

## Basic Journey of a Packet

Don Parker 2006-07-06

The purpose of this introductory article is to take a basic look at the journey of a packet across the Internet, from packet creation to switches, routers, NAT, and the packet's traverse across the Internet. This topic is recommended for those who are new to the networking and security field and may not have a basic understanding of the underlying process.

### Introduction

Previous articles by this author have looked at the importance of two key areas of computer security for new users: programming and networking. While they are different disciplines, both networking and programming should largely be viewed as complimentary. If it were it not for the early programming of networking protocols there would be no network. That said, does one have to be a programmer in order to fully grasp networking concepts and theory at a low level? In many cases, you do not. However, a reader's natural curiosity will likely lead him toward programming at some point, in order to further experiment with various protocols and networking theory.

For many people new to the field, the first encounter with a computer is still a rather memorable one. When one discovers the Internet, the sheer magnitude of information provokes a sense of awe, and hopefully creates some curiosity on how the underlying technologies work. Perhaps the following quote from Humpty Dumpty is quite apropos, "We've crossed through the looking glass, Alice." One has seemingly entered a whole new world when using a computer to interact with other systems across the world. It is the curiosity about how this works that drives those new to the field to need to understand how did the computer and network does its thing. How did the information go from one computer to the next, and in turn go through all the various devices in order to reach its destination, sometimes continents away?

### These are the voyages...

The opening lines of Star Trek are really rather appropriate for the journey undertaken by a packet. Once an Internet application is invoked, a whole series of events takes place. This article will be a simple introduction to how a packet is created and the various devices it will travel through on the way to its destination. Having an understanding of just what happens between point A and point Z can be quite helpful in furthering your understanding of networking.

What I shall now describe is what happens from the time an application is invoked to the time that the packets generated by this application reach their target. Suppose as a user you invoke Firefox to check out the news on your favorite web page. In the background, transparent to you the user, a whole series of events has now been set in motion. After the initial TCP/IP three way hand shake has been negotiated, your web browser will issue a request to the web server hosting your homepage asking for its homepage.

This HTTP GET request information now has to be sent to the web server. What happens is that Firefox, the application you invoked, will make a write request to the system. This process will cause the data that Firefox wants to send to be copied from the applications memory space to the socket send buffer within kernel space.

Depending on what transport protocol the application uses, the socket layer will call either UDP or TCP. We need to remember that not every application uses TCP as a transport protocol. DNS uses

both UDP and TCP, while other applications such as TFTP will only use UDP. Once the socket layer calls the proper transport protocol, the data will be copied over into a socket buffer.

## Fragmentation

When copying the data from the GET request issued over to a socket buffer, TCP will fragment this data if required. Although a GET request will fit easily into one packet, and will fall within the MTU for Ethernet without a problem, what happens if the browser's request exceeds the MTU? Should this occur, TCP itself will fragment the data in order to ensure that the size complies with the Ethernet MTU limit of 1500 bytes. A key point to remember here is that fragmentation will occur at the TCP layer if the application invoked uses TCP as its transport protocol. Should the application be using UDP then fragmentation, if required, will happen at the IP layer.

If you have ever done packet analysis, you will likely remember being confused by the two byte IP header field of, "flags and fragment offset". It is quite disconcerting to be confronted with a header value that has 13 bits assigned to it. Many budding analysts are confused, as up till then they had been dealing with bytes, nibbles, and sometimes one bit values. There is method to the madness here, as the people who designed these protocols were clever to say the very least. By looking at the IP header again we see that the "Total length" field is a two-byte one. We then realize that if we do the math correctly, the maximum size IP datagram ever allowable is 65535 bytes in length. The illustration below is an example of this calculation as it impacts the "flags and fragment offset."

X	DF	MF	4096	2048	1024	512	256		128	64	32	16	8	4	2	1
1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1

We can see from the above that even if all thirteen bits are set, you will only get a total value of 8191. That is nowhere near the 65535 maximal IP datagram size. What actually happens is that the fragmentation offset you see in packets is really in multiples of eight bytes. For instance the numerical values that you see in the fragmentation offset should be multiplied by eight so as to ascertain their exact place in the fragmented datagram.

## On down the stack we go

Once the data has made past its respective transport layer, let's go down to the IP layer. At this point the IP header is built and the all important IP addresses are added. After this, the data then drops down to the data link layer where both the logical link control and media access control layers do their parts. Finally the data is now ready to be transmitted by the physical layer as embodied by your NIC card. So in a series of 1's and 0's, off goes the GET request made from your browser along the physical medium until it reaches the Ethernet switch. For most users at home, a SoHo router is actually a combination of both a switch and a simple router. For corporate users, the switch is a separate piece of hardware from the router itself. If in a corporate environment then the computer is likely hooked to the switch via a length of CAT-5 cable or other medium. If the switch does not have a hard-coded CAM table then the switch takes note of the computer's MAC address (unique for each Ethernet card). When the packet comes back from its journey carrying the website's data, as requested in the outgoing GET request, the client computer's upstream switch will know where to send that packet back to.

How does the client computer know what its default gateway is? Whether or not it is in a corporate

setting or is a home computer, the system would have normally been issued a DHCP packet once it booted up to get key information from the DHCP server. While not all systems use DHCP, most do and therefore do not have their IP address or gateway predetermined. Information such as which DNS servers to use, its IP address, and the default gateway IP address. If DHCP is turned off, the system administrator would have to enter all of this information by hand in the operating system. Not very efficient, which is why DHCP is largely turned on in most networks.

With the default gateway at hand, the computer knows where to go to in order to access the Internet and retrieve the web page, as requested by Firefox when it was invoked. After the packet goes through the switch it makes its way to the router, and may very well go through a firewall prior to the router as well. Should the packet go through a firewall first, the firewall will make note of several key features. A stateful firewall, which is what most firewalls are nowadays, will log the source IP address and port, plus the destination IP address and port. The firewall will keep this information in a state table in memory, as this is how it regulates access to the internal network. If a packet has not been logged leaving the network then it simply won't be allowed back into it. In overly simplified terms, that's how a firewall protects your computer.

## Of routers and NAT

Now that the packet has passed through the firewall, if present, it is going to traverse the router. What will now happen is that the typically private IP address that this packet has (suppose it's a 192.168/16 based address) will be converted to a routable public IP address that was given by your ISP, and in turned assigned to your router. Once done, the packet now begins its voyage across the Internet and through a multitude of routers that direct its journey. Each and every time though that the packet is forwarded to a router from another router, something happens to the packet itself.

Let's start by looking at the Internet-facing router. It will route the packet based on its own routing table information. Once the next router receives this same packet it will consult its own routing table, and send it on its way based on what it considers the best path. Before this happens, the router will change several fields in the IP header of the packet. One of the fields that will change is the TTL field or "time to live." Now, as part of the IP header has changed, the router must compute a new checksum value for the packet.

This same situation continues to unfold until the packet reaches its intended target, in our case it's the server hosting the web page. This packet will then be able to get past the router, which would be forwarding connections to the web server's TCP port 80. Once the packet arrives at the web server, the computer which hosts the content will take action on the request.

The physical layer will issue an IRQ to the CPU indicating that there is data to be processed. Once that occurs, the data is passed up the data link layer where the webserver will recognize the MAC as indeed being its own, and will pass it up to the IP layer. The IP address is in turn recognized as belonging to it so that it then passes the data up to the transport layer where it is put into the TCP buffer. At this point the application notices that the data is for it and it processes the information. The end result of this is that the information requested for in the GET request issued by the Firefox client is then sent back. This return process of generating a socket, then a packet that follows is the same path as the one I described above for the Firefox web client when it issued a GET request.

All that now remains is for the new packet to makes its way back to the IP address that originated the request. The same series of events takes place with the packet traversing a series of routers, which may or may not be the same ones its took to get there. With all of this activity taking place in the background, it is often taken for granted but it truly a marvel to understand. Not the least of which is the sheer speed at which this all happens with today's broadband connections. Back in the

early days of the 9600 bps modems, this all occurred at a much slower speed than we now enjoy today.

## Conclusion

The overall theme of this article was to try and impart the wonder that is networking to those who are very new to the field. To be able to understand very basic concepts such as routing, switching, and NAT is no small achievement for someone new to security. Anyone can speak words remembered by rote, but how many can actually explain the theory behind it? It's an excellent exercise to think one's way through a packet being created, all the way till it makes it way back to you. You may be surprised at just how complex a journey that is. The author welcomes your feedback in e-mail.

## Further reading

[TCP/IP Illustrated, Volume 1](#), the original TCP/IP "bible"

[RFC Sourcebook](#) with detailed protocol explanations

[Privacy Statement](#)

Copyright 2006, SecurityFocus