15-853: Algorithms in the Real World
Internet Routing Algorithms

Material from lecture by Bruce Maggs

November 20, 2000

Contents

1 Dirty Tricks

1.1 IP Masquerading (a.k.a. Network Address Translation)

IP Masquerading is a method of translating between the private IP addresses used within a private network and the public IP addresses used on the Internet. IP Masquerading is typically applied to a private network in which all connections to the public Internet pass through a firewall. The firewall translates the IP addresses of packets as they pass through it. For example, the source address of a packet originating on the private network is translated to the firewall's public address. Conversely, as a packet enters the private network through the firewall, its destination address is changed from that of the firewall to that of its destination on the private network. The firewall may also translate the source and destination ports of packets as they pass through, and may use the modified port numbers to help keep track of the open connections between hosts on the private network and hosts on the public Internet. The translation is done transparently, so that no special configuration is needed on either the hosts on the private network, or on the hosts with which they communicate on the public Internet.

The firewall between the private network and the Internet implements IP Masquerading by tracking the source and destination of each connection to the public Internet from the private network, and translating all private IP addresses in the packets sent on those connections to its own Internet address (with an identifying port number for each connection). Figure 1 shows an example of this.

Example: First, host A sends a request for a connection to webserver `www.cs.cmu.edu` using its own (private) IP address and source port as the source. The firewall intercepts the packet and replaces host A's address with its own. It assigns a source port to the connection that distinguishes it from all other connections between this private network and `www.cs.cmu.edu`; in this case, port 1305 is already in use so the firewall assigns port number 1310. The packet is resent from the firewall to `www.cs.cmu.edu` with the new source values, and the firewall records the relevant information about this connection. Further packets from host A are translated in the same manner. Any packets from `www.cs.cmu.edu` must be translated in the reverse manner; the firewall must use the source address and port and the destination port to determine which connection the packet corresponds to. In this case, the firewall finds the connection established by host A and fills in A's address and port values before resending the packet on the private

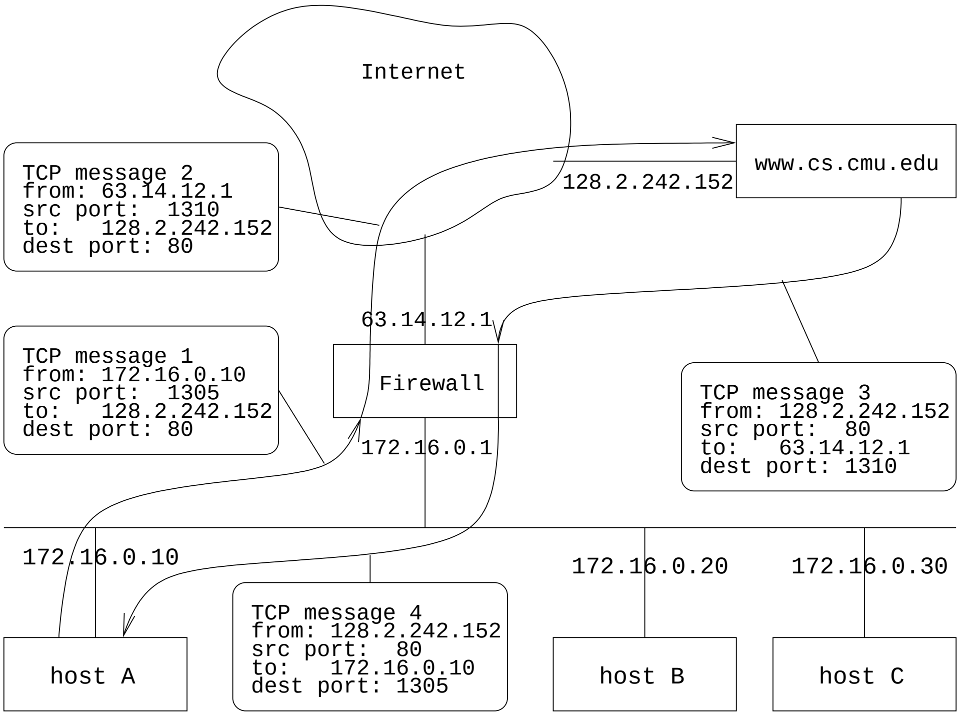

Figure 1: IP Masquerading example

network.

Of course, because any incoming packets have to be rerouted by the firewall based on a port number it has set previously, all connections between public and private addresses must be initiated from within the private network. This makes IP Masquerading a useful security feature for a private network; no external attacker can even start a connection with the internal machines. It does mean the firewall has to closely monitor the traffic for any connections that are used to establish a connectionless protocol - for instance, an RTSP connection that initiates UDP streaming - in order to properly translate addresses for that stream. This requires the firewall to process RTSP; in fact, in order to be able to extract the private addresses from the packets passing through the firewall, it must be able to process all of the protocols being used on top of IP in those packets (e.g., TCP, FTP, RTSP, etc.).

Another motivation for using a firewall, aside for protection from attacks, is the ability to grant Internet access to an entire private network using only

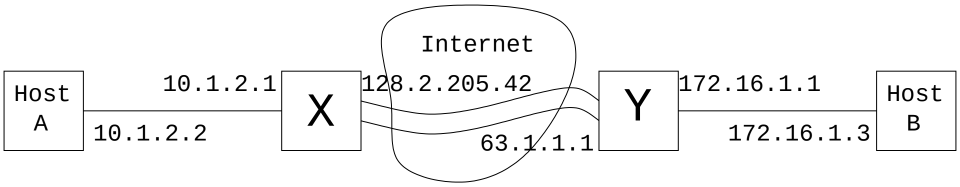

Figure 2: IP Tunneling

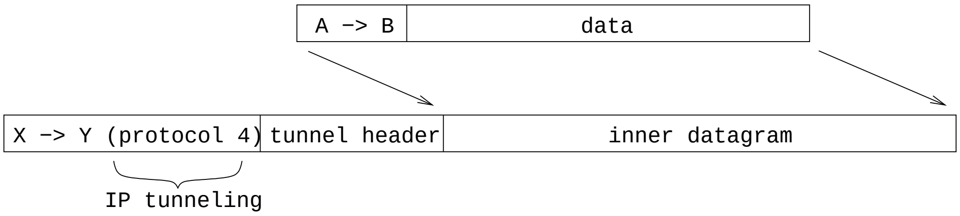

Figure 3: IP Tunneling Datagram

a single public IP address, thus saving IP namespace (and possibly money). The downside to firewalls is that in theory the single-machine firewall could become a bottleneck when processing the traffic of many hosts on the private network. In practice, however, (at least for today) firewalls are nearly always capable of saturating the outgoing connection and so bandwidth becomes a limitation long before the speed of the firewall does.

1.2 Tunneling

Because connections to hosts behind a firewall cannot be established from outside, no connections may be established at all between two hosts on separate private networks using different firewalls. IP Tunneling is a method used to overcome this limitation when the administrators of two private networks wish their internal addresses to be known to each other.

Whenever firewall X (see Figure 2) receives a message from Host A with a destination in Y's private address space (for instance, using address 172.16.1.3 to refer to Host B), it encapsulates that message in an IP tunneling datagram (Figure 3) to send to Y. When received, Y extracts the original message from the IP tunneling datagram and transmits it on the private network where Host B receives it.

This process - the process of translating one protocol into another and back again - is called "tunneling". Other examples of tunneling include the following. Not all Internet routers support the MBONE multicast protocol, so MBONE-enabled routers have to communicate with each other via tunneling. Early networkable computer games such as Doom and Descent were playable over a LAN but not over the Internet. Shareware programs such as Kali and Kahn would capture the ethernet packets these games produced and send them over the Internet to other players.

2 Autonomous Systems

The Internet is made up of Autonomous Systems. An Autonomous system is a group of hosts and routers under a single administration. Every Autonomous system has its own Autonomous System Number (ASN). There are 2^{16} possible ASNs. Carnegie-Mellon has ASN 9.

2.1 Connections between Autonomous Systems

Throughout the world there are hubs called Network Access Points (NAPs) or Metropolitan Area Exchanges (MAEs) where many autonomous systems connect. Autonomous systems can use these hubs to exchange packets from one system to another.

Unfortunately, these hubs tend to be somewhat congested. To avoid this congestion, pairs of autonomous systems will sometimes run connections directly from one system to another. These connections are called Private Peering Links. In general, two systems will make this connection only if they are about equal in size; otherwise the smaller system would end up sending most of its packets through the larger one.

2.2 Communication between Autonomous Systems

The Border Gateway Protocol (BGP) is used to communicate between different autonomous systems. BGP is thus an example of an External Gateway Protocol, or EGP. (In fact the predecessor to BGP was called EGP.) Later we will see that BGP can be used in two modes: by routers in different autonomous systems to exchange routes, and by routers in the same autonomous system to share learned routes. The former mode is called eBGP (external BGP), and the latter iBGP (internal BGP).

2.2.1 Classless Inter-Domain Routing (CIDR)

In BGP, every autonomous system advertises a list of the IP addresses it can reach. Since it would be expensive to advertise each address individually, the routers advertise routing blocks instead. Originally, these routing blocks were divided into classes, each of which divided a 32-bit IP address into block number and host number differently:

1. Class A: First bit is 1, first 8 bits determine block
2. Class B: First bits are 01, first 16 bits determine block
3. Class C: First bits are 001, first 24 bits determine block

Classless Inter-Domain Routing (CIDR) blocks are now supported, however, allowing arbitrary divisions between block and host specification regardless of the bit-prefix. CIDR blocks are written as "w.x.y.z/b", where "w.x.y.z" is a standard IP address, and "b" specifies the number of bits of that IP address that are relevant. For example, the address "5F.A0.00.00/12" describes all IP addresses which begin with "5F.A" (*i.e.*, "5F.Ax.xx.xx").

CIDR blocks can describe classes A, B, and C by using "/8", "/16", and "/24", respectively.

2.2.2 Border Gateway Protocol (BGP)

In eBGP, for every CIDR block an autonomous system can reach, it advertises three things:

1. the block itself,
2. the IP address of the next hop on the path (i.e., the router's own IP address), and
3. the list of the autonomous systems on the path to the destination, beginning with the ASN of router itself.

Advertising the entire path to the destination is important: it prevents routing loops. An autonomous system will ignore any advertisement that contains its own ASN as part of the path.

BGP-enabled routers use TCP to communicate. This allows BGP routers that are not directly connected to communicate through non-BGP routers. This type of configuration is called multihop eBGP, and is typically allowed only for eBGP.

2.3 Communication within Autonomous Systems

When a router within an autonomous system hears of a new route, it uses the iBGP protocol to send that route to all of the other routers within its system. One minor difference between iBGP and eBGP is that when a router sends a route to another router within the same autonomous system, it does not modify the next hop component of the route, but simply retransmits the next hop that it learned from an external source. Similarly, it does not prepend its own ASN to the route.

If an autonomous system is very large, the all-to-all peering in iBGP is inefficient, as the number of iBGP connections required by an AS with n routers would be $\Theta(n^2)$. There are various methods to deal with this.

One method of dealing with this problem is to break a large autonomous system into smaller sub-systems. These sub-systems all advertise the same Autonomous System Number to routers outside the larger system, but they have separate ASNs for internal use. All ASNs above 64000 are reserved for private usage in this manner. These sub-systems are called "Confederations".

Another way to deal with this problem is to have a central "route reflector" router that collects all route information. This router then duplicates and resends the routes to all the other routers.

2.4 The Multi-Exit Discriminator (MED)

Sometimes two autonomous systems are connected at more than one point. When this happens, it's usually the case that one system may prefer to receive packets along a particular connection. This preference is stated using a Multi-Exit Discriminator, or MED; lower values mean a higher preference. For example, if router SF (see Figure 4) is inside autonomous system XNET and needs to send packets to ANET, then ANET would prefer that all or most of those packets come into the AS through router SJ rather than router LA.

Note that the MED numbers do not have any particular scale. In particular, MED numbers from different autonomous systems are not comparable. If router SF is using some protocol that allows it to choose between ANET and YNET for a given route, it is not necessarily true that it should choose router NY over routers SJ and LA based solely on MEDs.

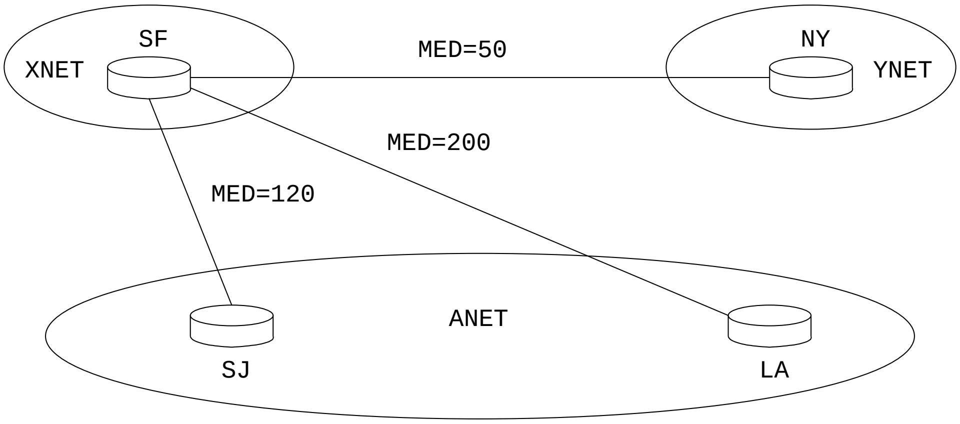

Figure 4: Multi-Exit Discriminators

3 Route Finding

There are two different types of protocols that routers can use to route packets within autonomous systems: Distance Vector protocols and Link State protocols.

3.1 Distance Vector Protocols

In a Distance Vector protocol, every router maintains a table which lists all the IP domains it can reach. For each domain, it lists both the distance (in hops) to that domain and the next router to which it sends packets headed for that domain. Every router advertises its distance vector to every adjacent router.

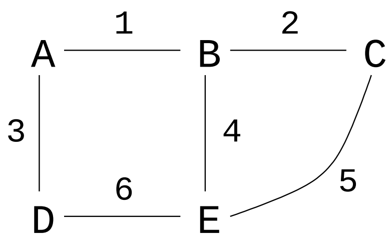

Figure 5: Distance-Vector Network

For example, in Figure 5, router C starts by broadcasting (C,2,1) across link 2 - telling whoever is on the other side (router B) that C is directly on the other side of link 2. Router B makes a note of this in its table and then sends (C,1,2) along link 1, telling router A that C is 2 hops away across link 1. This turns out to be the shortest path for router A to router C, so after all paths are determined router A will have (C,1,2) in its table.

The following is the final distance-vector table for router A:

destination	link	hops
A	local	0
B	1	1
C	1	2
D	3	1
E	1	2

There is some redundancy in this system. Each router rebroadcasts all of its link data across all of its links - so B will even rebroadcast (C,2,2) back to router C, thus determining a path from C to C through B. (Fortunately, router C's local path to itself is listed as shorter so this new path will not be recorded).

3.1.1 The Count-To-Infinity Problem

Suppose that link 2 goes down in a network composed of only routers A, B, and C. (Imagine isolating the top line of Figure 5.) Router B will now realize that it can no longer reach router C in one hop, since it could only do that through link 2. Unfortunately, router A still has the entry (C,1,2) in its table, which it will rebroadcast to router B as (C,1,3). Router B will record this, making the relevant table entries:

Router A			Router B		
destination	link	hops	destination	link	hops
B	1	1	A	1	1
C	1	2	C	1	3

After this, router B will sent the message (C,1,4) to A. Although this is longer than the current path in A's table, it represents the best known path to C through link 1, so A knows that its current path through link 1 is invalid. Router A records this new path and rebroadcasts it to B as (C,1,5) soon after. (Router B records this for the same reason that A recorded (C,1,4)) The tables now show:

Router A			Router B		
destination	link	hops	destination	link	hops
B	1	1	A	1	1
C	1	4	C	1	5

This process can repeat itself indefinitely while link 2 remains severed. The path lengths to router C in the tables of A and B will continue to get larger and larger, hence this is called the Count To Infinity problem. Packets from A or B to router C will be passed eternally between routers A and B, each believing the other to be the next hop on a valid path, until finally the distances are large enough that both routers realize the route does not exist.

3.1.2 Solution 1: Split Horizon

There's a simple fix for this problem: If A learned of its route to C through B, then A shouldn't send any messages about C to B. This prevents a lot of redundant data from passing across the network and solves our simple Count-To-Infinity problem. This technique is called "Split Horizon".

3.1.3 Solution 2: Poison Reverse

Another technique that is used to prevent routing loops and to speed up route convergence is called "Poison Reverse". When router B first sends its routing table to A, for each route that A receives that becomes a best-route entry in its routing table, e.g., the route to C, A advertises the entry back to B with a hop count of infinity. This tells B that the best route from A to C goes through B, and that therefore B should not attempt to route to C through A. Furthermore, if B has advertised to router A a route to C and then the link from B to C goes down, B should advertise to A a route from B to C with a hop count of infinity. Since A learned of its route to C from B, it should then remove the route to C from its table.

Poison Reverse and Split Horizon can be used together. Suppose that routers A, B, and C are connected as before, and that A is also connected to C through a fourth router D. If the link from B to C goes down, the entry for C in the routing table of B will change so that the best path now passes through A and then D. In this case, since B has learned of its current route to C from A, split horizon would prevent B from advertising to router A any route to C. When split horizon and poison reverse are used together, however, B temporarily turns off split horizon and then sends a poison reverse update to all of its neighbors, including A. In this way A learns that B is no longer connected to C.

3.1.4 Count-To-Infinity: Prognosis

Unfortunately, neither Poison Reverse nor Split Horizon solve the problem completely. It only takes a slightly more complex system than the one we describe to defeat these tricks, and then the Count-To-Infinity still happens.

3.2 Link State Protocols

A much more robust class of protocols is the Link State Protocols. In this class, every router tells every other router (not just its neighbors) about every link it's next to. When this is done, every router has a picture of the entire network sitting in memory, and it can apply Dijkstra's Shortest-Path Algorithm to decide where to send packets. This is somewhat slower than the Distance Vector Protocols, but it does not succumb to problems such as Counting To Infinity.

Link State Protocols such as OSPF in large autonomous systems can run into the problem of representing and traversing proportionately large graphs within each router. This problem is relieved by breaking down the autonomous system much like Confederations for iBGP. Within each subset of the system, an entire graph is known; each router then computes only the shortest path to a link out of its subsystem. This greatly reduces the storage and computation requirements of the algorithm.

Examples of Distance-Vector Protocols include RIP, IGRP, and EIGRP. Examples of Link-State Protocols include OSPF and IS-IS. Note that in BGP, routers don't explicitly advertise every link in the network, but they do advertise the full path to any IP domain they claim to reach. This is a nice compromise between avoiding the Count-To-Infinity problem and not having to implement graph analysis on a router.

3.3 Link State Refinements

When sending out messages such as Link-State Advertisements, it's important to make sure that nothing bad happens if the messages arrive out-of-order. A packet containing obsolete distance-vector paths, such as a claim that a dead link is alive, can propagate and cause major routing problems before it is fixed. Such messages are tagged with 32-bit sequence numbers, with higher sequence numbers being more recent than lower sequence numbers. If a router receives a message with a lower sequence number than what it already has, it discards the lower-numbered message.

When the maximum value for a sequence number has been reached (as can easily happen in a long-running router), these sequence numbers wrap around from $(2^{32} - 1)$ to 0. When a router compares sequence numbers x and y, it says that $x > y$ if x is one of the 2^{31} numbers which comes shortly after y in the sequence space. This is called a "Cyclic Sequence Space".

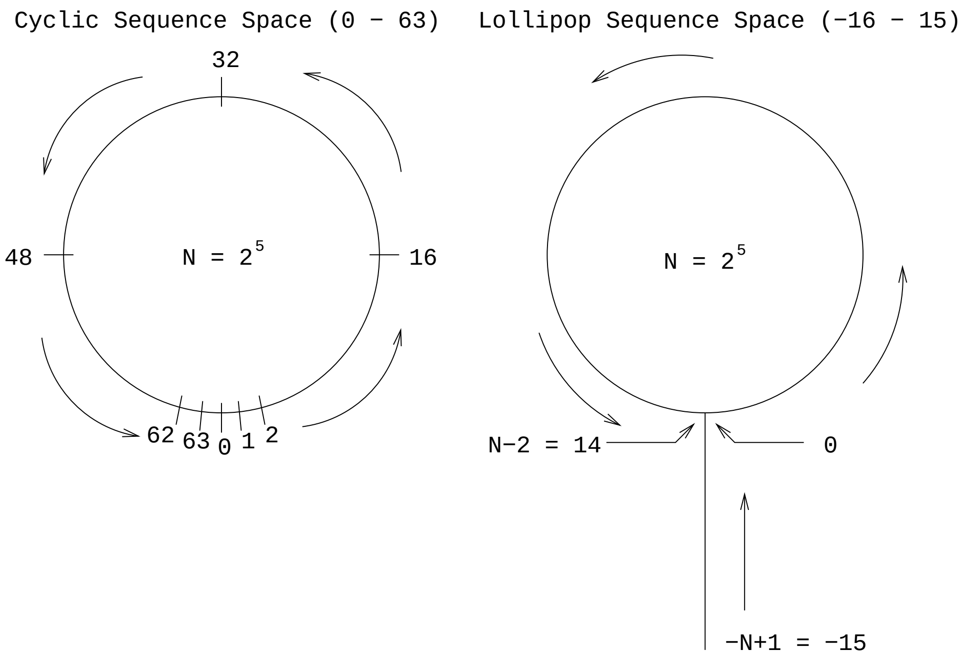

Figure 6: Link State Sequence Numbers

Even with this enhancement, we have to worry about what happens if a router crashes. How does it know what sequence number to start sending at? The solution is to reserve some of the numbers (usually the first half of the valid numbers, or the negative values of a signed number system) for the routers to use immediately after restarting. The rest of the numbers make up a Cyclic Sequence Space as normal. Once the router reaches the end of the reserved numbers, it starts cycling. This is called a "Lollipop Sequence Space" for reasons which are apparent upon viewing Figure 6. Now when a router receives a message with a sequence number drawn from the range reserved for start-up messages, the receiver knows that the sender has just started.