Chapter 10 – Dynamic Routing Protocols

Motivation: Avoid manual design of a correct routing system
(which is a tough thing to do with 100,000,000 hosts!)

Other advantages of use is dynamic adaptation to:

- Failures of links
- Recovery / Additions of links
- Changes in traffic loads

Potential disadvantages:

- Overhead added by the routing protocol
- Unstable routing

Implementation in IP

Routing is still driven by tables similar in form to host tables
Routing *daemons*
- interchange info with other routing daemons
- can add/delete/modify routing table entries.

Internet is comprised of many *autonomous systems (AS)*

ftp ftp.rs.internic.net
[netinfo/asn.txt] 1–Feb–97

This file contains a list of autonomous system numbers and names of all registered ASNs. The column on the right below contains the NIC database "handle" of the coordinator for the ASN. White–pages information may be obtained about any of the ASN coodinators in this list querying the InterNIC WHOIS server. For questions or updates on this information please contact the InterNIC Registration Services Hostmaster staff, HOSTMASTER@INTERNIC.NET.

ASN Numbers

```
1        BBNPLANET          [JC347]
2        DCN-AS             [EG76]
3        MIT-GATEWAYS       [RH164]
4        ISI-AS             [JKR1]
5        SYMBOLICS          [SG52]
6        HIS-MULTICS        [JLM23]
7        UK-MOD             [RNM1]
8        RICE-AS            [SESQ]
9        CMU-ROUTER         [EAN2]
10       CSNET-EXT-AS       [WHN2]
11       HARVARD            [SB28]
:
:
```

AS numbers are allocated by the usual Internet management agencies

```
    0    Reserved  - May be use to identify non-routed networks
    1 -  1876 Allocated by Internic
 1877 -  1901 Allocated by RIPE NCC
 1902 -  2042 Allocated by Internic
         2043 Allocated by RIPE NCC
 2044 -  2046 Allocated by Internic
         2047 Allocated by RIPE NCC
 2048 -  2106 Allocated by Internic
 2107 -  2136 Allocated by RIPE NCC
 2137 -  2584 Allocated by Internic
 2585 -  2614 Allocated by RIPE NCC
 2615 -  2772 Allocated by Internic
 2773 -  2822 Allocated by RIPE NCC
 2823 -  2829 Allocated by Internic
 2830 -  2879 Allocated by RIPE NCC
 2880 -  3153 Allocated by Internic
 3154 -  3353 Allocated by RIPE NCC
 3354 -  4607 Allocated by Internic
 4608 -  4864 Allocated by AP NIC
 4865 -  5376 Allcoated by Internic
 5377 -  5631 Allocated by RIPE NCC
 5632 -  6655 Allocated by Internic
 6656 -  6911 Allocated by RIPE NCC
 6912 -  7466 Allocated by Internic
 7467 -  7722 Allocated by AP NIC
 7723 -  8191 Allocated by Internic
 8192 -  9215 Allocated by the RIPE NCC
 9216 - 10239 Allocated by the AP NIC
10240 - 11263 Allocated by the InterNic
11264 - 32767 Held by the IANA
32768 - 64511 Reserved by the IANA
64512 - 65534 Designated for private use (Allocated to the IANA)
65535         Reserved
```

Each AS can determine and use its own dynamic routing protocol

Routing protocols used within AS domains are called *interior gateway protocols (IGPs)*

Examples
RIP I & II – Routing information protocol (Bellman – Ford)
HELLO (obsolete)
OSPF – Open shortest path first (Djkstra's algorithm)

Routing protocols used between AS's are called *exterior gateway protocols (EGPs)*

Examples
EGP – Exterior gateway protocol
BGP – Border gateway protocol.

IGP's, EGP's, and routing daemons

		IGPS		***EGPS***	
daemon	`HELLO`	`RIP`	`OSPF`	`EGP`	`BGP`
routed		V1			
gated – v2	Yes	V1		Yes	
gated – v3	Yes	V1, V2	V2	Yes	V1, V2, V3

Real World Complications

Clemson originally lived in the SURANet Autonomous system.
The InfoAve AS now advertises 130.127
Clemson addresses also are reachable via Abilene
Ugh!

RIP – RFC 1058

First disseminated via the `routed` daemon in BSD Unix
Hosts may listen and use RIP broadcasts
Only routers may transmit RIP packets
RIP is called a vector–distance protocol
Also known as Bellman–Ford or Ford – Fulkerson

The basic idea underlying RIP is as follows:

Each router maintains a table with an entry for *every* other (sub)–network it knows about

Remote-net_1	cost_1,	via_1
Remote-net_2	cost_2	via_2
:	:	:
Remote-net_n	cost_n	via_n

cost is measured in hops
via is the next hop router

The `routed daemon` distributes the table
to all routers to whom it is directly attached
every 30 seconds or so

On receiving a RIP update the `routed` program processes each (dest, cost metric) in the following way:

Dest not already in routing table
==> add it with cost = cost metric + 1, via = supplier of the message

Dest already in routing table but current cost > cost metric + 1
==> replace current cost with cost metric + 1. Replace current via with source of the RIP update (router might remain current router).

Dest already in routing table, current cost < cost metric + 1 and RIP packet received from current next router for dest.
==> replace current cost with cost metric + 1. (route just got worse)

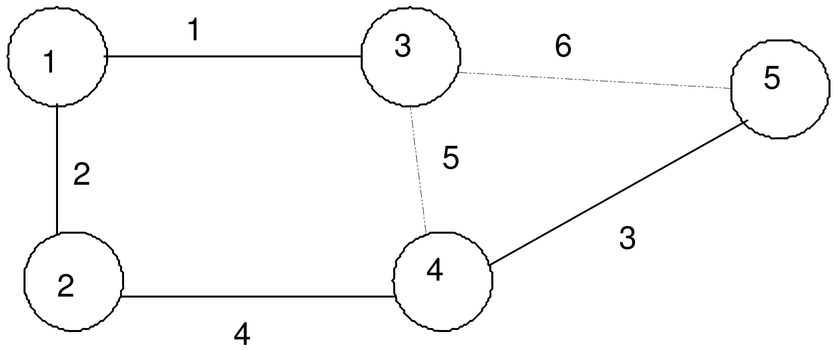
If this entry duplicates an existing entry store the time it was received.

If metrics other than "hops" are used then "1" above should be replace by cost to the sender of the RIP packet.

Entries are eliminated from the table as follows
––– When an entry is 3 minutes old mark it for deletion
––– After another minute delete it from the table

.

RIP routing table exercise



Show changes to the routing table of node 3

Assume the links are activated in the order listed

The main problem with RIP..

Responding properly to negative changes in topology

The basic algorithm generates transient routing loops

Loops are eliminated when a "count to infinity completes"

Infinity == the max diameter of the AS (16 for RIP)

Approaches that have been used to deal with the problem:

Split horizon

Never advertise a route to a gateway if your route passes through that gateway

or equivalently

Never claim reachability for a destination network to the neighbor(s) from which the route was learned

Split horizon with poisoned reverse

Continue to advertise such routes ... but use a metric of infinity

Triggered updates

When topology changes occur –– accelerate routing exchanges

==> Nodes get to infinity faster

The Counting to Infinity Problem

Suppose the A−B link becomes active after B−C, C−D, and D−E

```
     (Everyone's view of distance to A)
                                         Number of
                                         exchanges
A-------B-------C-------D-------E
       16      16        16     16          0
        1-A    16        16     16          1
        1-A     2-B      16     16          2
        1-A     2-B       3-C   16          3
        1-A     2-B       3-C    4-D        4
```

Suppose Link B−A Fails (No split horizon)

At exchange 5, B doesn't hear from A and thus uses the update from C
The count to infinity is demonstrated below
Each column indicates the column head nodes route to *A*
Values in the column are in the form (*cost, via)*

```
A-------B-------C-------D-------E
       3-C     2-B       3-C    4-D        5
       3-C     4-B/D     3-C    4-D        6
       5-C     4-B/D     5-C/E  4-D        7
       5-C     6-B/D     5-C/E  6-D        8
       7-C     6-B/D     7-C/E  6-D        9
       7-C     8-B/D     7-C/E  8-D       10
```

Suppose split horizon..and poisoned reverse are used

(At exchange 5 B no longer receives an optimistic update from C... because C's route passes through B)

```
A-------B-------C-------D-------E
       16     2-B       3-C     4-D        5
       16      16       3-C     4-D        6
       16      16        16     4-D        7
       16      16        16     16         8
```

The fatal flaw in split horizon:

When three nodes are involved in a deceptive cycle it may still be necessary to count to infinity.

```
              F
            /   \                             Number of
           /     \                            exchanges
A-------B-------C-------D-------E
```

B	F	C	D	E	Number of exchanges
16	16	16	16	16	0
1-A	16	16	16	16	1
1-A	2-B	2-B	16	16	2
1-A	2-B	2-B	3-C	16	3
1-A	2-B	2-B	3-C	4-D	4

Suppose Link B−A Fails (No split horizon)

B	F	C	D	E	
3-C/F	2-B	2-B	3-C	4-D	5
3-C/F	3-C	3-F	3-C	4-D	6
5-C/F	4-B	4-B	5-C	4-D	7
4?					

(etc... like 5 station case)

With split horizon..and poisoned reverse a really pathological count to infinity ensues (Thanks to X. Gong for pointing this one out!) –

B	C	F	D	E
16	3-F	3-C	3-C	4-D
4-C	16	16	4-C	4-D
16	16	5-B	16	5-D
16	6-F	16	16	16
7-C	16	16	7-C	16
16	16	8-B	16	8-D
16	9-F	16	16	16
10-C	16	16	10-C	16
16	16	11-B	16	11-D
16	12-F	16	16	16
13-C	16	16	13-C	16
16	16	14-B	16	14-D
16	15-F	16	16	16
16	16	16	16	16

Summary of RIP limitations:

Small networks < 15 hop diameter.. Even smaller if non unit costs are used.
Counting to infinity to resolve broken links
Fixed metrics rather than real time parameters

This situation is described in RFC 1058 which illustrates that three way deceptions aren't dealt with properly

Unfortunately, the question of how long convergence will take is not amenable to quite so simple an answer. Before going any further, it will be useful to look at an example (taken from [2]). Note, by the way, that what we are about to show will not happen with a correct implementation of RIP. We are trying to show why certain features are needed. Note that the letters correspond to gateways, and the lines to networks.

```
A-----B
 \   / \
  \ /  |
   C  /    all networks have cost 1, except
   | /     for the direct link from C to D, which
   |/      has cost 10
   D
   |<=== target network
   E
```

Each gateway will have a table showing a route to each network.

However, for purposes of this illustration, we show only the routes from each gateway to the network marked at the bottom of the diagram.

```
D:  directly connected, metric 1
B:  route via D, metric 2
C:  route via B, metric 3
A:  route via B, metric 3
```

Now suppose that the link from B to D fails. The routes should now adjust to use the link from C to D. Unfortunately, it will take a while for this to this to happen. The routing changes start when B notices that the route to D is no longer usable. For simplicity, the chart below assumes that all gateways send updates at the same time. The chart shows the metric for the target network, as it appears in the routing table at each gateway.

time ------>

Routing Exchange	**1**	**2**	**3**	**4**		**9**	**10**
D:	**dir, 1**	**dir, 1**	**dir, 1**	**dir, 1**	**...**	**dir, 1**	**dir, 1**
B:	**unreach**	**C, 4**	**C, 5**	**C, 6**		**C, 11**	**C, 12**
C:	**B, 3**	**A, 4**	**A, 5**	**A, 6**		**A, 11**	**D, 11**
A:	**B, 3**	**C, 4**	**C, 5**	**C, 6**		**C, 11**	**C, 12**

dir = directly connected
unreach = unreachable

Here's the problem: B is able to get rid of its failed route using a timeout mechanism. But vestiges of that route persist in the system for a long time. Initially, A and C still think they can get to D via B. So, they keep sending updates listing metrics of 3. In the next iteration, B will then claim that it can get to D via either A or C. Of course, it can't. The routes being claimed by A and C are now gone, but they have no way of knowing that yet. And even when they discover that their routes via B have gone away, they each think there is a route available via the other. Eventually the system converges, as all the mathematics claims it must. But it can take some time to do so. The worst case is when a network becomes completely inaccessible from some part of the system. In that case, the metrics may increase slowly in a pattern like the one

above until they finally reach infinity. For this reason, the problem is called "counting to infinity".

You should now see why "infinity" is chosen to be as small as possible. If a network becomes completely inaccessible, we want counting to infinity to be stopped as soon as possible. Infinity must be large enough that no real route is that big. But it shouldn't be any bigger than required. Thus the choice of infinity is a tradeoff between network size and speed of convergence in case counting to infinity happens. The designers of RIP believed that the protocol was unlikely to be practical for networks with a diameter larger than 15.

There are several things that can be done to prevent problems like this. The ones used by RIP are called "split horizon with poisoned reverse", and "triggered updates".

2.2.1. Split horizon

Note that some of the problem above is caused by the fact that A and C are engaged in a pattern of mutual deception. Each claims to be able to get to D via the other. This can be prevented by being a bit more careful about where information is sent. In particular, it is never useful to claim reachability for a destination network to the neighbor(s) from which the route was learned. "Split horizon" is a scheme for avoiding problems caused by including routes in updates sent to the gateway from which they were learned. The "simple split horizon" scheme omits routes learned from one neighbor in updates sent to that neighbor. "Split horizon with poisoned reverse" includes such routes in updates, but sets their metrics to infinity.

Example above with split horizon

time ------>

Routing Exchange	1	2	3	4		9	10
D:	dir, 1	dir, 1	dir, 1	dir, 1	...	dir, 1	dir, 1
B:	unreach	unreach	C, 5	C, 6		C, 11	C, 12
C:	B, 3	A, 4	A, 5	A, 6		A, 11	D, 11
A:	B, 3	C, 4	C, 5	C, 6		C, 11	C, 12

dir = directly connected
unreach = unreachable

If A thinks it can get to D via C, its messages to C should indicate that D is unreachable. If the route through C is real, then C either has a direct connection to D, or a connection through some other gateway. C's route can't possibly go back to A, since that forms a loop. By telling C that D is unreachable, A simply guards against the possibility that C might get confused and believe that there is a route through A. This is obvious for a point to point line. But consider the possibility that A and C are connected by a broadcast network such as an Ethernet, and there are other gateways on that network. If A has a route through C, it should indicate that D is unreachable when talking to any other gateway on that network. The other gateways on the network can get to C themselves. They would never need to get to C via A. If A's best route is really through C, no other gateway on that network needs to know that A can reach D. This is fortunate, because it means that the same update message that is used for C can be used for all other gateways on the same network. Thus, update messages can be sent by broadcast.

In general, split horizon with poisoned reverse is safer than simple split horizon. If two gateways have routes pointing at each other, advertising

reverse routes with a metric of 16 will break the loop immediately. If the reverse routes are simply not advertised, the erroneous routes will have to be eliminated by waiting for a timeout. However, poisoned reverse does have a disadvantage: it increases the size of the routing messages. Consider the case of a campus backbone connecting a number of different buildings. In each building, there is a gateway connecting the backbone to a local network. Consider what routing updates those gateways should broadcast on the backbone network. All that the rest of the network really needs to know about each gateway is what local networks it is connected to. Using simple split horizon, only those routes would appear in update messages sent by the gateway to the backbone network. If split horizon with poisoned reverse is used, the gateway must mention all routes that it learns from the backbone, with metrics of 16. If the system is large, this can result in a large update message, almost all of whose entries indicate unreachable networks.

2.2.2. Triggered updates

Split horizon with poisoned reverse will prevent any routing loops that involve only two gateways. However, it is still possible to end up with patterns in which three gateways are engaged in mutual deception. For example, A may believe it has a route through B, B through C, and C through A. Split horizon cannot stop such a loop. This loop will only be resolved when the metric reaches infinity and the network involved is then declared unreachable. Triggered updates are an attempt to speed up this convergence. To get triggered updates, we simply add a rule that whenever a gateway changes the metric for a route, it is required to send update messages almost immediately, even if it is not yet time for one of the regular update message. (The timing details will differ from protocol to protocol. Some distance vector protocols, including RIP, specify a small time delay, in order to avoid having triggered updates generate excessive network traffic.) Note how this combines with the rules for computing new metrics. Suppose a gateway's route to destination N goes through gateway G. If an update arrives from G itself, the receiving gateway is required to believe the new information, whether the new metric is higher or lower than the old one. If the result is a change in metric, then the receiving gateway will send triggered updates to all the hosts and gateways directly connected to it. They in turn may each send updates to their neighbors. The result is a cascade of triggered updates. It is easy to show which gateways and hosts are involved in the cascade. Suppose a gateway G times out a out to destination N. G will send triggered updates to all of its neighbors. However, the only neighbors who will believe the new information are those whose routes for N go through G. The other gateways and hosts will see this as information about a new route that is worse than the one they are already using, and ignore it. The neighbors whose routes go through G will update their metrics and send triggered updates to all of their neighbors. Again, only those neighbors whose routes go through them will pay attention. Thus, the triggered updates will propagate backwards along all paths leading to gateway G, updating the metrics to infinity. This propagation will stop as soon as it reaches a portion of the network whose route to destination N takes some other path.

If the system could be made to sit still while the cascade of triggered updates happens, it would be possible to prove that counting to infinity will never happen. Bad routes would always be removed immediately, and so no routing loops could form.

Unfortunately, things are not so nice. While the triggered updates are being sent, regular updates may be happening at the same time. Gateways that haven't received the triggered update yet will still be sending out information based on the route that no longer exists. It is possible that after the triggered update has gone through a gateway, it might receive a normal update from one of these gateways that hasn't yet gotten the word. This could reestablish an orphaned remnant of the faulty route. If triggered updates happen quickly enough, this is very unlikely. However, counting to infinity is still possible.

The limitations of RIP are summarized in RFC 1058

- The protocol is limited to networks whose longest path involves 15 hops. The designers believe that the basic protocol design is inappropriate for larger networks. Note that this statement of the limit assumes that a cost of 1 is used for each network. This is the way RIP is normally configured. If the system administrator chooses to use larger costs, the upper bound of 15 can easily become a problem.

- The protocol depends upon "counting to infinity" to resolve certain unusual situations. (This will be explained in the next section.) If the system of networks has several hundred networks, and a routing loop was formed involving all of them, the resolution of the loop would require either much time (if the frequency of routing updates were limited) or bandwidth (if updates were sent whenever changes were detected). Such a loop would consume a large amount of network bandwidth before the loop was corrected. We believe that in realistic cases, this will not be a problem except on slow lines. Even then, the problem will be fairly unusual, since various precautions are taken that should prevent these problems in most cases.

- This protocol uses fixed "metrics" to compare alternative routes. It is not appropriate for situations where routesneed to be chosen based on real-time parameters such a measured delay, reliability, or load. The obvious extensions to allow metrics of this type are likely to introduce instabilities of a sort that the protocol is not designed to handle.

RIP−I Mechanism

RIP packets

Sent/received as UDP datagrams

Port 520 is reserved for use by the routed server

| IP Hdr | UDP Hdr | RIP data

RIP data

```
 0                   1                   2                   3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| command (1)   | version (1)   |      must be zero (2)         |
+---------------+---------------+-------------------------------+
| address family identifier (2) |      must be zero (2)         |
+-------------------------------+-------------------------------+
|                         IP address (4)                        |
+---------------------------------------------------------------+
|                        must be zero (4)                       |
+---------------------------------------------------------------+
|                        must be zero (4)                       |
+---------------------------------------------------------------+
|                           metric (4)                          |
+---------------------------------------------------------------+
```

Last 20 bytes shown may be repeated up to 25 times for a total RIP payload of 504

Command

1 − Request

2 − Response

A true response to a request

A periodic update (every 30 seconds or so)

An update generated in response to a change (link goes up or down)

Version

1 − Shown here

2 − Supports subnet masks

Address family

2 − IP address

32−bit destination address (network or host)

Metric

distance to destination in hops max count = 15

16 = infinite distance.. used to react to down links.

Normal operation of `routed`

Initialization
    for all active interfaces
        send RIP request (network broadcast if supported)

    Request packet contents
        command – 1
        addr family – 0
        metric – 16

Request received
    if special type of request above
        return complete routing table
    else
        for all entries in the request
            if we have entry
                set cost–metric to our cost
            else
                set cost to 16
    return updated table

Response received
    update local table as described earlier

Regular update
    Approx every 30 seconds
    Send all (or part) of routing table

Triggered updates
    When any metric changes
    Distribute the changes to neighbors

RIP Examples

(On our systems (in 1995) citron, atlantic, plato, and diogenes participate in RIP)

This is the RIP packet generated by Citron

```
---------------------- #:4 ---------------------------------
  Delta Time:  4.500   Packet Length: 546 bytes (222 hex)
 DIX:   Dest: FF:FF:FF:FF:FF:FF   Source: 08:00:20:21:95:DD
 DIX:   Dest: 130.127.048.000.    Source: 130.127.048.001.
----------------------- IP HEADER -----------------------
 IP:  Version: 4 Correct    Header Length: 20 bytes
 IP:  Type Of Service: 00
 IP:     000. ....  Routine
 IP:     ...0 ....  Normal Delay
 IP:     .... 0...  Normal Throughput
 IP:     .... .0..  Normal Reliability
 IP:  Total Len: 532 (x214) bytes          Id: F7AD
 IP:  Flags: 0
 IP:     .0..       May Fragment
 IP:     ..0.       Last Fragment
 IP:  Fragment Offset: 000
 IP:  Time To Live: 60 sec    Protocol: 11 (UDP)
 IP:  Header Checksum: 202C
 IP:  No Options
---------------------- UDP HEADER ----------------------
 UDP:  Source Port: 520 (RouteD)     Dest Port: 520 (RouteD)
 UDP:  Length: 512 (x200)
 UDP:  Checksum: 0
----------------------- RIP Packet -----------------------
 RIP:  Command: 2    Response
 RIP:  Version: 1
 RIP:  IP Address: 130.127.210.064.    Metric: 5
 RIP:  IP Address: 130.127.192.000.    Metric: 2
 RIP:  IP Address: 000.000.000.000.    Metric: 4 <---- ???????
 RIP:  IP Address: 130.127.032.000.    Metric: 5
 RIP:  IP Address: 130.127.096.000.    Metric: 4
 RIP:  IP Address: 130.127.128.000.    Metric: 4
 RIP:  IP Address: 130.127.160.000.    Metric: 4
 RIP:  IP Address: 130.127.224.000.    Metric: 4
 RIP:  IP Address: 130.127.034.000.    Metric: 4
 RIP:  IP Address: 130.127.098.000.    Metric: 5
 RIP:  IP Address: 130.127.002.000.    Metric: 2
 RIP:  IP Address: 130.127.130.000.    Metric: 2
 RIP:  IP Address: 130.127.162.000.    Metric: 3
 RIP:  IP Address: 130.127.194.000.    Metric: 4
 RIP:  IP Address: 130.127.226.000.    Metric: 4
 RIP:  IP Address: 192.221.004.000.    Metric: 4
 RIP:  IP Address: 130.127.004.000.    Metric: 3
 RIP:  IP Address: 130.127.036.000.    Metric: 2
 RIP:  IP Address: 130.127.068.000.    Metric: 3
 RIP:  IP Address: 130.127.100.000.    Metric: 3
```

The packet generated by Diogenes has only a single entry

Why?
Split horizon mechanism... Not the lack of routing table entries

```
--------------------------------                               #:10
--------------------------------
  Delta Time:  1.219   Packet Length: 66 bytes (42 hex)
 DIX:   Dest: FF:FF:FF:FF:FF:FF   Source: 08:00:20:11:60:8B
 DIX:   Dest: 130.127.048.000.    Source: 130.127.048.004.
----------------------- IP HEADER -----------------------
 IP:  Version: 4 Correct    Header Length: 20 bytes
 IP:  Type Of Service: 00
 IP:     000. ....  Routine
 IP:     ...0 ....  Normal Delay
 IP:     .... 0...  Normal Throughput
 IP:     .... .0..  Normal Reliability
 IP:  Total Len: 52 (x34) bytes          Id: 0A6B
 IP:  Flags: 0
 IP:     .0..       May Fragment
 IP:     ..0.       Last Fragment
 IP:  Fragment Offset: 000
 IP:  Time To Live: 60 sec    Protocol: 11 (UDP)
 IP:  Header Checksum: 0F4C
 IP:  No Options
---------------------- UDP HEADER -----------------------
 UDP:  Source Port: 520 (RouteD)     Dest Port: 520 (RouteD)
 UDP:  Length: 32 (x20)
 UDP:  Checksum: 0
----------------------- RIP Packet -----------------------
 RIP:  Command: 2    Response
 RIP:  Version: 1
 RIP:  Address Family Identifier: 2
 RIP:  IP Address: 130.127.066.000.    Metric: 1
```

The packet generated by Plato is similar to that of Diogenes

```
---------------------------------                                   #:11
---------------------------------
  Delta Time:  8.000   Packet Length: 66 bytes (42 hex)
 DIX:   Dest: FF:FF:FF:FF:FF:FF   Source: 08:00:20:73:F9:A8
 DIX:   Dest: 130.127.048.000.    Source: 130.127.048.002.
----------------------- IP HEADER -----------------------
 IP:  Version: 4 Correct    Header Length: 20 bytes
 IP:  Type Of Service: 00
 IP:     000. ....  Routine
 IP:     ...0 ....  Normal Delay
 IP:     .... 0...  Normal Throughput
 IP:     .... .0..  Normal Reliability
 IP:  Total Len: 52 (x34) bytes          Id: D972
 IP:  Flags: 0
 IP:     .0..       May Fragment
 IP:     ..0.       Last Fragment
 IP:  Fragment Offset: 000
 IP:  Time To Live: 60 sec    Protocol: 11 (UDP)
 IP:  Header Checksum: 4046
 IP:  No Options
---------------------- UDP HEADER ----------------------
 UDP:  Source Port: 520 (RouteD)     Dest Port: 520 (RouteD)
 UDP:  Length: 32 (x20)
 UDP:  Checksum: 0
----------------------- RIP Packet -----------------------
 RIP:  Command: 2    Response
 RIP:  Version: 1
 RIP:  Address Family Identifier: 2
 RIP:  IP Address: 130.127.070.000.    Metric: 1
```

However the packet generated by Atlantic looks like the one generated by Citron

Why?

Atlantic isn't using split horizon... or

Atlantic isn't routing through Citron

How can we tell

Look at metrics to common destinations

They are the same ==>

Atlantic is using split horizon and

is not routing through Citron

Thus Atlantic is directly connected to the campus backbone

```
-------------------------------- #:13 -------------------
  Delta Time:  0.906   Packet Length: 546 bytes (222 hex)
 DIX:   Dest: FF:FF:FF:FF:FF:FF   Source: 08:00:20:04:F0:01
 DIX:   Dest: 130.127.048.255.    Source: 130.127.048.005.
----------------------- IP HEADER -----------------------
 IP:  Version: 4 Correct    Header Length: 20 bytes
 IP:  Type Of Service: 00
 IP:     000. ....  Routine
 IP:     ...0 ....  Normal Delay
 IP:     .... 0...  Normal Throughput
 IP:     .... .0..  Normal Reliability
 IP:  Total Len: 532 (x214) bytes          Id: 6B23
 IP:  Flags: 2
 IP:     .1..       Don't Fragment
 IP:     ..0.       Last Fragment
 IP:  Fragment Offset: 000
 IP:  Time To Live: 01 sec    Protocol: 11 (UDP)
 IP:  Header Checksum: A6B3
 IP:  No Options
---------------------- UDP HEADER ----------------------
 UDP:  Source Port: 520 (RouteD)     Dest Port: 520 (RouteD)
 UDP:  Length: 512 (x200)
 UDP:  Checksum: AAAF
----------------------- RIP Packet -----------------------
 RIP:  Command: 2    Response
 RIP:  Version: 1
 RIP:  IP Address: 130.127.210.064.    Metric: 5
 RIP:  IP Address: 130.127.160.000.    Metric: 4
 RIP:  IP Address: 130.127.128.000.    Metric: 4
 RIP:  IP Address: 130.127.096.000.    Metric: 4
 RIP:  IP Address: 130.127.032.000.    Metric: 5
 RIP:  IP Address: 000.000.000.000.    Metric: 4
 RIP:  IP Address: 130.127.192.000.    Metric: 2
 RIP:  IP Address: 130.127.224.000.    Metric: 4
 RIP:  IP Address: 130.127.034.000.    Metric: 4
 RIP:  IP Address: 130.127.098.000.    Metric: 5
 RIP:  IP Address: 130.127.194.000.    Metric: 4
 RIP:  IP Address: 130.127.162.000.    Metric: 3
 RIP:  IP Address: 130.127.130.000.    Metric: 2
 RIP:  IP Address: 130.127.002.000.    Metric: 2
```

RIP−2 (RFC 1388 − 1993)

RIP−2 motivated by the advantages of RIP (in small network)

Low overhead in bandwidth consumed
Easy to configure and manage (compared to newer IGP's)

Main advantage of RIP−2

Support for subnetted addresses

RIP − 2 Packet format

Like RIP − 1 but also has the next hop addresses filled in..

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| command (1)   | version (1)   |      Routing Domain           |
+---------------+---------------+-------------------------------+
| address family identifier (2) |      Route tag                |
+-------------------------------+-------------------------------+
|                         IP address (4)                        |
+---------------------------------------------------------------+
|                        32 bit subnet mask                     |
+---------------------------------------------------------------+
|                        32 bit next hop IP address             |
+---------------------------------------------------------------+
|                          metric (4)                           |
+---------------------------------------------------------------+
```

Routing domain
PID of the routing daemon to which packet belongs (routed)
Route tag
Used to support EGP's it is the AS number of the router that generated it.
Subnet mask
That of the IP address above
Next hop address
Normally 0... but can be used to provide an ICMP−redirect−like improved route.

(Weak) Authentication scheme
Addr family 0xffff
Route tag 2
==> next 16 bytes carry *cleartext* password!

Multicast rather than broadcast distribution is also used.

HELLO
A RIP like protocol used in the original ARPANET backbone
Routing metrics in HELLO were instantaneous delays

OSPF−2 (RFC 1247/1583)

Based upon *link state* as opposed to *distance vector* routing computations

Claimed advantages:

Includes explicit support for

IP subnetting,

Type Of Service − based routing
Each interface can be assigned a cost for each TOS
Loads can be balanced across equal cost routes
Traffic is equally distributed when "ties" occur.
But not proportionally distributed when they do not.

Tagging of externally−derived routing information.
(e.g., routes learned from the Exterior Gateway Protocol (EGP)) is passed transparently throughout the Autonomous System. This externally derived data is kept separate from the OSPF protocol's link state data. Each external route can also be tagged by the advertising router, enabling the passing of additional information between routers on the boundaries of the Autonomous System.

Provides for the authentication of routing updates
Point−to−point links no longer require IP addresses at each end.
Utilizes IP multicast when sending/receiving the updates.
Responds quickly to topology changes,
Requires involves small amounts of routing protocol traffic.
(Uses multicast instead of broadcast routing).

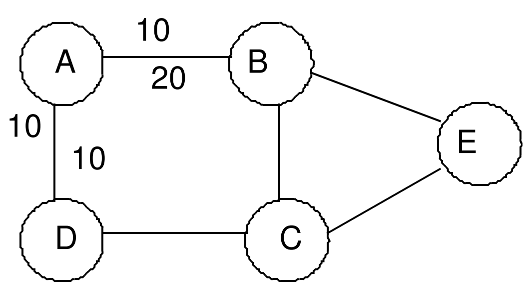
Each router maintains a database describing the Autonomous System's topology.
All routers construct (roughly) the same database.
Each routers local state can be considered a table with the following entries

Interface	Destination	Type of Service	Cost
le0	A	1	10
le0	A	2	20
le1	D	1	21
le2	C	3	30

Periodically each node sends this info *to every router in the AS* by flooding
Each router in the AS is then able to fill in a cost/reachability matrix for each type of service

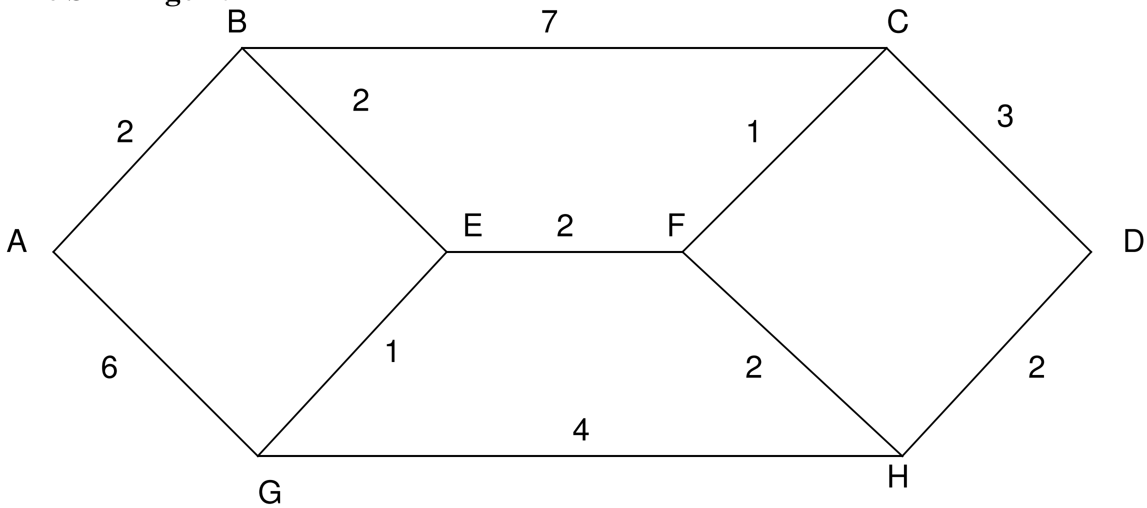
	A	B	C	D	E
A	–	10		10	
B	20	–	15		25
C		20	–	6	
D	10		10	–	12
E		30	20		–

One of these tables exists for *each* type of service



For each class of service the router then computes a routing tree.
The contents of the routing tree allow the construction of the routing tables.
Note: old format routing table has to get dumped!
Traffic is equally distributed when "ties" occur.
But not proportionally distributed when they do not.

The SPF Algorithm



Step 1: Construct cost table for all nodes:
Each station sends delays to each of its connected neighbors
Messages are sent via flooding in IP
Each message fills in one row of the table.

	A	B	C	D	E	F	G	H
A	0	2					6	
B	2	0	7		2			
C		7	0	3		1		
D			3	0				2
E		2			0	2	1	
F			1		2	0		2
G	6				1		0	4
H				2		2	4	0

Step 2: Initialize the routing table:

Initial routing table for source A:

Dest	Via	Cost	Status
B	?	∞	N/A
C	?	∞	N/A
D	?	∞	N/A
E	?	∞	N/A
F	?	∞	N/A
G	?	∞	N/A
H	?	∞	N/A

Step 3: Make N passes over the routing table adding a single node for each pass.

After Pass 1

Dest	Via	Cost	Status
B	B	2	Perm*
C	?	∞	N/A
D	?	∞	N/A
E	?	∞	N/A
F	?	∞	N/A
G	G	6	Temp
H	?	∞	N/A

After Pass 2

Dest	Via	Cost	Status
B	B	2	Perm
C	BC	9	Temp
D	?	∞	N/A
E	BE	4	Perm*
F	?	∞	N/A
G	G	6	Temp
H	?	∞	N/A

After Pass 3

Dest	Via	Cost	Status
B	B	2	Perm
C	BC	9	Temp
D	?	∞	N/A
E	BE	4	Perm
F	BEF	6	Temp
G	BE	5	Perm*
H	?	∞	N/A

After Pass 4

Dest	Via	Cost	Status
B	B	2	Perm
C	BC	9	Temp
D	?	∞	N/A
E	BE	4	Perm
F	BEF	6	Perm*
G	BE	5	Perm
H	BEGH	9	Temp

After Pass 5

Dest	Via	Cost	Status
B	B	2	Perm
C	BEFC	7	Perm*
D	?	∞	N/A
E	BE	4	Perm
F	BEF	6	Perm
G	BE	5	Perm
H	BEFH	8	Temp

After Pass 6

Dest	Via	Cost	Status
B	B	2	Perm
C	BEFC	7	Perm
D	BEFCD	10	Temp
E	BE	4	Perm
F	BEF	6	Perm
G	BE	5	Perm
H	BEFH	8	Perm*

After Pass 7

Dest	Via	Cost	Status
B	B	2	Perm
C	BEFC	7	Perm
D	BEFCD	10	Perm*
E	BE	4	Perm
F	BEF	6	Perm
G	BE	5	Perm
H	BEFH	8	Perm

Analysis of Distributed SPF routing:

N = Number of stations.
M = Average number of links

Complexity of computation performed by each node $O(N^2)$.
Number of messages interchanged $O(N^2)$
Length of each message $O(M)$
Amount of data interchanged $O(N^2M)$

OSPF Implementation –

The topological database

Entries include networks and routers

OSPF network types

Point–to–point	Joins a single pair of Routers (Dedicated T1 Link)
Broadcast network	Many attached routers reachable by broadcast (Ethernet)
Non–Broadcast nets	Many attached routers but no broadcast (X.25 PDN)

The topological DB is represented as a di–graph

Vertices
Routers and networks

Edges
Connect routers on point–to–point networks
Routers *to* non–point–to–point networks

Costs
assigned to each router interface
cost is configurable by system administrator
lower cost => more likely to be used
no labeled cost => cost = 0 (e.g connections from networks to routers)

Policies vs. Mechanisms

The OSPF protocol provides a *mechanism* for
distributing routing information
computing routing tables based upon the costs received.

The underlying routing *policies* are
defined by the semantics of "cost"

This situation is analogous to dispatching within an Unix system
the unix scheduler provides a mechanism
the system administrator can set policies by assigning priorities
no single *policy* may be optimal for all environments (e.g. workstation vs. large time shared system.
however one decently designed *mechanism* may be suitable.

Inconsistent routing policies within an AS should (and can) be avoided (since an AS is basically an administrative domain.)

Possible ways to assign costs:

Hops
Hops normalized by link speed
Dynamic delay (see McQuillen's paper: *The New Routing Algorithm for the ARPANet −IEEE Trans Comm. COM−28.*

Each node measures average delay experienced by outgoing packets every 10 sec.
Delay = sent time − arrival time + propagation delay + transmission delay
Significant change in delay causes a routing update to be generated.
Threshold is initially 64 ms.
Reduced by 12.8 ms each 10 sec.
Becomes 0 after a minute.

or see Khanna and Zinky's: *The "Revised" ARPANET Routing Metric − Proc Sigcomm 91.*

In reality an OSPF−2 graph has 3 distinct vertex types

Vertex type	Vertex name	Transit?
1	Router	yes
2	Network	yes
3	Stub network	no

Table 1: OSPF vertex types.

The three types of connection are shown in figure 1 of RFC 1583

A complete example autonomous system is shown in figure 2 of RFC 1583

H1 is a host connected by SLIP
RT12 is advertising a host route
Only point−to−point network with endpoint addresses is RT6 − RT10
RT5 and RT7 have E(B)GP connections to the rest of the internet.
The associated di−graph is shown in figure 3

Cost distribution

Each router distributes a link state packet giving cost of its links
Each network has an *elected* designated router that performs the task.

An example of link state advertisment is shown in figure 4.

The entire routing tree for RT6 is shown in figure 5
RT6 knows the optimal path to EVERY destination
But the tree is only used to route to the next hop (no full path source route)
===> inconsistent trees may lead to transient routing loops.

Externally originated cost metrics −
- Type 1 −
  - Cost to destination is computed as
    - cost to border router + externally originated cost metric
- Type 2−
  - Cost to destination is computer as
    - externally originated cost metric
    - internal cost is used as a tie−breaker
- Type 1 metrics have precedence over type 2

Boundary routers participating in OSPF can specify forwarding addresses in their advertisements to eliminate the extra hop problem (for example to other Boundary –– but not OSPF routers)

Furthermore OSPF allows AS's to be further broken down into *areas* that must be connected by a single *backbone.*

Support for areas is –– ***ugly*** –– see RFC 1583 if you dare!

OSPF Data Structures

OSPF packets are transmitted as IP datagrams using their own protocol number
Each packet carries a common header as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Version #   |     Type      |         Packet length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                          Router ID                            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                           Area ID                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |             Autype            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Authentication                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Authentication                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The *Router ID* is a 32 bit number that uniquely identifies the router within the AS.
The specification suggests using the numerically lowest IP address of all the routers links.

The contents of the type field are as follows

Type	Description
1	Hello
2	Database Description
3	Link State Request
4	Link State Update
5	Link State Acknowledgment

Link state updates:

All link state advertisements begin with a common 20 byte header.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            LS age             |    Options    |    LS type    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Link State ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     Advertising Router                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     LS sequence number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         LS checksum           |             length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

LS age
The time in seconds since the link state advertisement was originated.

Options
The optional capabilities supported by the described portion of the routing domain. OSPF's optional capabilities are documented in Section A.2.

LS type
The type of the link state advertisement. Each link state type has a separate advertisement format. The link state types are as follows (see Section 12.1.3 for further explanation):

LS Type	Description
1	Router links
2	Network links
3	Summary link (to IP networks outside the AREA but within AS)
4	Summary link (to ASBR Autonomous System Boundary Routers)
5	AS external link (Destinations in another AS –– originated by ASBRs

Link State ID (*The from index in the routing matrix)*
This field identifies the portion of the internet environment that is being described by the advertisement. The contents of this field depend on the advertisement's LS type. For example, in network links advertisements the Link State ID is set to the IP interface address of the network's Designated Router (from which the network's IP address can be derived). The Link State ID is further discussed in Section 12.1.4.

Advertising Router
The Router ID of the router that originated the link state advertisement. For example, in network links advertisements this field is set to the Router ID of the network's Designated Router.

LS sequence number

Detects old or duplicate link state advertisements. Successive instances of a link state advertisement are given successive LS sequence numbers. See Section 12.1.6 for more details.

LS checksum
The Fletcher checksum of the complete contents of the link state advertisement. See Section 12.1.7 for more details.

length
The length in bytes of the link state advertisement. This includes the 20 byte link state header.

Router link advertisements

In router link advertisements, the Link State ID field is set to the router's OSPF Router ID. The T–bit is set in the advertisement's Option field if and only if the router is able to calculate a separate set of routes for each IP TOS. Router links advertisements are flooded throughout a single area only.

```
  0                   1                   2                   3
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |            LS age             |     Options   |       1       |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                        Link State ID                          |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     Advertising Router                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                     LS sequence number                        |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |         LS checksum           |             length            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      0      |E|B|      0      |            # links            |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Link ID                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Link Data                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |     Type      |     # TOS     |        TOS 0 metric           |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      TOS      |        0      |            metric             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                              ...                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |      TOS      |        0      |            metric             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                          Link ID                              |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |                         Link Data                             |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

bit E
When set, the router is an AS boundary router (E is for external)

bit B
When set, the router is an area border router (B is for border)

links
The number of router links described by this advertisement. This must be the total collection of router links to the area.

The following fields are used to describe each router link. Each router link is typed (see the below Type field). The type field indicates the kind of link being described. It may be a link to a transit network, to another router or to a stub network. The values of all the other fields describing a router link depend on the link's type. For example, each link has an associated 32−bit data field. For links to stub networks this field specifies the network's IP address mask. For the other link types the Link Data specifies the router's associated IP interface address.

Type

A quick description of the router link. One of the following. Note that host routes are classified as links to stub networks whose network mask is 0xffffffff.

Type	Description
1	Point−to−point connection to another router
2	Connection to a transit network
3	Connection to a stub network
4	Virtual link

Link ID *(the "to" index in the routing matrix)*

Identifies the object that this router link connects to. Value depends on the link's type (above). When connecting to an object that also originates a link state advertisement (i.e., another router or a transit network) the Link ID is equal to the other advertisement's Link State ID. This provides the key for looking up said advertisement in the link state database. See Section 12.2 for more details.

Type	Link ID
1	Neighboring router's ID
2	IP address of Designated Router
3	IP network/subnet number
4	Neighboring router's ID

Link Data

Contents again depend on the link's Type field. For connections to stub network, it specifies the network mask. For the other link types it specifies the router's associated IP interface address. This latter piece of information is needed during the routing table build process, when calculating the IP address of the next hop. See Section 16.1.1 for more details.

#metrics

The number of different TOS metrics given for this link, not counting the required metric for TOS 0. For example, if no additional TOS metrics are given, this field should be set to 0.

TOS 0 metric

The cost of using this router link for TOS 0.

For each link, separate metrics may be specified for each Type of Service (TOS). The metric for TOS 0 must always be included, and was discussed above. Metrics for non−zero TOS are described below. The encoding of TOS in OSPF link state advertisements is described in Section 12.3. Note that the cost for non−zero TOS values that are not specified defaults to the TOS 0 cost. Metrics must be listed in order of increasing TOS encoding. For example, the metric for TOS 16 must always follow the metric for TOS 8 when both are specified.

TOS IP type of service that this metric refers to. The encoding of TOS in OSPF link state advertisements is described in Section 12.3.

metric

The cost of using this outbound router link, for traffic of the specified TOS.

Exterior Gateway Protocols

Internet Routing Background

Routing with partial information –– default routes

The key to internet routing
You don't have to know how to get where you're going
You just have to know how to get to someone who does!

Early Internet routers belonged to 2 classes

Core – Controlled by Internet Network Operations Center (INOC)
Were responsible for providing
Authoritative, consistent, correct routes for *all* destination networks.

Other – Controlled by anyone else
Were responsible for advertising themselves to the core
But only had to deliver cross "AS" traffic to a core router to "ensure" its delivery

Possible approaches to routing within the core

Default route based
A loop of default routes could be set up within INOC routers
Advantage
Simple routing tables
Disadvantage
The extra hop(s) problem.. (logically a routing ring)

Explicit route based
Advantage
Faster routing
Disadvantage
Core routers had to know the *whole* Internet
Each router's address table needed to have every network address

The resolution
Use explict routes (EGP)

Complications
- Peer backbones
  - ARPANET
  - NFSNET
- The problem
  - Possible routing loops (esp for non−existent destinations).

The resolution
- (Try to) avoid peer backbones
- or have a single core router system encompassing all backbones

The EGP Protocol

EGP routers communicate across AS boundaries to designated EGA neighbors

Neighbors should be
- 1 IP hop away
- Configured by system administrators to speak to each other

Designed to propogate rechability data across the internet
Assumes a single core backbone
Uses raw IP datagrams − w/ Protocol # = 8.
Core routers can advertise rechability of nets they have learned about
Non−core routers can advertise only networks that are within their own AS

Limitations of EGP

Reachability in EGP is absolute.. I provide *the* path to the network you seek.
Topology of an EGP based internet is *necessarily a tree with the core AS at root*

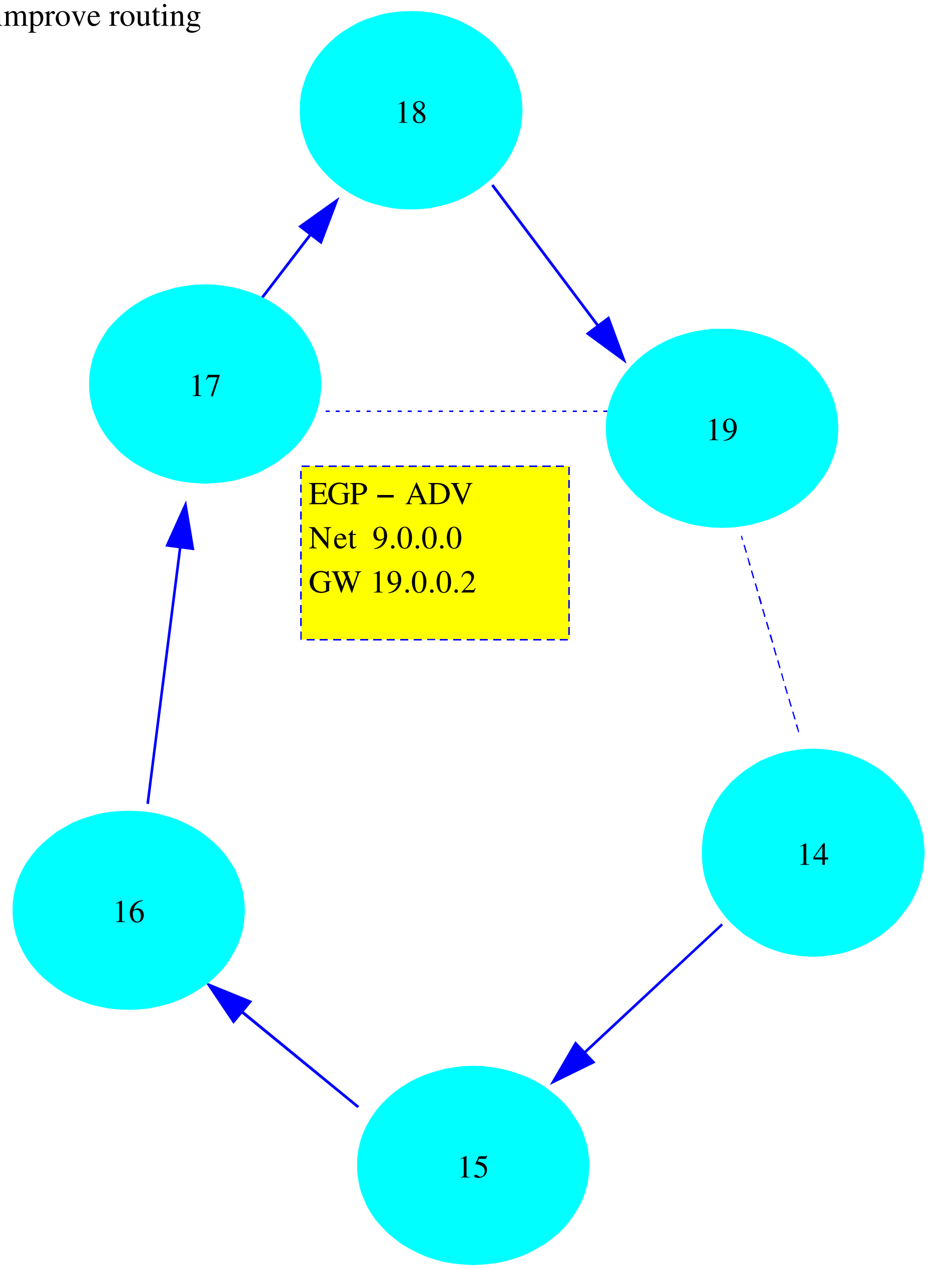
Implications
- There is a single *core* AS.
- Core system failure => internet failure
- EGP can advertise only one path to a given network
- EGP does not support load sharing when multiple routers connecting 2 AS's
- EGP was not suitable for combining ARPANET and NFSNET (but a solution was hacked together by subdividing NFSNET into multiple AS's.

The Fatal Flaw in EGP

Possible actions

Accept the last "use me"
    Routing loop for net 9.0.0.0

Accept only 1st "use me" for any net
    No routing loop
    No recovery from lost links
    No way to improve routing



The BGP Protocol

Developed as a replacement for EGP

Traffic categories
- Local – Source or dest in current AS
- Transit – Source and dest in other AS's

AS categories
- Stub – A single connection to one AS (local traffic only)
- Multihomed – Connections to multiple other AS ... but carries only local traffic
- Transit – Multiple connections and carries transit traffic.

BGP vs. EGP
- EGP *is acceptable* for stub and multihomed AS's
- BGP is designed for Transit AS's (ARPANET.. NFSNET)
- (but its designers recommend its use in *all* networks)

BGP details
- A vector distance protocol but unlike RIP enumerates whole path
- Note that the path elements are AS's *not routers.*
- BGP advertises only paths that it uses
- BGP uses TCP as its transport.
- A routing update describes a path through N AS's followed by a list of networks numbers that can be reached via the path.

BGP topological model
- Two AS's are BGP connected if
  - They are physically connected
  - Contain BGP protocol connected routers
- Connected BGP routers must be on same network (or physically connected)
- BGP eliminates topological constraints of EGP
- Typically as many BGP speakers as AS connections are used in an AS

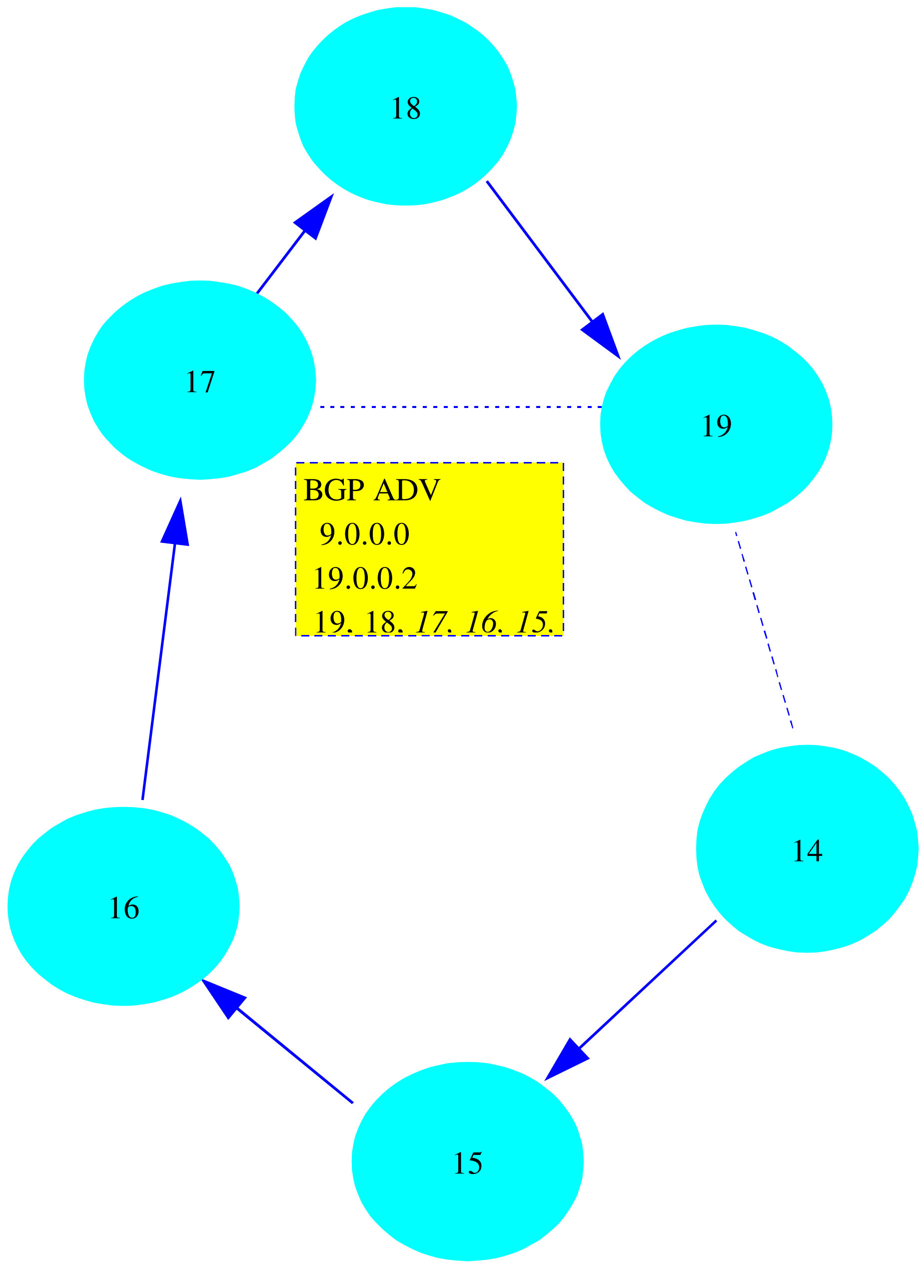
BGP policy managment supports concerns of type
- Economic
- Political
- Security

A multihomed AS can

- Refuse to carry transit traffic
- Carry transit traffic for a restricted subset of adjacent AS's
- Favor or disfavor use of other AS's for its own transit traffic.

Performance related preferences

- Minimize transit AS's



CIDR – Classless Interdomain routing

Objective

Reduce Core router table space explosion (caused primarily by class C's)

Mechanism

Allocate network addresses having a common gateway in blocks
e.g. European Class C's 0xc2000000: 0xc3ffffffff
In binary

```
1100 001x  xxxx xxxx  xxxx xxxx  xxxx xxxx
```

A network mask of

```
1111 1110  0000 0000  0000 0000  0000 0000
```

is associated with this address
Everyone *outside* Europe could then have a table entry

```
Destination          Net Mask     Via
c2 00 00 00   -   fe 00 00 00   ?? ?? ?? ??
```

One table entry of this form could then be used to address *every* class C network in Europe from the outside

Inside Europe, further decompositions would be possible

```
Destination          Net Mask     Via
c2 00 00 00   -   ff f0 00 00   ?? ?? ?? ??  (G.B)
c2 10 00 00   -   ff f0 00 00   ?? ?? ?? ??  (Fr.)
 :  :  :
c3 f0 00 00   -   ff f0 00 00   ?? ?? ?? ??  (It.)
```

Routing rule.. Longest match wins

Claimed advantage... Size of backbone routing tables could be reduced from 10,000 to 200 entries (*if* all existing hosts were renumbered).