ICMP

Traceroute

There are lots of routes of more than 9 hops, so using ping to discover a route is limited; besides many routers ignore or discard this option

ICMP

Traceroute

There are lots of routes of more than 9 hops, so using ping to discover a route is limited; besides many routers ignore or discard this option

The `traceroute` program is a clever way to find routes by deliberately generating errors and looking at the ICMP messages that result

ICMP

Traceroute

There are lots of routes of more than 9 hops, so using ping to discover a route is limited; besides many routers ignore or discard this option

The `traceroute` program is a clever way to find routes by deliberately generating errors and looking at the ICMP messages that result

It sends a packet to the intended destination, but with an artificially small time-to-live

ICMP

Traceroute

When the TTL drops to zero on a hop, the packet is dropped and an ICMP “TTL exceeded” is returned by the router that dropped it

ICMP

Traceroute

When the TTL drops to zero on a hop, the packet is dropped and an ICMP “TTL exceeded” is returned by the router that dropped it

As the source address on this ICMP error is the router’s, this tells us where the packet had got to

ICMP

Traceroute

When the TTL drops to zero on a hop, the packet is dropped and an ICMP “TTL exceeded” is returned by the router that dropped it

As the source address on this ICMP error is the router’s, this tells us where the packet had got to

Repeat for increasing values of TTL to get the entire route

```
% traceroute mary.bath.ac.uk
traceroute to mary.bath.ac.uk (138.38.32.14), 30 hops max, 46 byte packets
 1  136.159.7.1 (136.159.7.1)  0.779 ms  1.131 ms  0.642 ms
 2  136.159.28.1 (136.159.28.1)  1.369 ms  0.910 ms  1.489 ms
 3  136.159.30.1 (136.159.30.1)  2.339 ms  1.937 ms  0.988 ms
 4  136.159.251.2 (136.159.251.2)  1.458 ms  1.071 ms  1.831 ms
 5  192.168.47.1 (192.168.47.1)  1.434 ms  1.554 ms  1.008 ms
 6  192.168.3.25 (192.168.3.25)  29.192 ms  30.094 ms  25.374 ms
 7  REGIONAL2.tac.net (205.233.111.67)  25.413 ms  33.002 ms  32.677 ms
 8  * * *
 9  * 117.ATM3-0.XR2.CHI6.ALTER.NET (146.188.209.182)  82.403 ms  58.747 ms
10  190.ATM11-0-0.GW4.CHI6.ALTER.NET (146.188.209.149)  56.376 ms  67.898 ms  7
11  if-4-0-1-1.bb1.Chicago2.Teleglobe.net (207.45.193.9)  66.853 ms  46.089 ms
12  if-0-0.core1.Chicago3.Teleglobe.net (207.45.222.213)  48.817 ms *  75.093 m
13  if-8-1.core1.NewYork.Teleglobe.net (207.45.222.209)  106.198 ms  94.249 ms
14  ix-5-3.core1.NewYork.Teleglobe.net (207.45.202.30)  75.286 ms  89.873 ms  9
15  us-gw.ja.net (193.62.157.13)  143.686 ms  159.212 ms  166.020 ms
16  external-gw.ja.net (193.63.94.40)  172.803 ms  189.216 ms  191.260 ms
17  external-gw.bristol-core.ja.net (146.97.252.58)  206.403 ms  185.438 ms  19
18  bristol.bweman.site.ja.net (146.97.252.102)  196.685 ms  206.221 ms  183.76
19  man-gw-2.bwe.net.uk (194.82.125.210)  197.968 ms *  174.809 ms
20  bath-gw-1.bwe.net.uk (194.82.125.198)  209.307 ms  221.512 ms  199.168 ms
21  * * *
22  mary.bath.ac.uk (138.38.32.14)  250.670 ms *  186.400 ms
```

ICMP

Traceroute

The `traceroute` command sends *three* probes for each stage so we can see time variations

ICMP

Traceroute

The `traceroute` command sends *three* probes for each stage so we can see time variations

Hop 8: no error packet was received for this TTL. There are many possible reasons, e.g., on a long route it is possible the router is setting an initial TTL on the reply that is too small to reach us

ICMP

Traceroute

The `traceroute` command sends *three* probes for each stage so we can see time variations

Hop 8: no error packet was received for this TTL. There are many possible reasons, e.g., on a long route it is possible the router is setting an initial TTL on the reply that is too small to reach us

An increasingly common possibility is that the router refuses to send ICMP errors for TTL exceeded in a weak attempt at security

ICMP

Traceroute

A $*$ before the name means the name lookup took so long traceroute decided to stop waiting and carry on. The name subsequently turned up

ICMP

Traceroute

A ∗ before the name means the name lookup took so long traceroute decided to stop waiting and carry on. The name subsequently turned up

Sometimes the same line is repeated: this is because some routers forward packets with TTL 0. This is a bug

ICMP

Traceroute

A ∗ before the name means the name lookup took so long traceroute decided to stop waiting and carry on. The name subsequently turned up

Sometimes the same line is repeated: this is because some routers forward packets with TTL 0. This is a bug

There are many bugs out there in the real world!

ICMP

Traceroute

A $*$ before the name means the name lookup took so long traceroute decided to stop waiting and carry on. The name subsequently turned up

Sometimes the same line is repeated: this is because some routers forward packets with TTL 0. This is a bug

There are many bugs out there in the real world!

Exercise Traceroute usually sends out UDP packets as probes, while some implementations use ICMP pings, while others use TCP SYNs. Find out why

ICMP

Traceroute

ICMP errors are required to contain the IP header and **at least 8 bytes of the original data** in the packet that caused the problem

ICMP

Traceroute

ICMP errors are required to contain the IP header and **at least 8 bytes of the original data** in the packet that caused the problem

This is so the OS in the source machine can match up the ICMP packet with the original packet and relay the error message back to the appropriate original application

ICMP

Traceroute

ICMP errors are required to contain the IP header and **at least 8 bytes of the original data** in the packet that caused the problem

This is so the OS in the source machine can match up the ICMP packet with the original packet and relay the error message back to the appropriate original application

There may be several applications running, sending packets, and getting ICMPs back

ICMP

Traceroute

ICMP errors are required to contain the IP header and **at least 8 bytes of the original data** in the packet that caused the problem

This is so the OS in the source machine can match up the ICMP packet with the original packet and relay the error message back to the appropriate original application

There may be several applications running, sending packets, and getting ICMPs back

Eight bytes contains the interesting parts of the next layer headers (in particular the ports of UDP and TCP) and this will be enough to identify which outgoing packet this is a reply to

ICMPv6

ICMPv6 in IPv6 plays a similar, but expanded role

ICMPv6

ICMPv6 in IPv6 plays a similar, but expanded role

For example, the ICMPv6 *Neighbour Discovery Protocol* also does the job of ARP

ICMPv6

ICMPv6 in IPv6 plays a similar, but expanded role

For example, the ICMPv6 *Neighbour Discovery Protocol* also does the job of ARP

With a *Neighbour Solicitation* request and *Neighbour Advertisement* reply

ICMPv6

ICMPv6 in IPv6 plays a similar, but expanded role

For example, the ICMPv6 *Neighbour Discovery Protocol* also does the job of ARP

With a *Neighbour Solicitation* request and *Neighbour Advertisement* reply

Exercise Read about this and compare with ARP

Routing

We now look at one of the fundamental aspects of IP: routing

Routing

We now look at one of the fundamental aspects of IP: routing

A packet does not know how to get to its destination

Routing

We now look at one of the fundamental aspects of IP: routing

A packet does not know how to get to its destination

It must rely on the routers to send it in the right direction

Routing

We now look at one of the fundamental aspects of IP: routing

A packet does not know how to get to its destination

It must rely on the routers to send it in the right direction

So how do the routers do that?

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

Routing

A router can’t possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

Routing

A router can’t possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

- The smallest number of hops

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

- The smallest number of hops
- The fastest: some links might be faster than others, e.g., undersea cable vs. satellite

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

- The smallest number of hops
- The fastest: some links might be faster than others, e.g., undersea cable vs. satellite
- The cheapest: transit is not free!

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

- The smallest number of hops
- The fastest: some links might be faster than others, e.g., undersea cable vs. satellite
- The cheapest: transit is not free!
- The most reliable

Routing

A router can't possibly know where everything in the world is: it is only connected to a handful of neighbour routers

How can a router in England know that to send a packet to Australia it might have to forward it to America first?

If there is more than one path, which should be chosen?

There are many relevant criteria:

- The smallest number of hops
- The fastest: some links might be faster than others, e.g., undersea cable vs. satellite
- The cheapest: transit is not free!
- The most reliable
- And so on (c.f., the TOS field in IP)

Routing

There is also *policy based routing*, where non-technical issues must be taken into account

Routing

There is also *policy based routing*, where non-technical issues must be taken into account

You may wish to restrict where your traffic passes through

Routing

There is also *policy based routing*, where non-technical issues must be taken into account

You may wish to restrict where your traffic passes through

At one point, there was a law in Canada that said all traffic that starts and ends in Canada must never leave Canada. Even if it would be cheaper and faster to go via the USA, say

Routing

We normally think of two classes of routing:

Routing

We normally think of two classes of routing:

- Local routing within an organisation, requiring an *interior gateway protocol* (IGP)
- Non-local routing between organisations, requiring an *exterior gateway protocol* (EGP)

Routing

We normally think of two classes of routing:

- Local routing within an organisation, requiring an *interior gateway protocol* (IGP)
- Non-local routing between organisations, requiring an *exterior gateway protocol* (EGP)

Very different requirements, with exterior protocols mostly driven by politics and economics

Host Routing

We should revisit small routing tables:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
213.121.147.69	*	255.255.255.255	UH	0	0	0	ppp0
172.18.0.0	*	255.255.0.0	U	0	0	0	eth0
172.17.0.0	*	255.255.0.0	U	0	0	0	eth1
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	213.121.147.69	0.0.0.0	UG	0	0	0	ppp0

Host Routing

We should revisit small routing tables:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
213.121.147.69	*	255.255.255.255	UH	0	0	0	ppp0
172.18.0.0	*	255.255.0.0	U	0	0	0	eth0
172.17.0.0	*	255.255.0.0	U	0	0	0	eth1
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	213.121.147.69	0.0.0.0	UG	0	0	0	ppp0

The destination address on a packet is ANDed with each netmask (`Genmask`) in turn: if the result is equal to the `Destination`, route via the given interface

Host Routing

We should revisit small routing tables:

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
213.121.147.69	*	255.255.255.255	UH	0	0	0	ppp0
172.18.0.0	*	255.255.0.0	U	0	0	0	eth0
172.17.0.0	*	255.255.0.0	U	0	0	0	eth1
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
default	213.121.147.69	0.0.0.0	UG	0	0	0	ppp0

The destination address on a packet is ANDed with each netmask (`Genmask`) in turn: if the result is equal to the `Destination`, route via the given interface

Use the first match moving from the longest mask to the shortest: top to bottom in this table

Host Routing

Flags:

Host Routing

Flags:

- U: the interface is up (i.e., working)

Host Routing

Flags:

- U: the interface is up (i.e., working)
- G: the route is to a gateway/router. Otherwise the destination is on the local network

Host Routing

Flags:

- U: the interface is up (i.e., working)
- G: the route is to a gateway/router. Otherwise the destination is on the local network
- H: the route is to a host. The destination address is a single host, not a network

Host Routing

Flags:

- U: the interface is up (i.e., working)
- G: the route is to a gateway/router. Otherwise the destination is on the local network
- H: the route is to a host. The destination address is a single host, not a network
- D: this entry was created by an ICMP *redirect*

Host Routing

Flags:

- U: the interface is up (i.e., working)
- G: the route is to a gateway/router. Otherwise the destination is on the local network
- H: the route is to a host. The destination address is a single host, not a network
- D: this entry was created by an ICMP *redirect*
- M: this entry was modified by an ICMP *redirect*

Host Routing

A *static route* is one added by hand, e.g., the `ip route add` command in Linux

Host Routing

A *static route* is one added by hand, e.g., the `ip route add` command in Linux

Routing tables on most non-routers are trivial and set up "manually" by the operating system at boot time, often with the use of DHCP

Host Routing

A *static route* is one added by hand, e.g., the `ip route add` command in Linux

Routing tables on most non-routers are trivial and set up "manually" by the operating system at boot time, often with the use of DHCP

In this context, DHCP is regarded as "setting by hand"

Host Routing

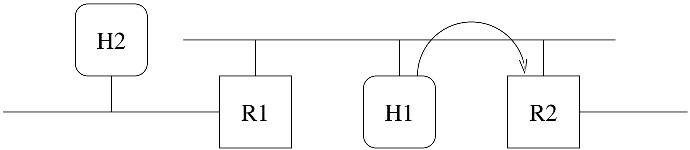
A *static route* is one added by hand, e.g., the `ip route add` command in Linux

Routing tables on most non-routers are trivial and set up "manually" by the operating system at boot time, often with the use of DHCP

In this context, DHCP is regarded as "setting by hand"

However, sometimes routing tables are not perfectly set up

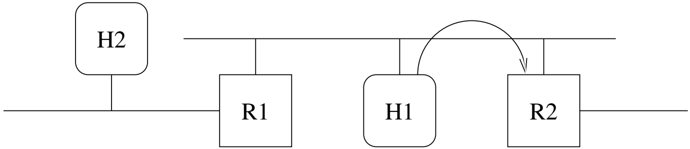
Host Routing



ICMP Redirect

H1 wants to send to H2 but H1's routing table tells it to route via R2;

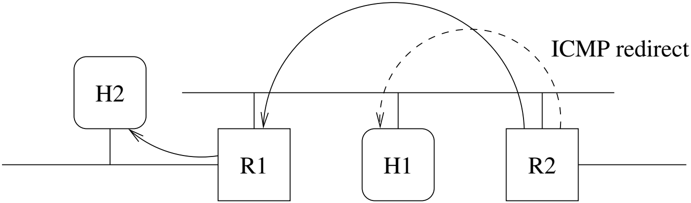
Host Routing



ICMP Redirect

When the packet reaches R2, R2 sees it should be routed out on the interface it came in on: so R2 knows H1's table needs improving;

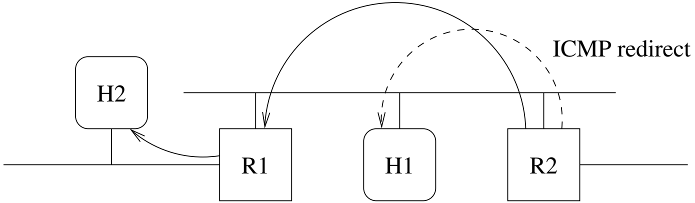
Host Routing



ICMP Redirect

R2 forwards the packet to R1 and sends an ICMP *redirect* to H1;

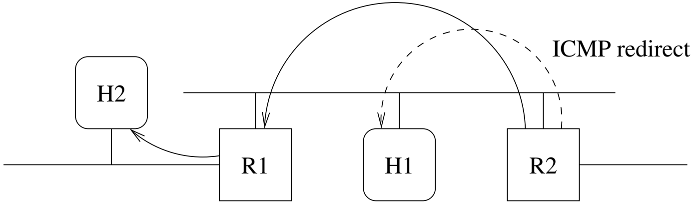
Host Routing



ICMP Redirect

H1 gets the redirect and uses it to update its routing table. The route will be marked D or M;

Host Routing



ICMP Redirect

Next time H1 will be able to route directly to R1

Routing

ICMP can modify routing tables in a small way, but is not the main way routes are set up in big routers

Routing

ICMP can modify routing tables in a small way, but is not the main way routes are set up in big routers

We could get administrators to set up the tables

Routing

ICMP can modify routing tables in a small way, but is not the main way routes are set up in big routers

We could get administrators to set up the tables

But better is to get the routers to do it themselves

Routing

ICMP can modify routing tables in a small way, but is not the main way routes are set up in big routers

We could get administrators to set up the tables

But better is to get the routers to do it themselves

Dynamic routing is the passing of routing information between routers

Dynamic Routing

There are many dynamic routing protocols:

- Routing Information Protocol (RIP)
- Open Shortest Path First (OSPF)
- Border Gateway Protocol (BGP)
- Exterior Gateway Protocol (EGP)
- And so on

Dynamic Routing

There are many dynamic routing protocols:

- Routing Information Protocol (RIP)
- Open Shortest Path First (OSPF)
- Border Gateway Protocol (BGP)
- Exterior Gateway Protocol (EGP)
- And so on

Each protocol is suited to a certain purpose, no single protocol fits all

Dynamic Routing

We start at routing the top level, namely the Internet. Local networks have different requirements

Dynamic Routing

We start at routing the top level, namely the Internet. Local networks have different requirements

The Internet is managed as a collection of *Autonomous Systems* (AS), each administered by a single entity, e.g., a University or company

Dynamic Routing

We start at routing the top level, namely the Internet. Local networks have different requirements

The Internet is managed as a collection of *Autonomous Systems* (AS), each administered by a single entity, e.g., a University or company

Between ASs run *exterior gateway protocols* (EGP), currently BGP and formerly EGP (now obsolete)

Dynamic Routing

We start at routing the top level, namely the Internet. Local networks have different requirements

The Internet is managed as a collection of *Autonomous Systems* (AS), each administered by a single entity, e.g., a University or company

Between ASs run *exterior gateway protocols* (EGP), currently BGP and formerly EGP (now obsolete)

Each AS chooses a suitable routing protocol to direct packets *within* itself: these might be interior gateway protocols, e.g., RIP and OSPF. Large institutions might even run BGP internally and have their own internal ASs

BGP

An AS is denoted by a 32 bit integer

BGP

An AS is denoted by a 32 bit integer

There are currently (2023) over 115000 ASs

BGP

An AS is denoted by a 32 bit integer

There are currently (2023) over 115000 ASs

Top-level routers will need an entry in their tables for each AS

BGP

The University of Bath is within AS786, JANET

BGP

The University of Bath is within AS786, JANET

All of JANET is one big AS: routing within JANET is an internal issue

BGP

The University of Bath is within AS786, JANET

All of JANET is one big AS: routing within JANET is an internal issue

In fact JANET runs BGP internally. Bath has internal AS64857 within JANET

BGP

BGP allows policy based routing: it’s not just the shortest or fastest path that it chooses

BGP

BGP allows policy based routing: it's not just the shortest or fastest path that it chooses

It is a *distance-vector protocol*

Routing

There are two main ways of finding routes:

- distance-vector protocols
- link-state protocols

Routing

There are two main ways of finding routes:

- distance-vector protocols
  - distance-vector
  - path-vector
- link-state protocols

And distance-vector is usually sub-divided into distance-vector and path-vector

Routing

Distance-vector gathers collections (vectors) of hop counts (distances) from its neighbouring routers to selected destinations. From this it computes its own vector of distances

Routing

Distance-vector gathers collections (vectors) of hop counts (distances) from its neighbouring routers to selected destinations. From this it computes its own vector of distances

RIP is an example of a distance-vector protocol, occasionally used in smaller networks

Routing

Distance-vector gathers collections (vectors) of hop counts (distances) from its neighbouring routers to selected destinations. From this it computes its own vector of distances

RIP is an example of a distance-vector protocol, occasionally used in smaller networks

In contrast, *link-state* gathers graphs of connectivity from all the routers (or some subset) and uses this to compute its own map. OSPF is an example

Routing

Distance-vector gathers collections (vectors) of hop counts (distances) from its neighbouring routers to selected destinations. From this it computes its own vector of distances

RIP is an example of a distance-vector protocol, occasionally used in smaller networks

In contrast, *link-state* gathers graphs of connectivity from all the routers (or some subset) and uses this to compute its own map. OSPF is an example

Distance-vector is simple, but has problems

Routing

Distance-vector gathers collections (vectors) of hop counts (distances) from its neighbouring routers to selected destinations. From this it computes its own vector of distances

RIP is an example of a distance-vector protocol, occasionally used in smaller networks

In contrast, *link-state* gathers graphs of connectivity from all the routers (or some subset) and uses this to compute its own map. OSPF is an example

Distance-vector is simple, but has problems

Link-state is more complex, but has advantages

Routing

In either case routers periodically send all or parts of their view of the world to their neighbours

Routing

In either case routers periodically send all or parts of their view of the world to their neighbours

Some protocols use broadcast, some multicast

Routing

In either case routers periodically send all or parts of their view of the world to their neighbours

Some protocols use broadcast, some multicast

A message would be "My view of the network is this..." in the case of link-state

Routing

In either case routers periodically send all or parts of their view of the world to their neighbours

Some protocols use broadcast, some multicast

A message would be “My view of the network is this. . . ” in the case of link-state

Or “I know a route to this destination using this number of hops” in the case of distance vector

Routing

The routers can then update their tables in light of this

Routing

The routers can then update their tables in light of this

For example, in distance vector:

Routing

The routers can then update their tables in light of this

For example, in distance vector:

- suppose a router knows a route (i.e., the next hop router) for a given destination of a given number n of hops

Routing

The routers can then update their tables in light of this

For example, in distance vector:

- suppose a router knows a route (i.e., the next hop router) for a given destination of a given number n of hops
- it receives a message from another neighbour that includes a route of m hops to that destination

Routing

The routers can then update their tables in light of this

For example, in distance vector:

- suppose a router knows a route (i.e., the next hop router) for a given destination of a given number n of hops
- it receives a message from another neighbour that includes a route of m hops to that destination
- if $m + 1 < n$ it can update its route to now go through that neighbour, as that is a shorter (fewer hops) route