Lecture 16: IPv6, NATs, VPNs, VC Routing in Backbone, and ICMP

1 IPv6

In the early 90's the IETF started working on a new version of IP, **IPv6**. The motivation was the fear that all 32-bit addresses would soon get used up, despite the subnetting and CIDR techniques that we discussed last lecture.

We do not have time to go into all the details of the IPv6, but you should read Section 4.7 of Kurose Ross.

I would like to only mention a few key **differences** between IPv4 and IPv6.

1. IPv6 has a **4 word address size** (**128 bits**, or 16 bytes) rather than the 1 word address size in IPv4. An IPv6 address looks like

$$F74C : 327A : B321 : 179A : BCDA : 1234 : 37DC : AB3C$$

   where each character above occupies 4 bits and there are 16 bits between each colon.

2. IPv6 uses **classless addressing**. The addressing is modeled after **IPv4 with CIDR**. The overall goal in picking addresses is to keep the forwarding tables of the gateway routers short. Towards this end, the idea is to assign a single prefix (network number) to each AS based on the size of the AS, and let the AS assign addresses within. **Geographic-based addressing** has also been considered for IPv6. The idea is to assign one prefix per continent. Thus all addresses in Europe will start with one prefix, all addresses in North America with another, etc.. This again should limit the size of the forwarding tables by adding more hierarchy.

3. IPv6 will **streamline IP operation** in several ways:
   (a) **No more variable-length header**.
   (b) **No more checksum**.
   (c) **No more Fragmentation and Reassembly**. If packet is too big, router drops the packet and sends a message back to the sender saying "Packet too big." It's now up to the sender to make packets the right size. (Note the sender will

get ICMP messages telling it what this right size is). The point is that doing the fragmentation at the routers was too costly. The theory in systems design is that you want to optimize for the **common case** to speed up mean performance. The common case should be very fast. In the common case, fragmentation is not necessary, therefore we don't want to bother including this capability in routers because it slows them down to have to check these extra headers.

4. There is a new header field called **flow id**. The flow id is used to specify a flow (same idea as in the case of virtual connections). The flow field could be used to specify that all packets belonging to this flow should be routed a particular way. More likely, the flow field will be used to specify that all packets belonging to this flow should get a certain QOS or real-time service. We'll discuss this later in the course.

5. There are other features proposed for IPv6 dealing with **new security features** and **multicast**. We will touch on these when we cover these topics in later lectures.

It is interesting to note that while we've been slow to adopt IPv6 because of practical difficulties, many of the features proposed for IPv6 have already been added to todays IPv4 routers, including Path MTU discovery, some support for real-time services, some support for mobile hosts, etc..

The most interesting question regarding IPv6, however, is how we can **transition** from IPv4 to IPv6.

Question: Any ideas?

The obvious, but wrong solution is to have a world-wide **"flag day"**. On this day, we'll assign every machine a new 4-word address and recompute all routing tables.

Question: What are some problems with having a flag day?

Answer: A flag day is both not administratively feasible. First of all, everyone has to agree to change all their machines. Then of course there's the cost of buying IPv6 software/hardware for all routers. Lastly, there's the time that it would take to make these changes. A flag day is also obviously a big security hole.

Another solution is to more slowly add IPv6 capability and add IPv6 addresses to nodes, however all nodes which get the new capabilities should also maintain their original IPv4 capabilities. This is called the **dual stack** approach and these new dually capable nodes are often referred to as IPv6/IPv4 nodes. If an IPv6/IPv4 node is talking to an IPv4 node, it sends IPv4-type datagrams. If on the other hand an IPv6/IPv4 node is talking to another IPv6/IPv4 node, it can send IPv6-type datagrams.

Question: This solution "works" in that it allows IPv6 technology to slowly be integrated into the Internet, however, until all machines are IPv6 machines, a lot of the IPv6 capability will be lost. Why is that?

Answer: Suppose that you are sending a message from IPv6/IPv4 router A to another IPv6/IPv4 router B. You send IPv6 datagrams that include all sorts of other info like the **flow id** for example. Now, suppose that there is an IPv4 router C between A and B, so that the path looks like

$$A \longrightarrow C \longrightarrow B.$$

When A speaks to C, it will have to send an IPv4 datagram. Thus A will never be able to send the flow id information to B.

Question: Any other idea for how A can send B IPv6 datagrams given that there are several IPv4 routers between A and B?

Answer: **Tunneling**, as shown in Figure 1. Imagine that node A would like to send a packet to node F. The problem is that nodes B and E are separated by IPv4 routers. Router B and E agree to form a logical tunnel. Router B takes the entire IPv6 datagram and **encapsulates** it in an IPv4 datagram which is addressed to the IPv4 address of Router E. All the IPv4 routers in the tunnel route this IPv4 datagram amongst themselves, just as they would any other datagram, totally unaware of the contents of the IPv4 datagram. When router E gets the packet, it strips the IPv4 header and determines that the content of the datagram is an IPv6 datagram.

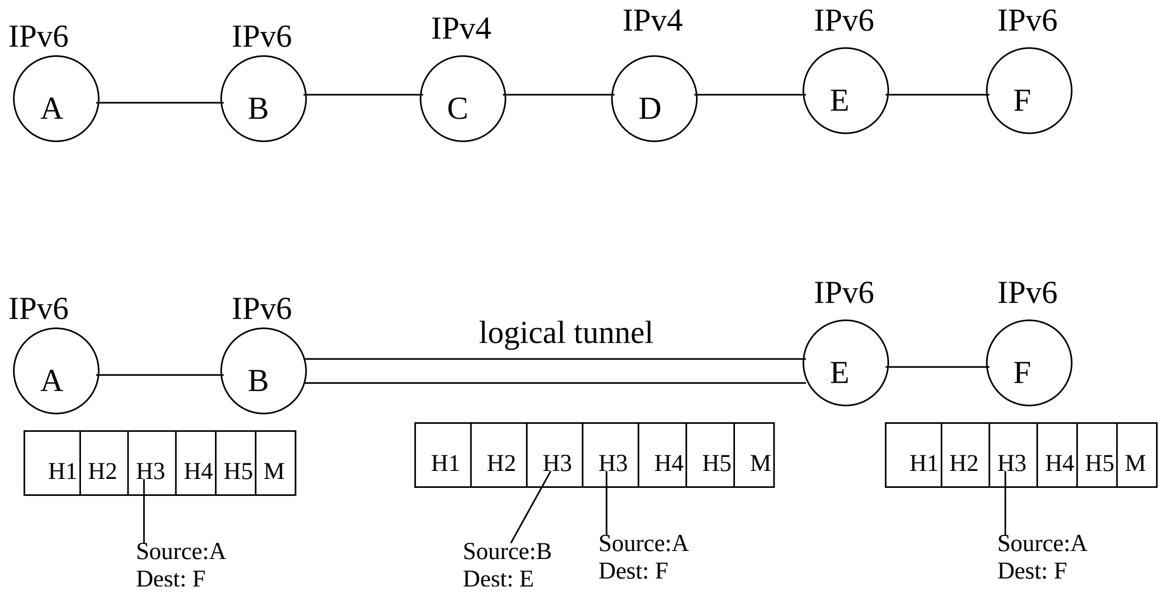

Figure 1: *Illustration of tunneling in IPv4/IPv6.*

Question: What do the headers look like for an encapsulated packet? How many headers are there?

Answer: There are now 6 headers (2 IP headers), as shown in Figure 1.

2 VPNs and NAT

The past few lectures have been all about routing. We started with the basics of routing algorithms like Distance Vector and Link State. We moved on to hierarchical routing. We then discussed several solutions to limit the size of the forwarding tables and preserve addresses. These included CIDR, Subnetting, and finally IPv6. We will now move on to more advance topics like routing through the Internet for private companies (Section 2), and routing in the backbone (Section 3).

2.1 Routing within a company

We discuss the following scenario: Imagine that you have a company (like IBM). Everyone at IBM would like to reach others at IBM. They would like to do this under the illusion that IBM has its own private routes (all within IBM) to allow anyone inhouse to reach anyone else. They would like to believe that all the packets that they send are kept very private. No one outside of IBM can ever see these packets. Another goal is that it would be nice for administrative purposes if all the IBM hosts can have IP addresses on the same network. In fact it would be even nicer if IBM could make up fake addresses for all the IBM hosts.

That is what we want to achieve.

This would not be difficult if IBM was all co-located within one building, or at least one AS. The problem is that IBM is spread out all over the place. Locations in Tokyo, locations in San Francisco, in NY, etc..

Each of these IBM locations is in a different AS.

We want to allow people within IBM to talk to each other in a secure fashion. One way to do this is of course for IBM to buy its own cables (T1 lines) and connect every machine within IBM to every other machine using IBMs own network lines. This provides security and everything else, but is of course very expensive.

Question: IBM would of course prefer to use the existing Internet links to send messages between IBM hosts. How can it do this while providing security?

Question: Also is it possible to allow all IBM hosts to have IP addresses with the same network number, or even to have made-up addresses?

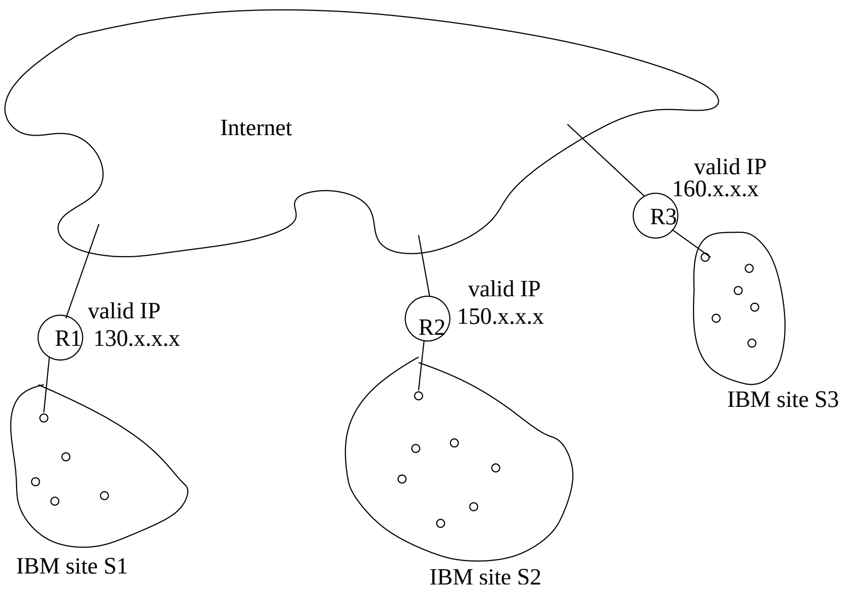

Figure 2: *Illustration of routing between IBM hosts using IBM's VPN.*

Answer: The answer to both these questions is shown in Figure 2. Figure 2 shows IBM subdivided into 3 far-away sites. Each site is connected to the Internet by a trusted router. Observe that the IP addresses of these routers are on different networks, since the sites are in different ASs. Each of these trusted routers has the function of forwarding packets from one IBM host to another far away IBM host without anyone ever seeing the contents of those packets.

How does this work?

The router R1 receives a packet from site S1 destined for some IBM address, say 129.3.3.1. Note that this destination address within IBM may not even be a valid IP addresses, or it may be a valid IP address on network 129.3.3, which IBM was able to get from IANA, and which IBM would like to use for all its IP addresses.

R1 has a table which tells it that that IBM address 129.3.3.1 is at site S3, and that that site uses secure router R3. Router R1 first **encrypts** the entire datagram. It then **encapsulates** the IBM datagram from site S1 with a new IP header which has as its source R1.ip and destination R3.ip. Observe that R1.ip and R3.ip are real (valid) IP addresses. The packet can now be sent over the Internet like a regular datagram. Anyone who peels off the IP header can only see an encrypted black box.

Question: How many headers does this packet have?

Answer: 6

When R3 gets the encapsulated datagram, it peels off the network header, **decrypts** the packet, peels the next header to obtain the IBM IP address 129.3.3.1, and sends the datagram to that address.

IBM has thus set up a **Virtual Private Network**, known as a **VPN** between its hosts. This is transparent to the hosts which are communicating among themselves. Note that this is another beautiful example of **tunneling**.

Question: What are the advantages of this scheme?

Answer: IBM can send packets from any IBM location to any other and no one can see the contents of these packets. IBM has also saved a lot of money by not having to buy IP addresses.

2.2 Routing from within company to Internet destinations outside the company

Until now we've only discussed how hosts within IBM can speak with each other in a secure way using their own addressing scheme. Suppose now one of the hosts within IBM wants to send a message to some outside Internet location (outside IBM), say it wants to retrieve a page from a web server. If the IBM machine has a valid IP address, it can send a message to anyone it likes in the Internet using the regular Internet routing protocols.

The problem is that IBM may be too cheap to purchase an IP address for each of its hosts, or, more likely, too security conscious to do so. However a packet needs a valid source IP address to traverse the Internet.

Question: What can be done?

Answer: The solution is to use **NAT** – **Network Address Translation** at the outgoing routers (R1, R2, R3) in Figure 2. NAT technology provides transparent IP-level access to the Internet from a host with a private address.

Here's how it works.

The NAT box sitting at the router bordering the IBM site has a list of **a few valid available IP addresses (perhaps a class C network number's worth of addresses)**. When a host from within the IBM site wants to send a message to someone outside IBM, the host sends the message which is intercepted by this router.

The router then **replaces** the host's private address with one of the valid IP addresses. The host application's port number is also replaced by a new port number. This router then sends the datagram (note, there's no tunneling going on here) to the destination host. Observe that the NAT-enabled router does not look like a router to the outside world. It looks like a single host with a *single* IP address.

The destination host thinks that the packet came from the router because that's the only IP address it sees. The destination host send its reply back to the router. The router must now pass this reply back to the internal IBM host.

Question: How does the NAT-enabled router know the internal IBM host to which it should forward the packet?

Answer: The NAT-enabled router has created a **NAT translation table**. When it gets the original packet from the internal host, it creates an entry that looks like:

(Internal host's IP, Internal host's port, Router's IP, Router port)

This entry is used by the router to direct packets back to the correct source host and port.

Question: Why did the NAT-enabled router have to convert the port number as well as the IP address?

Answer: Suppose that the router didn't convert the port number (just used the one given by the source host). Then the router would have to give each host its own IP address, so that it could differentiate between the hosts (because two hosts may be using the same source port). Thus if the router only has a class C network address to begin with, it would only be able to support 255 simultaneous connections from IBM to the rest of the world. By using port numbers, the NAT protocol can support 60,000 simultaneous connections (each on a different port).

NAT is becoming increasingly popular. There are many implementations of NAT out there. Please read the handouts I'm passing out and look around the Web for more info.

3 VC Routing in Backbone (IP over ATM)

3.1 Motivation

Remember that backbone routers have huge forwarding tables with 50,000 – 100,000 entries. A forwarding table lookup therefore takes a long time, even for a fast router. Furthermore, these routers typically have CIDR entries which means that we need to look through the whole table for a longest prefix match.

A datagram traversing from one country to another may need to traverse many backbone routers. Each of these router lookups will be costly.

Remember how we've always said that routers using Virtual Circuit technology are much faster at forwarding? Wouldn't it be nice if we could use Virtual Circuit technology at the backbone routers, which are especially slow?

There are 2 parts to this discussion:

1. I want to start by explaining how VC forwarding works. Why is it faster?
2. Next I'll describe how VC's are integrated into our IP-based Internet.

3.2 How VC forwarding works

Consider the abstraction of a network shown in Figure 3.

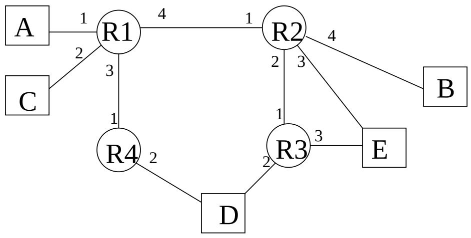

Figure 3: *Network with interfaces marked.*

In datagram routing, independent datagrams run around, each marked with an IP destination. The routers look up this destination in their forwarding tables and forward the datagram accordingly. It is conceivable that different datagrams from the same connection may go along different paths.

In VC routing, all the packets from a single connection have the same **"label"**. This label is not an IP address, but rather a VC # which specifies the connection. Routers route all packets with the same VC # label along the same path.

You'll recall that there are many reasons for doing VC routing .. such as bandwidth reservation and speed.

But how does VC routing work? How are the labels assigned? Why is the forwarding faster than under traditional packet-switching?

Let's try to figure this out with some ideas.

Idea # 1: Suppose an application on host A wants to set up a virtual circuit to an application on host D. We could assign this VC some unique VC identifier, say 19823. One could now imagine host A routing a single packet with label 19823 (connection setup packet) to host D using the traditional datagram forwarding tables. Let's say that this packet takes the following path:

$$A \to R1 \to R4 \to D$$

The packet would then establish an entry in each of the routers along the path stating that any other packet with this same label 19823 should be routed the same way (the packet may also make bandwidth and buffer reservations for VC 19823 as it moves down the path). All future packets from this same connection would carry the same label 19823 in their header. Routers would know how to forward each packet by just looking up the 19823 entry in their table. Here's how the VC table might look in a router.

Unique VC Number	Next Router
19823	R1
789	R4
98789	R2
12	R2

Figure 4: *Idea 1 for Virtual Circuit routing table structure.*

Question: What's the problem with this solution?

Answer:

1. The number of possible Virtual Circuits is huge! This means that the number of bits needed to store the Virtual Circuit number entry in the table is huge. The time required to look up a VC number in this table will be likewise huge.

2. Another problem: We need some kind of centralized source for distributing unique Virtual Circuit number.

Idea # 1 above is just way too expensive. We'd like a solution where the packet header is small and forwarding tables are small. Also, we don't want to rely on having a centralized source which generates numbers. We'd like the generation of Virtual Circuit numbers to be *distributed*!

Key observation: Although there are a lot of Virtual Circuits in the network overall, there are many fewer Virtual Circuits going through any particular router.

Idea # 2: It would be really nice if each router could make up its own Virtual Circuit numbers, independently of every other router. The router could assign the first Virtual Circuit to pass through it the number 1, the second Virtual Circuit to pass through it the number 2, the third Virtual Circuit to pass through it the number 3, etc..

Question: But how would this work? Find a solution that answers the following questions:

1. A single Virtual Circuit would has different names at different routers?
2. What should the packet then carry as its Virtual Circuit identifier?
3. Does the packet need to know the different Virtual Circuit identifiers at the different routers?
4. What happens if two different routers, say R4 and R2 above are handling two different Virtual Circuits and they both assign their Virtual Circuits the same number (say #3). Now say for both the Virtual Circuits, the next hop is to go to router R1. R1 now receives two Virtual Circuits both with the identifier #3. How can R1 differentiate between these?

(Before you read the solution, you should try to work out an answer for yourself.)

Solution: A router has several **interfaces**. For example, R1 has 4 interfaces. Assume that each router independently numbers its interfaces, as shown in Figure 3

Let's modify a router table entry to look like this:

Incoming Interface	Incoming VC#	Outgoing Interface	Outgoing VC#

Figure 5: *Router table entry for Virtual Circuit forwarding.*

We will now show how a sequence of connections is established.

Let's start with the following connection:

$$A \rightarrow R1 \rightarrow R4 \rightarrow D$$

Host A starts by assigning the connection Virtual Circuit number 1, because this is the first connection that host A has seen.

Host A sends a packet (first packet of connection) to router R1 (remember the first packet is just forwarded using datagram forwarding tables). R1 then creates the following entry in its routing table, and sends the packet to R4.

	Inc. Interf.	Inc. VC#	Outgoing Interf.	Outgoing VC#
R1:	1	1	3	1

The first 3 fields in this table are obvious. In the 4th field, R1 assigns the *lowest number that it has not yet used for that particular outgoing interface.* Since R1 hasn't sent anything on interface 3 yet, it assigns the connection an outgoing Virtual Circuit of 1.

R4 now gets the packet and creates the following entry in its table:

	Inc. Interf.	Inc. VC#	Outgoing Interf.	Outgoing VC#
R4:	1	1	2	1

R4 then forwards the packet to host D.

Observe that the Virtual Circuit number of the packet remained 1 in this example (not always true, as we'll see). All subsequent packets which start with Virtual Circuit number 1 at host A will now follow this path, because the VC forwarding tables are all set up.

Let's now suppose that another Virtual Circuit needs to be set up. This one looks like this:

$$C \to R1 \to R4 \to D$$

Again the first packet creates the VC forwarding tables. Here's how it works. Host C assigns the first packet Virtual Circuit number 1, since this is the first connection that C has seen. C then forwards the packet to R1.

R1 adds another entry to its routing table. Because R1 already has a Virtual Circuit on outgoing interface 3, it needs to assign outgoing Virtual Circuit number 2 this time.

	Inc. Interf.	Inc. VC#	Outgoing Interf.	Outgoing VC#
R1:	1	1	3	1
	2	1	3	2

R1 strips the packet's original header and replaces it with one that says Virtual Circuit number 2. Then it forwards the packet to R4.

R4 adds another entry to its routing table as follows:

	Inc. Interf.	Inc. VC#	Outgoing Interf.	Outgoing VC#
R4:	1	1	2	1
	1	2	2	2

R4 then forwards the packet to host D. From now on all connections starting at host C with Virtual Circuit number 1 will follow the above route.

Lastly, suppose we now want to set up the following Virtual Circuit.

$$A \to R1 \to R2 \to B$$

A assigns the first packet Virtual Circuit number 2 and forwards it to R1. R1 adds a third entry to its routing table as follows:

R1:

Inc. Interf.	Inc. VC#	Outgoing Interf.	Outgoing VC#
1	1	3	1
2	1	3	2
1	2	4	1

R1 changes the packet header to have a Virtual Circuit number of 1, and forwards the packet to R2.

R2 then creates the following routing table and forwards the packet to Host B.

R2:

Inc. Interf.	Inc. VC#	Outgoing Interf.	Outgoing VC#
1	1	4	1

All subsequent packets starting at Host A with Virtual Circuit number 2, will now travel this route by using these VC forwarding tables.

Some important observations:

- The first packet for the connection is forwarded using some other method (e.g., the datagram forwarding tables) which assign it a good route.
- After the first packet, however, all the remaining packets are forwarded using Virtual Circuit forwarding tables.

Question: Which do you think requires fewer bits: the Virtual Circuit number or the datagram destination address?

Answer: Almost always the Virtual Circuit number. Way fewer bits! The Virtual Circuit number at any router is only as high as the maximum number of connections which leave that particular router at a particular link. The small size of the Virtual Circuit number has many advantages!

Question: Now consider the length of the Virtual Circuit table at a router vs. the datagram routing table at a router. Which has more entries.

Answer: Usually the datagram routing table has more entries, because it needs an entry for every possible destination. Thus the Virtual Circuit table is faster to search.

Question: Why does this Virtual Circuit numbering algorithm work?

Answer: Every packet coming to router R has an incoming interface and incoming Virtual Circuit number. It turns out that this pair will give it a *unique* entry in the table, thereby allowing R to determine the appropriate outgoing interface and outgoing Virtual Circuit number.

Question: Why is the

(Incoming Interface, Incoming Virtual Circuit number)

pair unique, within a router. That is, why can't there be two entries in the router with this same starting pair?

Answer: Suppose by contradiction that the pair

(Incoming Interface = 3, Incoming Virtual Circuit number = 2)

occurred twice in router R's routing table. Consider the router R_{prec} which preceded R via interface 3. Router R_{prec} must have used the Virtual Circuit number 2 twice on the same outgoing link! But that's not allowed by our algorithm.

Question: How amenable is the Virtual Circuit system to adding on new hosts and routers?

Answer: Very amenable! Just assign whatever additional interface numbers are necessary, and you're ready to go. It doesn't affect the existing structure or tables at all.

Question: So are there any *drawbacks* to Virtual Circuits?

Answer: The problem is that with Virtual Circuits you have this extra overhead of establishing the connection in the first place. The thing that opponents argue about most though, is not the overhead of establishing the connection, but rather the issue of whether routers should be **"smart enough"** to be able to establish a connection. That is to be able to reason about what the next Virtual Circuit number that they're going to hand out will be. There are many people in the research world who believe strongly that routers should be "**dumb forwarding machines**," with no computing power at all. Then there are the other people who believe the opposite (anyone ever heard of "**active routers**?").

3.3 Integrating VCs into IP-based Internet

We have now established that VC forwarding (after the connection establishment phase) is faster than datagram forwarding. Let's now return to where VC routing is used within the Internet, given that the Internet is considered to use datagram routing.

Recall that it's the backbone routers that have the very large forwarding tables (50,000 – 100,000 entries). It would be nice if could use VC forwarding in the backbone, which would be much speedier. ATM (Asynchronous Transfer Mode) is a technology that uses VCs for routing. What's done in the backbone is that TCP/IP is run on top of ATM. That is, ATM is viewed as a link-layer.

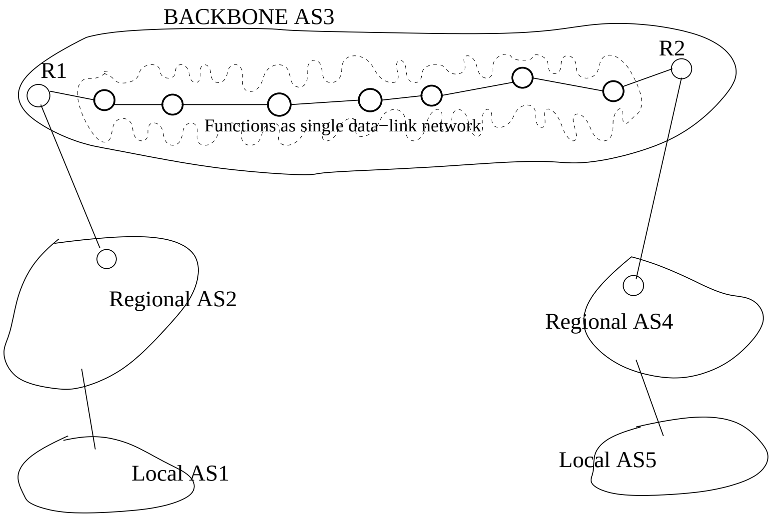

Figure 6: *IP over ATM.*

Here's how it works: Consider Figure 6. Suppose Local AS1 is sending a message to local AS5. The packets sent by local AS1 are all datagrams – they all carry the IP address of some desination host in AS5. All the routers work normally forwarding the datagram based on its destination IP, until the datagram gets to gateway R1 in the backbone. The datagram now needs to pass from R1, through the backbone routers, all the way to R2, the other gateway router in the backbone. Observe that R1 and R2 both have IP addresses. They also both have ATM addresses (think of these as link-layer addresses). The idea is to make R1 and R2 view all the routers between them as a single datalink network of type ATM. That is, R1 should think that the next IP hop is R2.

R1 looks at the destination IP address on the datagram. In R1's forwarding table, it has listed that for that IP address the next IP hop (next router) is R2. To move datagram to R2, R1 does it in the usual way, viewing ATM as just another link-layer protocol. R1 uses ARP to get the ATM address (link layer address) of R2. Then R1 passes the datagram down to link layer (i.e. ATM), where it is encapsulated with the ATM address of R2. The link layer is then responsible for getting the datagram to R2. It does this by attaching a VC label to the datagram, which is then forwarded via VC forwarding to router R2.

Note, to make things efficient permanent Virtual Circuits are used so that we don't have to go through VC establishment each time. There is a permanent VC start label associated with going between any pair of gateway routers in backbone. Note that the VC label is still switched as we pass through the routers in the backbone.

When the packet arrives at R2, R2 strips off the "datalink" header, which contains the ATM address. It sees that it is the right destination with respect to the datalink layer. It then strips off the IP header and sees that it is not the final IP destination. The R2 router now forwards the datagram in the usual Internet packet-switched style to its destination in AS5, using its destination IP.

4 ICMP – Error Reporting

Before we leave the network layer, it's important to at least mention the ICMP protocol.

Remember that IP is perfectly happy to drop packets when the going gets tough. It will drop a packet if the router doesn't know how to forward the datagram. It will drop a packet if one fragment of the packet failed to arrive. It will drop a packet if the packet's TTL field is zero.

However, when IP drops a packet, the router dropping the packet always sends back an **error/warning** message to the source host (specified in the IP packet being dropped). The protocol which sends these error messages is called **ICMP – Internet Control Message Protocol**.

Here's how it works: The router needs to send an error message back to the source host. It takes the error message and puts it in the **payload** of an IP packet. The destination of this IP packet is the original source host.

Question: How does the source host know what to do with this packet when it gets it?

Hint: How does the source host know that this is a special type of packet that's not supposed to just be passed up to the transport layer then application layer and read by some application?

Answer: In the protocol field of this packet, we don't write UDP or TCP, but rather **ICMP**. Thus the source host knows to pass the packet from IP to the ICMP protocol, where it will get handled appropriately.

Question: So what level protocol is ICMP?

Answer: One might think of ICMP as a transport protocol, since ICMP packets are carried within IP headers. However, I don't like to think of ICMP as a transport protocol, since it's not an end-to-end protocol. I prefer to think of it as sitting just slightly above IP.

Most of the ICMP messages, you as the application programmer never see. However, some you see indirectly. For example, if you try starting up a telnet session or an ftp session, you might sometimes encounter the message **"Destination network unreachable."** That message originally came from ICMP. It was passed up from ICMP, to your application.

ICMP messages are used not only for signalling error messages. The `ping` program works by having the source host send an ICMP type 8 code 0 message to the specified destination host. This type of ICMP message is an **echo request**. The destination host seeing the echo request sends back an ICMP type 0 code 0 message to the source, which is an **echo reply**.

The Kurose,Ross *Computer Networking* book lists many other examples of ICMP messages (p. 340).

The **traceroute** program relies on ICMP. The **traceroute** program allows you to trace the route of a packet from the source host to the destination host.

Question: How do you think traceroute is implemented?

Answer: Traceroute is implemented by sending a sequence of packets from the source to the destination. The first packet has a TTL of 1. The second packet has a TTL of 2. The third packet has a TTL of 3, etc. Each of these packets send back an ICMP error message when their TTL drops to 0. Along with the error message comes the name of the router sending the error message. Knowing all these router names tells us the route that the packet has taken.

5 Handouts

I am passing out some handouts to help you learn more about the material presented today. Please come see me if you would like additional information.

Thank You!