine TCP port numbers.

Unfortunately, in a traditional demultiplexing scheme, a datagram does not reach the transport layer unless the datagram is destined for the local network system. Therefore, protocol software must be reorganized to handle MPLS path selection. We can summarize:

> *A traditional protocol stack is insufficient for the task of MPLS path selection because path selection often involves transport layer information and a traditional stack will not send transit datagrams to the transport layer.*

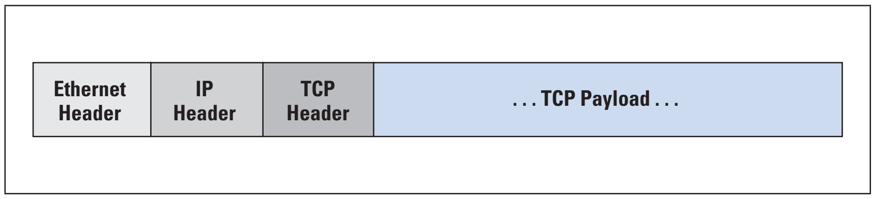
Classification Instead of Demultiplexing

How should protocol software be structured to handle tasks such as MPLS path selection? The answer lies in the use of *classification*. A classification system differs from conventional demultiplexing in two ways:

- Ability to cross multiple layers
- Higher speed than demultiplexing

To understand classification, imagine a packet that has been received at a router and placed in memory. *Encapsulation* means that the packet will have a set of contiguous protocol headers at the beginning. For example, Figure 3 illustrates the headers in a TCP packet (for example, a request sent to a web server) that has arrived over an Ethernet.

Figure 3: Layout of a Packet in Memory



Given a packet in memory, how can we quickly determine whether the packet is destined to the web? A simplistic approach simply looks at one field in the headers: the TCP destination port number. However, it could be that the packet is not a TCP packet at all. Maybe the frame is carrying *Address Resolution Protocol* (ARP) data instead of IP. Or maybe the frame does indeed contain an IP datagram, but instead of TCP the transport layer protocol is the *User Datagram Protocol* (UDP). To make certain that it is destined for the web, software needs to verify each of the headers: the frame contains an IP datagram, the IP datagram contains a TCP segment, and the TCP segment is destined for the web.

Instead of parsing protocol headers, think of the packet as an array of octets in memory. Consider IPv4 as an example. To be an IPv4 datagram, the Ethernet Type field (located in array positions 12 and 13) must contain `0x0800`. The IPv4 Protocol field, located at position 23, must contain `6` (the protocol number for TCP). The Destination Port field in the TCP header must contain `80`. To know the exact position of the TCP header, we must know the size of the IP header. Therefore, we check the header length octet of the IPv4 header. If the octet contains `0x45`, the TCP destination port number will be found in array positions 36 and 37.

As another example, consider classifying *Voice over IP* (VoIP) traffic that uses the *Real-Time Transport Protocol* (RTP). Because RTP is not assigned a specific UDP port, vendors use a heuristic to determine whether a given packet carries RTP traffic: check the Ethernet and IP headers to verify that the packet carries UDP, and then examine the octets at a known offset in the RTP packet to verify that the value matches the value used by a known codec.

Observe that all the checks described in the preceding paragraphs require only array lookup. That is, the lookup mechanism treats the packet as an array of octets and merely checks to verify that location *X* contains value *Y,* location *Z* contains value *W,* and so on—the mechanism does not need to understand any of the protocol headers or the meaning of values. Furthermore, observe that the lookup scheme crosses multiple layers of the protocol stack.

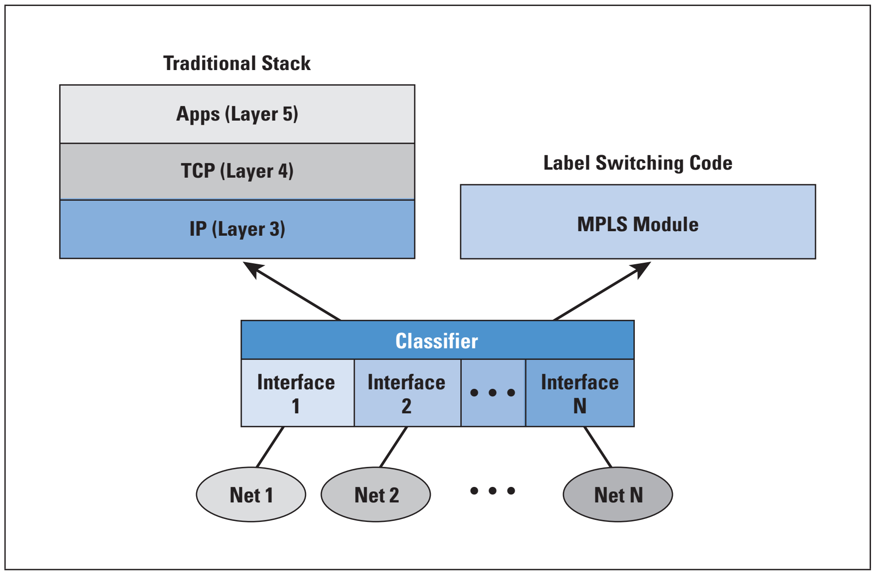
We use the term *classifier* to describe a mechanism that uses the lookup approach described previously, and we say that the result is a packet *classification.* In practice, a classification mechanism usually takes a list of classification *rules* and applies them until a match is found. For example, a manager might specify three rules: send all web traffic to MPLS path *1,* send all FTP traffic to MPLS path *2,* and send all VPN traffic to MPLS path *3.*

Layering When Classification Is Used

If classification crosses protocol layers, how does it relate to traditional layering diagrams? We can think of classification as an extra layer that has been squeezed between Layer 2 and Layer 3. When a packet arrives, the packet passes from a Layer 2 module to the classification module. All packets proceed to the classifier; no demultiplexing occurs before classification. If any of the classification rules matches the packet, the classification layer follows the rule. Otherwise, the packet proceeds up the traditional protocol stack. For example, Figure 4 illustrates layering when classification is used to send some packets across MPLS paths.

Interestingly, a classification layer can subsume all demultiplexing. That is, instead of classifying packets only for MPLS paths, the classifier can be configured with additional rules that check the Type field in a frame for IPv4, IPv6, ARP, *Reverse ARP* (RARP), and so on.

Figure 4: Layering in a Router that Uses Classification to Select MPLS Paths



Classification Hardware and Network Switches

The text in the previous section describes a classification mechanism that is implemented in software—an extra layer is added to a software protocol stack that classifies frames after they arrive at a router. Classification can also be implemented in hardware. In particular, Ethernet switches and other packet-processing hardware devices contain classification hardware that allows packet classification and forwarding to proceed at high speed. The next sections explain hardware classification mechanisms.

We think of network devices, such as switches, as being divided into broad categories by the level of protocol headers they examine and the consequent level of functions they provide:

- Layer 2 Switching
- Layer 2 *Virtual Local-Area Network* (VLAN) Switching
- Layer 3 Switching
- Layer 4 Switching

A *Layer 2 Switch* examines the *Media Access Control* (MAC) source address in each incoming frame to learn the MAC address of the computer that is attached to each port. When a switch learns the MAC addresses of all the attached computers, the switch can use the destination MAC address in each frame to make a forwarding decision. If the frame is unicast, the switch sends only one copy of the frame on the port to which the specified computer is attached. For a frame destined to the broadcast or a multicast address, the switch delivers a copy of the frame to all ports.

A *VLAN Switch* adds one level of virtualization by permitting a manager to assign each port to a specific VLAN. Internally, VLAN switches extend forwarding in a minor way: instead of sending broadcasts and multicasts to all ports on the switch, a VLAN switch consults the VLAN configuration and sends them only to ports on the same VLAN as the source.

A *Layer 3 Switch* acts like a combination of a VLAN switch and a router. Instead of using only the Ethernet header when forwarding a frame, the switch can look at fields in the IP header. In particular, the switch watches the source IP address in incoming packets to learn the IP address of the computer attached to each switch port. The switch can then use the IP destination address in a packet to forward the packet to its correct destination.

A *Layer 4 Device* extends the examination of a packet to the transport layer. That is, the device can include the TCP or UDP Source and Destination Port fields when making a forwarding decision.

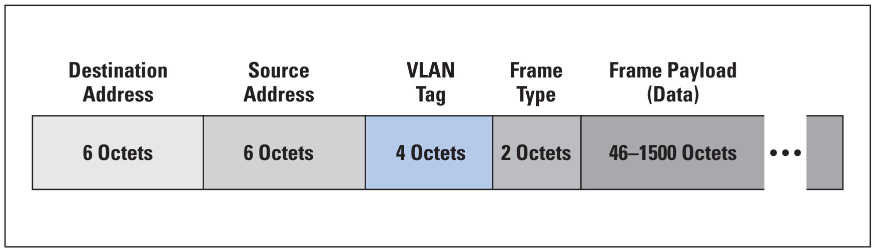
Switching Decisions and VLAN Tags

All types of switching hardware described previously use classification. That is, switches operate on packets as if a packet is merely an array of octets, and individual fields in the packet are specified by giving offsets in the array. Thus, instead of demultiplexing packets, a switch treats a packet syntactically by applying a set of classification rules similar to the rules described previously.

Surprisingly, even VLAN processing is handled in a syntactic manner. Instead of merely keeping VLAN information in a separate data structure that holds meta information, the switch inserts an extra field in an incoming packet and places the VLAN number of the packet in the extra field. Because it is just another field, the classifier can reference the VLAN number just like any other header field.

We use the term *VLAN Tag* to refer to the extra field inserted in a packet. The tag contains the VLAN number that the manager assigned to the port over which the frame arrived. For Ethernet, IEEE standard 802.1Q specifies placing the VLAN Tag field after the MAC Source Address field. Figure 5 illustrates the format.

Figure 5: An Ethernet Frame with a VLAN Tag Inserted



A VLAN tag is used only internally—after the switch has selected an output port and is ready to transmit the frame, the tag is removed. Thus, when computers send and receive frames, the frames do not contain a VLAN tag.

An exception can be made to the rule: a manager can configure one or more ports on a switch to leave VLAN tags in frames when sending the frame. The purpose is to allow two or more switches to be configured to operate as a single, large switch. That is, the switches can share a set of VLANs—a manager can configure each VLAN to include ports on one or both of the switches.

Classification Hardware

We can think of hardware in a switch as being divided into three main components: a classifier, a set of units that perform actions, and a management component that controls the overall operation. Figure 6 illustrates the overall organization and the flow of packets.

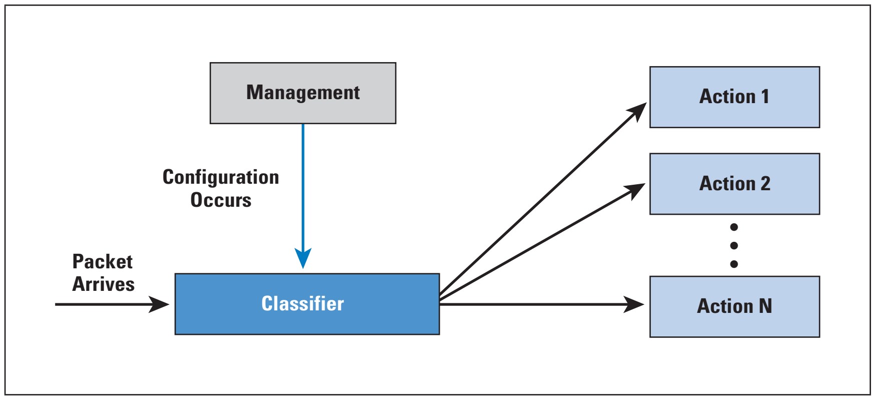


Figure 6: Hardware Components Used for Classification

As black arrows in the figure indicate, the classifier provides the high-speed data path that packets follow. When a packet arrives, the classifier uses the rules that have been configured to choose an action. The management module usually consists of a general-purpose processor that runs management software. A network administrator can interact with the management module to configure the switch, in which case the management module can create or modify the set of rules the classifier follows.

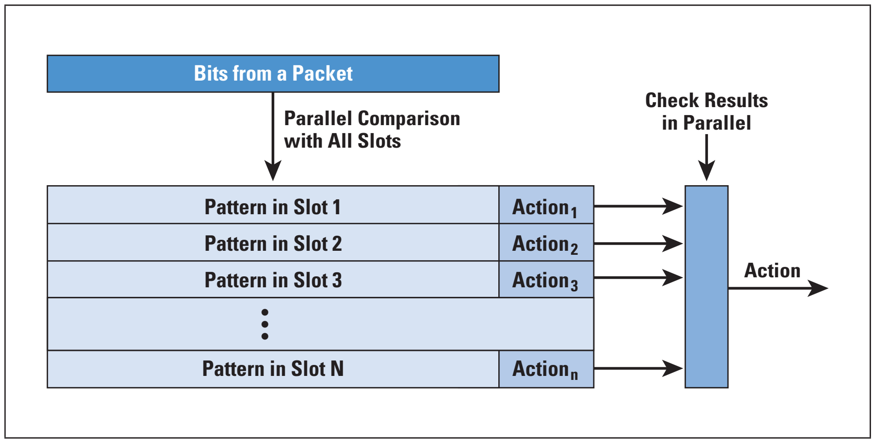
A network system, such as a switch, must be able to handle two types of traffic: transit traffic and traffic destined for the switch itself. For example, to provide management or routing functions, a switch may have a local TCP/IP protocol stack and packets destined for the switch must be passed to the local stack. Therefore, one of the actions a classifier takes may be *"pass packet to the local stack for Demultiplexing"*.

High-Speed Classification and TCAM

Modern switches can allow each interface to operate at 10 Gbps. At 10 Gbps, a frame takes only 1.2 microseconds to arrive, and a switch usually has many interfaces. A conventional processor cannot handle classification at such speeds, so a question arises: how can a hardware classifier achieve high speed? The answer lies in a hardware technology known as *Ternary Content Addressable Memory* (TCAM).

TCAM uses parallelism to achieve high speed—instead of testing one field of a packet at a given time, TCAM checks all fields simultaneously. Furthermore, TCAM performs multiple checks at the same time. To understand how TCAM works, think of a packet as a string of bits. We imagine TCAM hardware as having two parts: one part holds the bits from a packet and the other part is an array of values that will be compared to the packet. Entries in the array are known as *slots*. Figure 7 illustrates the idea.

Figure 7: The Conceptual Organization of TCAM



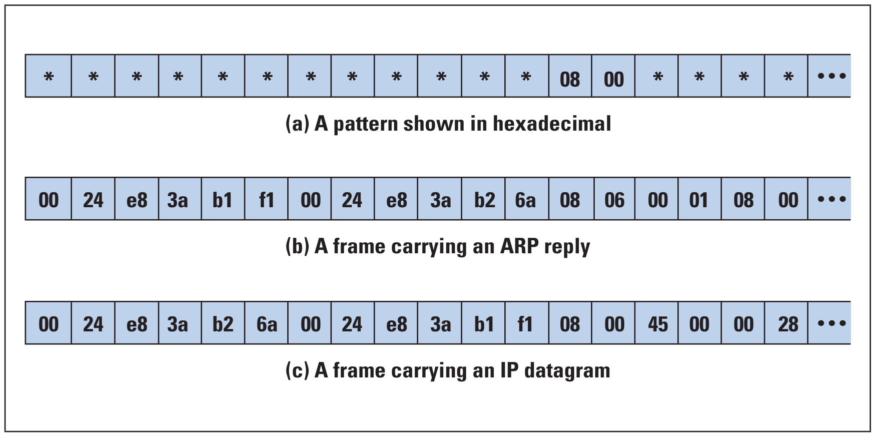
In the figure, each slot contains two parts. The first part consists of hardware that compares the bits from the packet to the pattern stored in the slot. The second part stores a value that specifies an action to be taken if the pattern matches the packet. If a match occurs, the slot hardware passes the action to the component that checks all the results and announces an answer.

One of the most important details concerns the way TCAM handles multiple matches. In essence, the output circuitry selects one match and ignores the others. That is, if multiple slots each pass an action to the output circuit, the circuit accepts only one and passes the action as the output of the classification. For example, the hardware may choose the lowest slot that matches. In any case, the action that the TCAM announces corresponds to the action from one of the matching slots.

The figure indicates that a slot holds a *pattern* rather than an exact value. Instead of merely comparing each bit in the pattern to the corresponding bit in the packet, the hardware performs a pattern match. The adjective *ternary* is used because each bit position in a pattern can have three possible values: a one, a zero, or a "don't care". When a slot compares its pattern to the packet, the hardware checks only the one and zero bits in the pattern—the hardware ignores pattern bits that contain "don't care". Thus, a pattern can specify exact values for some fields in a packet header and omit other fields.

To understand TCAM pattern matching, consider a pattern that identifies IP packets. Identifying such packets is easy because an Ethernet frame that carries an IPv4 datagram will have the value `0x0800` in the Ethernet Type field. Furthermore, the Type field occupies a fixed position in the frame: bits 96 through 111. Thus, we can create a pattern that starts with 96 "don't care" bits (to cover the Ethernet destination and source MAC addresses) followed by 16 bits with the binary value `0000100000000000` (the binary equivalent of `0x0800`) to cover the Type field. All remaining bit positions in the pattern will be "don't care". Figure 8 illustrates the pattern and example packets.

Figure 8: A TCAM Pattern and Example Packets


* * * * * * * * * * * * 08 00 * * * * •••
(a) A pattern shown in hexadecimal
00 24 e8 3a b1 f1 00 24 e8 3a b2 6a 08 06 00 01 08 00 •••
(b) A frame carrying an ARP reply
00 24 e8 3a b2 6a 00 24 e8 3a b1 f1 08 00 45 00 00 28 •••
(c) A frame carrying an IP datagram

Although a TCAM hardware slot has one position for each bit, the figure does not display individual bits. Instead, each box corresponds to one octet, and the value in a box is a hexadecimal value that corresponds to 8 bits. We use hexadecimal simply because binary strings are too long to fit into a figure comfortably.

The Size of a TCAM

A question arises: how large is a TCAM? The question can be divided into two important aspects:

- *The number of bits in a slot:* The number of bits per slot depends on the type of Ethernet switch. A basic switch uses the destination MAC address to classify a packet. Because a MAC address is 48 bits, TCAM in a basic switch needs only 48 bit positions. A VLAN switch needs 128 bit positions to cover the VLAN tag as well as source and destination MAC addresses. A Layer 3 switch must have sufficient bit positions to cover the IP header as well as the Ethernet header. For IPv6, the header size is large and variable—in most cases, a pattern will need to cover extension headers as well as the base header.
- *The total number of slots:* The total number of TCAM slots determines the maximum number of patterns a classifier can hold. When a switch learns the MAC address of a computer that has been plugged into a port, the switch can store a pattern for the address. For example, if a computer with MAC address X is plugged into port 29, the switch can create a pattern in which destination address bits match X and the action is *"send packet to output port 29"*.

A switch can also use patterns to control broadcasting. When a manager configures a VLAN, the switch can add an entry for the VLAN broadcast. For example, if a manager configures VLAN 9, an entry can be added in which the destination address bits are all 1s (that is, the Ethernet broadcast address) and the VLAN tag is 9. The action associated with the entry is *"broadcast on VLAN 9"*.

A Layer 3 switch can learn the IP source address of computers attached to the switch, and can use TCAM to store an entry for each IP address. Similarly, it is possible to create entries that match Layer 4 protocol port numbers (for example, to direct all web traffic to a specific output). SDN technologies allow a manager to place patterns in the classifier to establish paths through a network and direct traffic along the paths. Because such classification rules cross multiple layers of the protocol stack, the potential number of items stored in a TCAM can be large.

TCAM seems like an ideal mechanism because it is both extremely fast and versatile. However, TCAM has two significant drawbacks: cost and heat. The cost is high because TCAM has parallel hardware for each slot and the overall system is designed to operate at high speed. In addition, because it operates in parallel, TCAM consumes much more energy than conventional memory (and generates more heat). Therefore, designers minimize the amount of TCAM to keep costs and power consumption low. A typical switch has 32,000 entries.

Classification-Enabled Generalized Forwarding

Perhaps the most significant advantage of a classification mechanism arises from the generalizations it enables. Because classification examines arbitrary fields in a packet before any demultiplexing occurs, cross-layer combinations are possible. For example, classification can specify that all packets from a given MAC address should be forwarded to a specific output port regardless of the packet contents. In addition, classification can make forwarding decisions depend on combinations of source and destination. An *Internet Service Provider* (ISP) can choose to forward all packets with IP source address X that are destined for web server W along one path while forwarding packets with IP source address Y that are destined to the same web server along another path.

ISPs need the generality that classification offers to handle traffic engineering that is not usually available in a conventional protocol stack. In particular, classification allows an ISP to offer tiered services in which the path a packet follows depends on a combination of the type of traffic and how much the customer pays.

Summary

Classification is a fundamental performance optimization that allows a packet-processing system to cross layers of the protocol stack without demultiplexing. A classifier treats each packet as an array of bits and checks the contents of fields at specific locations in the array.

Classification offers high-speed forwarding for network systems such as Ethernet switches and routers that send packets across MPLS tunnels. To achieve the highest speed, classification can be implemented in hardware; a hardware technology known as TCAM is especially useful because it employs parallelism to perform classification at extremely high speed.

The generalized forwarding capabilities that classification provides allow ISPs to perform traffic engineering. When making a forwarding decision, a classification mechanism can use the source of a packet as well as the destination (for example, to choose a path based on the tier of service to which a customer subscribes).

Acknowledgment
Material in this article has been taken with permission from Douglas E. Comer, *Internetworking With TCP/IP Volume 1: Principles, Protocols, and Architecture,* Sixth edition, 2013.

References

[1] Douglas E. Comer and David L. Stevens, *Internetworking With TCP/IP Volume 2: Design, Implementation, and Internals,* Prentice-Hall, Upper Saddle River, NJ, Third edition, 1999.

[2] **`yuba.stanford.edu/~nickm/papers/classification_tutorial_01.pdf`**

[3] Patrick McHardy, "nfttables: A Successor to iptables, ip6tables, ebtables and arptables," *Netfilter Workshop 2008,* Paris, 2008.

[4] Douglas E. Comer, *Network Systems Design Using Network Processors,* Intel IXP 2xxx version, Prentice-Hall, Upper Saddle River, NJ, 2006.

DOUGLAS E. COMER is a Distinguished Professor of Computer Science at Purdue University. Formerly, he served as VP of Research and Research Collaboration at Cisco Systems. As a member of the original IAB, he participated in early work on the Internet, and is internationally recognized as an authority on TCP/IP protocols and Internet technologies. He has written a series of best-selling technical books, and his three-volume *Internetworking* series is cited as an authoritative work on Internet technologies. His books, which have been translated into 16 languages, are used in industry and academia in many countries. Comer consults for industry, and has lectured to thousands of professional engineers and students around the world. For 20 years he was editor-in-chief of the journal *Software—Practice and Experience.* He is a Fellow of the ACM and the recipient of numerous teaching awards. E-mail: **`comer@cs.purdue.edu`**

Fragments

Internet Society Disappointed over Fundamental Divides at WCIT-12

On December 14, 2012, The Internet Society released the following statement from President and CEO Lynn St. Amour:

"The Internet Society, like other participants at the *World Conference on International Telecommunications* (WCIT), came to this conference looking for a successful outcome. We were hopeful that it would result in a treaty that would enable growth, further innovation, and advance interoperability in international telecommunications. It was extremely important that this treaty not extend to content, or implicitly or explicitly undermine the principles that have made the Internet so beneficial.

While progress was made in some areas such as transparency in international roaming fees, fundamental divides were exposed leaving a significant number of countries unable to sign the *International Telecommunication Regulations* (ITRs). Statements made by a host of delegations today made it very clear that Internet issues did not belong in the ITRs and that they would not support a treaty that is inconsistent with the multi-stakeholder model of Internet Governance.

We are disappointed that the conference has not been successful in reaching consensus. The Internet Society is dedicated to working with all stakeholders around the world to create the environment that will allow the Internet to grow for the betterment of all people."

For more information, see:
`http://www.internetsociety.org/wcit`

See also:

[0] Geoff Huston, "December in Dubai," *The Internet Protocol Journal,* Volume 15, No. 2, June 2012.

[1] World Conference on International Telecommunications (WCIT-12), `http://www.itu.int/en/wcit-12/Pages/default.aspx`

[2] "NRO contribution to the WCIT Public Consultation Process," `http://www.nro.net/wp-content/uploads/2012/joint-submission-WCIT-RIR.pdf`

[3] "Stop the Net Grab": NRO Shares Concerns About the WCIT Process," `http://www.nro.net/news/nro-shares-concerns-aboutwcit-process`

[4] WCITLeaks.org "Bringing transparency to the ITU," `http://wcitleaks.org`

NRO Observations on WCIT-12 Process

The *Number Resource Organization* (NRO), representing the world's five *Regional Internet address Registries* (RIRs), issued the following statement from Dubai, the site of the recent *World Conference on International Telecommunications* (WCIT):

The conference has clearly not met expectations of many *International Telecommunications Union* (ITU) Member States, and with this unfortunate outcome now clear, we feel compelled to put the following observations on record.

The NRO is concerned about aspects of the WCIT-12 meetings, which have just ended in Dubai, particularly with events in the last days of the conference. Neither the content of this conference, nor its conduct during this critical final period, have met community expectations or satisfied public assurances given prior to the event.

Internet stakeholders around the world watched the WCIT preparations closely, and were hopeful, throughout those processes, of two things: that WCIT would have no bearing on the Internet, its governance or its content; and that the event would allow all voices to be heard. The ITU Secretary General himself made these assurances on multiple occasions, and reiterated them in his opening remarks to the conference.

Regrettably, expected WCIT discussions on traditional telecommunication issues were eclipsed by debates about Internet-related issues. The intensity and length of these debates revealed clearly the depth of genuine concern about the proposals, and also the determination of those who brought them to the meeting.

Perhaps more importantly, an open multi-stakeholder conduct of the WCIT conference did not eventuate. Plenary sessions of the conference were webcast, but contributions were allowed only from official Government delegates and ITU officials, relegating all other stakeholders to an observer role.

Furthermore, an important number of critical negotiations occurred in small groups accessible only to Member States; and key experts and other stakeholders were unable even to observe them.

The NRO strongly supports the principles established in 2005 by the *World Summit on the Information Society,* which call for Internet Governance to be carried out in a multi-stakeholder manner, and we note that these represent the view of the global community as expressed through the United Nations system itself.

The NRO has also participated in many ITU conferences and study groups over the years, at very substantial cost, in genuine efforts to build relationships between our communities and to demonstrate the value of multi-stakeholder cooperation and collaboration. The NRO will continue to participate in the ITU, itself a member of the UN system, in expectation that its processes can evolve visibly, and much more rapidly, towards these accepted principles.

John Jason Brzozowski, Donn Lee, and Paul Saab win 2012 Itojun Awards

The fourth annual *Itojun Service Awards* were recently presented to John Jason Brzozowski for his tireless efforts in providing IPv6 connectivity to cable broadband users across North America and evangelizing the importance of IPv6 deployment globally, and to Donn Lee and Paul Saab for their efforts in making high-profile online content available over IPv6 and for their key contributions to *World IPv6 Day* and *World IPv6 Launch*. The awardees were recognized at the *Internet Engineering Task Force* (IETF) 85 meeting in November 2012 in Atlanta, Georgia.

First awarded in 2009, the award honors the memory of Dr. Jun-ichiro "Itojun" Hagino, who passed away in 2007 at the age of 37. The award, established by the friends of Itojun and administered by the Internet Society, recognizes and commemorates the extraordinary dedication exercised by Itojun over the course of IPv6 development. IPv6, the next-generation Internet protocol developed within the IETF, provides more than 340 trillion, trillion, trillion addresses, enabling billions of people and a huge range of devices to connect with one another, and helping ensure the Internet continues its current growth rate indefinitely.

"The combined work of John, Donn, and Paul has made IPv6 a technology used every day by people around the world as they access some of the most popular websites from their homes and offices," said Jun Murai of the Itojun Service Award committee and founder of the WIDE Project.

"On behalf of the Itojun Service Award committee, I am extremely pleased to present this award to them for their ongoing efforts that have made IPv6 a mainstream technology for global web companies looking to ensure their continued growth."

The Itojun Service Award is focused on pragmatic contributions to developing and deploying IPv6 in the spirit of serving the Internet. With respect to the spirit, the selection committee seeks contributors to the Internet as a whole; open source developers are a common example of such contributors, although this is not a requirement for expected nominees.

While the committee primarily considers practical contributions such as software development or network operation, higher level efforts that help those direct contributions will also be appreciated in this regard. The contribution should be substantial, but could be at an immature stage or be ongoing; this award aims to encourage the contributor to continue their efforts, rather than just recognizing well established work. Finally, contributions of a group of individuals will be accepted, as deployment work is often done by a large project, not just a single outstanding individual.

The award includes a presentation crystal, a US$3,000 honorarium, and a travel grant.

John Jason Brzozowski said, "It is truly humbling to be a recipient of the Itojun Service Award, being recognized with others that have worked tirelessly to make IPv6 a reality is rewarding personally and professionally. I would like to thank the award committee and the Internet Society as well as my family and co-workers for their support. As many are aware, the IPv6 journey at Comcast has been unfolding since 2005. It is an honor and pleasure to provide the technical and strategic leadership for IPv6 that has led to the success of our program and the widespread adoption of IPv6."

Donn Lee said, "Deploying IPv6 continues to be an amazing experience. I'm thankful to be sharing this award with my colleagues Paul and John, whom I have worked alongside through the challenging and exciting milestones of World IPv6 Day 2011 and World IPv6 Launch 2012. I especially want to thank the award committee for this honor that remembers Itojun, a truly inspirational IPv6 scientist, leader, and visionary."

Paul Saab said, "I'm honored to be sharing the Itojun Service Award with Donn and John. We should never forget that we would not be here today if it were not for Itojun's trailblazing work and passion for IPv6. To be recognized is extremely humbling, as Facebook's participation could not have been done without our amazing co-workers and their own hard work to bring IPv6 to our users. Thank you for recognizing us and remember that this journey is only 2% complete."

For more information about the Itojun Service Award see:
`http://www.internetsociety.org/what-we-do/grants-and-awards/awards/itojun-service-award`

Left to right: Jun Murai, John Jason Brzozowski, Paul Saab and Don Lee

Leading Global Standards Organizations Endorse "OpenStand" Principles

Five leading global organizations—the *Institute for Electrical and Electronics Engineers* (IEEE), the *Internet Architecture Board* (IAB), *the Internet Engineering Task Force* (IETF), the *Internet Society* and the *World Wide Web Consortium* (W3C)—recently announced that they have signed a statement affirming the importance of a jointly developed set of principles establishing a modern paradigm for global, open standards. The shared "OpenStand" principles—based on the effective and efficient standardization processes that have made the Internet and Web the premiere platforms for innovation and borderless commerce—are proven in their ability to foster competition and cooperation, support innovation and interoperability and drive market success.

The IEEE, IAB, IETF, Internet Society and W3C invite other standards organizations, governments, corporations and technology innovators globally to endorse the principles, available at **`open-stand.org`**

The OpenStand principles strive to encapsulate that successful standardization model and make it extendable across the contemporary, global economy's gamut of technology spaces and markets. The principles comprise a modern paradigm in which the economics of global markets—fueled by technological innovation—drive global deployment of standards, regardless of their formal status within traditional bodies of national representation. The OpenStand principles demand:

- Cooperation among standards organizations;
- Adherence to due process, broad consensus, transparency, balance and openness in standards development;
- Commitment to technical merit, interoperability, competition, innovation and benefit to humanity;
- Availability of standards to all; and
- Voluntary adoption.

"New dynamics and pressures on global industry have driven changes in the ways that standards are developed and adopted around the world," said Steve Mills, president of the IEEE Standards Association.

"Increasing globalization of markets, the rapid advancement of technology and intensifying time-to-market demands have forced industry to seek more efficient ways to define the global standards that help expand global markets. The OpenStand principles foster the more efficient international standardization paradigm that the world needs."

Added Leslie Daigle, chief Internet technology officer with the Internet Society: "International standards development for borderless economics is not ad hoc; rather, it has a paradigm—one that has demonstrated agility and is driven by technical merit.

The OpenStand principles convey the power of bottom-up collaboration in harnessing global creativity and expertise to the standards of any technology space that will underpin the modern economy moving forward."

Standards developed and adopted via the OpenStand principles include IEEE standards for the Internet's physical connectivity, IETF standards for end-to-end global Internet interoperability and the W3C standards for the World Wide Web.

"The Internet and World Wide Web have fueled an economic and social transformation, touching billions of lives. Efficient standardization of so many technologies has been key to the success of the global Internet," said Russ Housley, IETF chair. "These global standards were developed with a focus toward technical excellence and deployed through collaboration of many participants from all around the world. The results have literally changed the world, surpassing anything that has ever been achieved through any other standards-development model."

Globally adopted design-automation standards, which have paved the way for a giant leap forward in industry's ability to define complex electronic solutions, provide another example of standards developed in the spirit of the OpenStand principles. Another technology space that figures to demand such standards over the next decades is the global smart-grid effort, which seeks to augment regional facilities for electricity generation, distribution, delivery and consumption with a two-way, end-to-end network for communications and control.

"Think about all that the Internet and Web have enabled over the past 30 years, completely transforming society, government and commerce," said W3C chief executive officer Jeff Jaffe. "It is remarkable that a small number of organizations following a small number of principles have had such a huge impact on humanity, innovation and competition in global markets."

Bernard Aboba, chair of the IAB said: "The Internet has been built on specifications adopted voluntarily across the globe. By valuing running code, interoperability and deployment above formal status, the Internet has democratized the development of standards, enabling specifications originally developed outside of standards organizations to gain recognition based on their technical merit and adoption, contributing to the creation of global communities benefiting humanity. We now invite standards organizations, as well as governments, companies and individuals to join us at **`open-stand.org`** in order to affirm the principles that have nurtured the Internet and underpin many other important standards—and will continue to do so."

New Year's Day 2013 Marks 30th Anniversary of Major Milestone for the Internet

On January 1, 1983, the ARPANET, a direct predecessor of today's Internet, implemented the *Transmission Control Protocol/Internet Protocol* (TCP/IP) in a transition that required all connected computers to convert to the protocol simultaneously. The open TCP/IP protocol is now a foundational technology for the networks around the world that make up the global Internet and interconnect billions of devices. The transition, which was carefully planned over several years before it actually took place, is documented in RFC 801[1] authored by Jon Postel[2].

Throughout its history, the Internet has continued to evolve. Today, deploying IPv6, the latest generation of the IP protocol, is critical to ensuring the Internet's continued growth and to connect the billions of people not yet online. Thousands of major *Internet Service Providers* (ISPs), home networking equipment manufacturers, and web companies around the world are coming together to permanently enable IPv6 for their products and services through efforts such as *World IPv6 Launch*[3] organized by the Internet Society.

For more information about the Internet Society's work to facilitate the open development of standards, protocols, and administration, and to ensure a robust, secure technical infrastructure, see the *Internet Technology Matters* blog[4] and the *Deploy360 Programme*[5]. For further details about the Internet's history and development, see [6].

[1] Jon Postel, "NCP/TCP transition plan," RFC 801, November 1981.

[2] **http://www.internethalloffame.org/inductees/jon-postel**

[3] **http://www.worldipv6launch.org/**

[4] **http://www.internetsociety.org/what-we-do/internet-technology-matters**

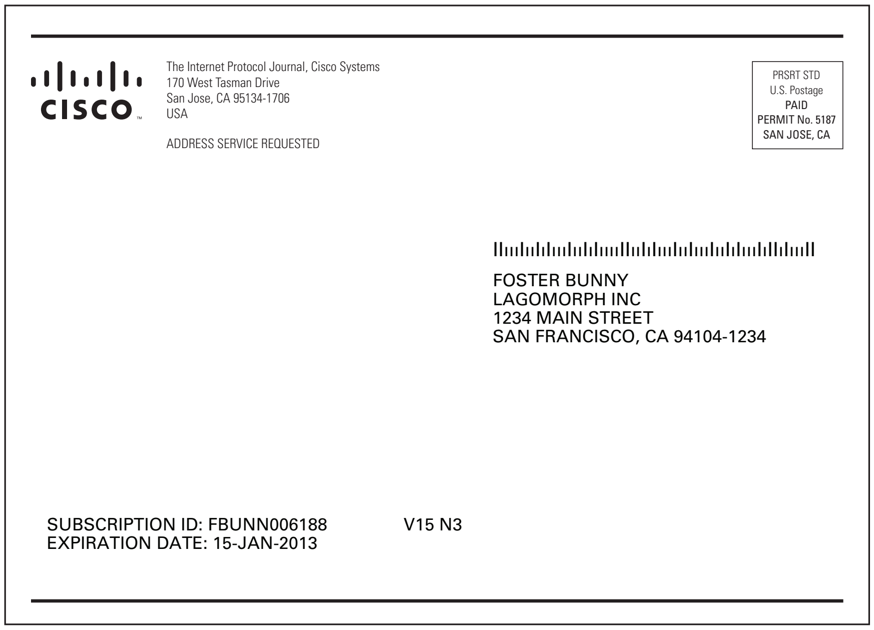
[5] **http://www.internetsociety.org/deploy360/**

[6] Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, and Stephen Wolff, "Brief History of the Internet," **http://www.internetsociety.org/internet/what-internet/history-internet/brief-history-internet**

IPJ Subscription FAQ

What is my "Subscription ID" for The Internet Protocol Journal (IPJ) and where do I find it?

Your Subscription ID is a unique combination of letters and numbers used to locate your subscription in our database. It is printed on the back of your IPJ issue or on the envelope. You will also find information about your subscription expiration date near your Subscription ID. Here is an example:



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

ADDRESS SERVICE REQUESTED

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

FOSTER BUNNY
LAGOMORPH INC
1234 MAIN STREET
SAN FRANCISCO, CA 94104-1234

SUBSCRIPTION ID: FBUNN006188 V15 N3
EXPIRATION DATE: 15-JAN-2013

How do I renew or update my subscription?

From the IPJ homepage (**`www.cisco.com/ipj`**) click "Subscriber Service" and then enter your Subscription ID and your e-mail address in the boxes. After you click "Login" the system will send you an e-mail message with a unique URL that allows access to your subscription record. You can then update your postal and e-mail details, change delivery options, and of course *renew* your subscription.

What will you use my e-mail address and postal address for?

This information is used *only* to communicate with you regarding your subscription. You will receive renewal reminders as well as other information about your subscription. We will never use your address for any form of marketing or unsolicited e-mail.

I didn't receive the special URL that allows me to renew or update my Subscription. Why?

This is likely due to some form of spam filtering. Just send an e-mail message to **`ipj@cisco.com`** with your Subscription ID and any necessary changes and we will make the changes for you.

Do I need my Subscription ID to read IPJ online? What is my username and password?

Your Subscription ID is used *only* for access to your subscription record. No username or password is required to read IPJ. All back issues are available for online browsing or for download at **`www.cisco.com/ipj`**

I can't find my Subscription ID and I have since changed e-mail address anyway; what do I do now?

Just send a message to **`ipj@cisco.com`** and we will take care of it for you.

Call for Papers

The Internet Protocol Journal (IPJ) is published quarterly by Cisco Systems. The journal is not intended to promote any specific products or services, but rather is intended to serve as an informational and educational resource for engineering professionals involved in the design, development, and operation of public and private internets and intranets. The journal carries tutorial articles ("What is...?"), as well as implementation/operation articles ("How to..."). It provides readers with technology and standardization updates for all levels of the protocol stack and serves as a forum for discussion of all aspects of internetworking.

Topics include, but are not limited to:

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

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

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

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

ADDRESS SERVICE REQUESTED

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

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