# Internet Engineering

241-461

Robert Elz

kre@munnari.OZ.AU
kre@fivedots.coe.psu.ac.th

http://fivedots.coe.psu.ac.th/~kre

## Contents



## Routing Algorithms

- Routing Algorithm
  - Graph Traversal Algorithm
- Routing Protocol
  - Algorithm
  - Plus details needed
    - To make it practical
    - (Where does the information come from?)
- Concentrate on
  - Distributed
  - Dynamic
    - Protocols
- This is the interesting selection
  - Centralised
    - Not different (Harder information collection)
  - Static
    - Either the same (Or very boring)

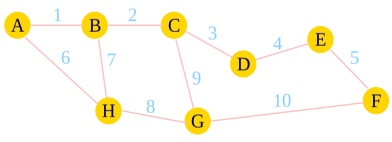
# Algorithms

- Two classes of algorithm
  - Bellman-Ford
  - Dijkstra
- Examine both
  - Start with Bellman-Ford
- Bellman-Ford Algorithms
  - Each node tells neighbours
    - All destinations it knows
    - And cost to reach each
  - And tells again
    - Each time information changes
  - That's It!

# Contents

- What is Routing?
- Types of Routing
- What has to be done?
- The Routing Problem
- Routing Algorithms
  - Bellman-Ford (Distance Vector)
  - Dijkstra (Link State)
- Hierarchical Routing
- Exterior Routing

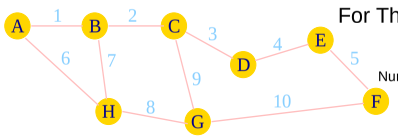
# B-F Example




For Our Example:

- Letters indicate node names
- Numbers indicate destinations (network numbers)
  - Not Costs
- All links have cost of 1
  - Then path cost == distance -- aim is to minimize distance
    - Number of hops -- measure network distance in hops, not km
  - Distance Vector Protocols
    - Note: possible to have different cost/link - depends upon configuration
- Each node sends vector of all known destinations to all neighbours
  - Contains destination, and cost (distance) from sending node
  - Receiving node adds cost (1 here) to get to sender

# B-F Example (2)




For This

Numbers indicate which network (network

Cost is 1 for every link

Result is smallest number of

Exa

numbe

travers

hops

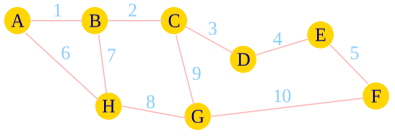
Initial State (just a subset)

A: Know 1 cost 0
A: Know 6 cost 0

B: Know 1 cost 0
B: Know 2 cost 0
B: Know 7 cost 0

H: Know 6 cost 0
H: Know 7 cost 0
H: Know 8 cost 0

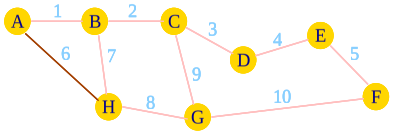
# B-F Example (3)




| | | |
|---|---|---|
| A: Know 1 cost 0 | direct | |
| A: Know 6 cost 0 | direct | |
| A: Know 2 cost 1 | via B | |
| A: Know 7 cost 1 | via B | (or H) |
| A: Know 8 cost 1 | via H | |
| B: Know 1 cost 0 | direct | |
| B: Know 2 cost 0 | direct | |
| B: Know 7 cost 0 | direct | |
| B: Know 6 cost 1 | via A | (or H) |
| B: Know 8 cost 1 | via H | |

- Continue the example

# Split Horizon



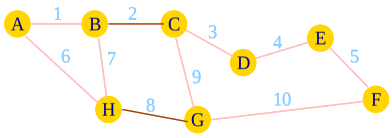

H: Know 6 cost 0 direct
G: Know 6 cost 1 via H
F: Know 6 cost 2 via G

- Link from H to net 6 breaks

H: 6 ???
F: Know 6 cost 2 via G
G: Know 6 cost 3 via F

- Forwarding Loop
  - Don't advertise route to the node
  - to which you will send traffic
    - Split Horizon
    - Don't send same information everywhere

# Counting to Infinity




B: Know 3 cost 1
A: Know 3 cost 2
H: Know 3 cost 2

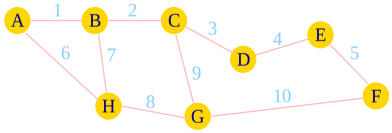
- Links 2 & 8 break
  - H: Know 3 cost 3
  - B: Know 3 cost 4
  - A: Know 3 cost 5
  - H: Know 3 cost 6
  - B: Know 3 cost 7

# RIP

- Routing Information Protocol
- Sends distance vector to neighbours
  - every 30 seconds
  - 180 seconds + no update -> gone
- 24 (25) routes/packet (512 byte limit)
- Infinity == 16
  - So cost/link almost always 1 (not required)
- Usually implements split horizon
- Usually implements triggered updates
  - Send an update whenever a route change occurs
- Usually implements hold-downs
  - Don't overreact to routing changes

# Contents

# Dijkstra's Algorithm

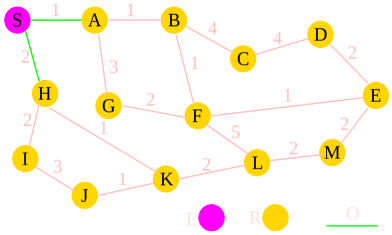



- Each node tells every other node its links
  - A: I have connections to 1, 6
  - B: I have connections to 1, 2, 7
  - C: I have connections to 2, 3, 9
  - H: I have connections to 6, 7, 8
    - This is Link State
      - Hence: Link State Algorithms
- All nodes receive all advertisements
  - Each can build the graph
  - Each knows topology of the net

# Shortest Path First

- Algorithm from Dijkstra
- Allows nodes to find path to any destination through a graph
  - Create 2 sets & a list
    - Set of all known reachable nodes (E)
    - Set of all unknown destinations (R)
    - ordered list of all paths found (O)
  - Order list of paths by path cost (length)
    - Shortest Path First
  - Set E to contain the source node (S)
  - Add direct paths from S to O

# SPF (1)

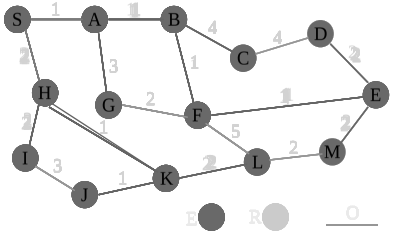



- Here
  - Numbers on arcs are costs
  - Names on nodes are destinations

# SPF (2)

- Take shortest path from O
- If destination in E, drop, repeat
- Add destination to E
- add to O all destinations from that path
  - keep list ordered
- repeat
- When done, anything in unknown set is unreachable
- Every other path is shortest to destination

# SPF Example




- At end O contains

S-A (1) S-A-B (2) S-H (2)
S-H-K (3) S-A-B-F (3) S-A-G (4)
S-H-I (4) S-H-K-J (4) S-A-B-F-E (4)
S-H-K-L (5) S-A-B-C (6) S-A-B-F-E-D (6)
S-A-B-F-E-M (6)

- Paths from S
  - To all destinations
  - Paths from other sources different

Building Forwarding Table

Last node (orange label) in each path in O is a destination

Node after S in each path is next hop to that destination

To A: Send to A
To B: Send to A
To K: Send to H
To L: Send to H

# Link State Protocols

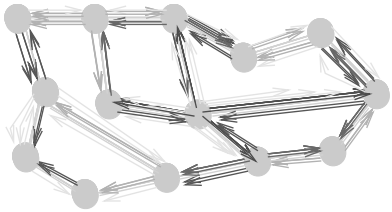


- Finding paths is done using SPF algorithm
- But
  - requires all nodes get
  - the complete link state database
- How?

# Distributing the Link Info

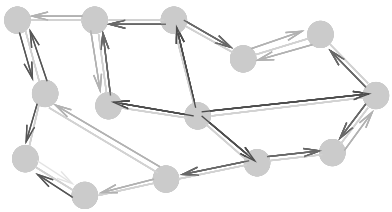
- Each node must send its link info to every other node
- Cannot use IP routing
  - the purpose of this is to build the routing table
- Need to use IP though
  - the only protocol guaranteed to work on the net
- Flooding
  - Simply send the info to every link
  - Nodes that receive info
    - forward it to every link
    - except the one they received it from

# Flooding...



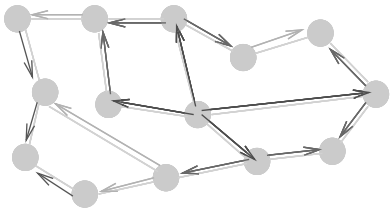
- Need loop control
  - Easy here
    - nodes have to remember info anyway
    - Don’t forward any info that is a duplicate

# Better Flooding



- Can do even better
  - delay sending for short while
  - don’t send to any node
    - from which a copy of the data was received

# Flooding as it is done



- This shows the link state distribution
  - from one node.
- Same thing happens from every other node
- Important that all of this arrives everywhere
  - Need ACK or retransmit

# Contents



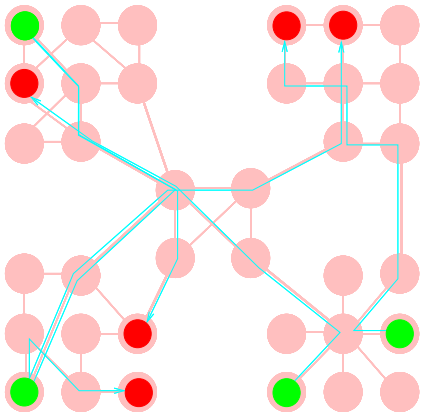
# The problem...



- Calculate all paths.
  - Not easy
  - Very many paths

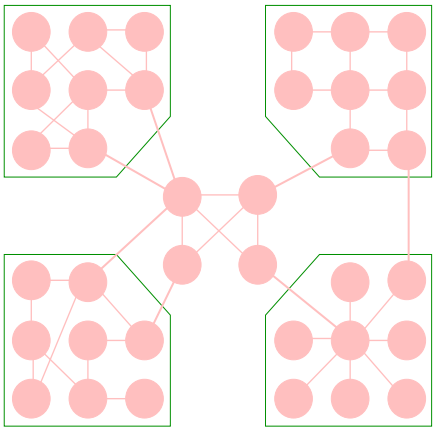
# Path Examples

- Some of the paths



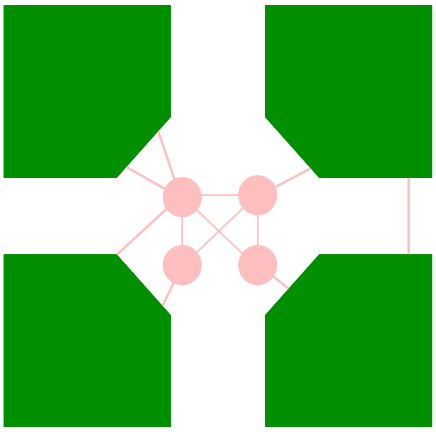
  - Can we simplify?

# Hierarchical Routing



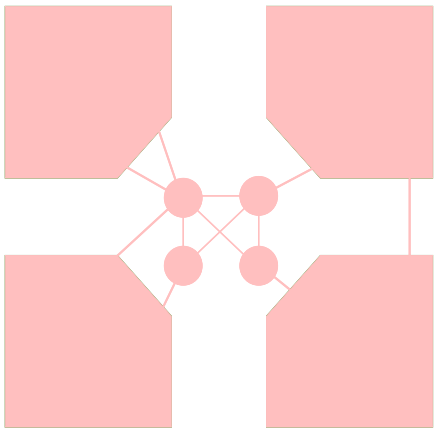
- Make groups of nets
  - Boundaries somewhere sensible

# Hierarchical Routing (2)



- And we can then hide the contents

# Hierarchical Routing (3)



- And re-colour to look like a net

# Hierarchical Routing (4)

- After simplification of the diagram...



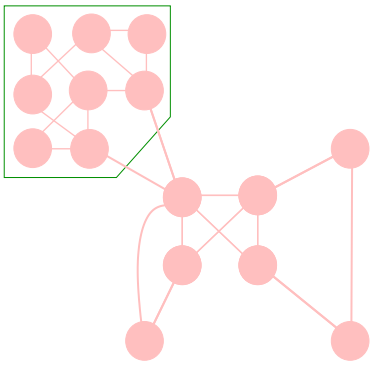
- Note similarity
  - to first routing simplification
- Hosts & nets
  - simplified to just net
- Here group of nets
  - simplified to just group/net

# Hierarchical Routing Issues

- Similar to net/host simplification
  - But not the same
- There we used link layer
  - From one host to another host
    - Link layer's problem
  - Routing need not care
- Here that cannot work:
  - There is no single net
  - Instead, many nets
    - We just hide them
  - Routing is still required
- How ?

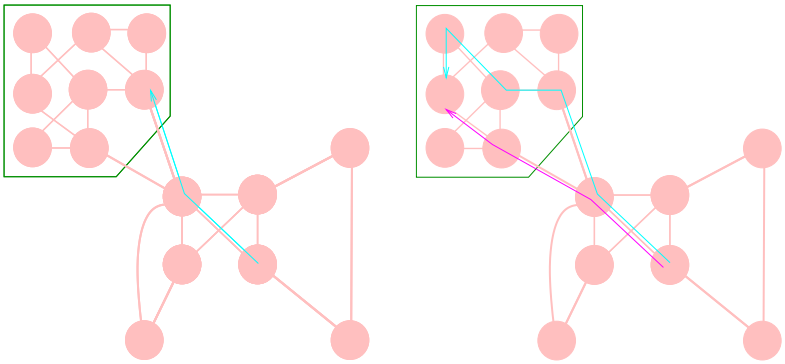
# Hierarchical Routing Issues (2)

- Nodes inside hidden net
  - Must know internal details
- Nodes outside
  - Leave it all hidden



# Hierarchical Routing Issues (3)

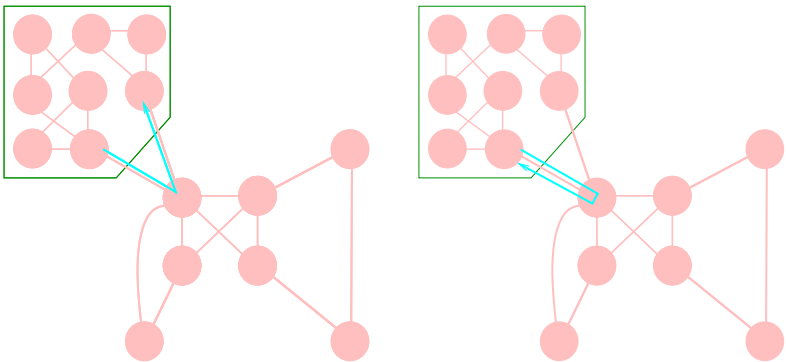
- We lose optimal paths



- Two paths into hidden net
  - Outside has no knowledge of inside
  - Which path to take?
  - Maybe the right one
    - but perhaps not always

# Hierarchical Routing Issues (4)

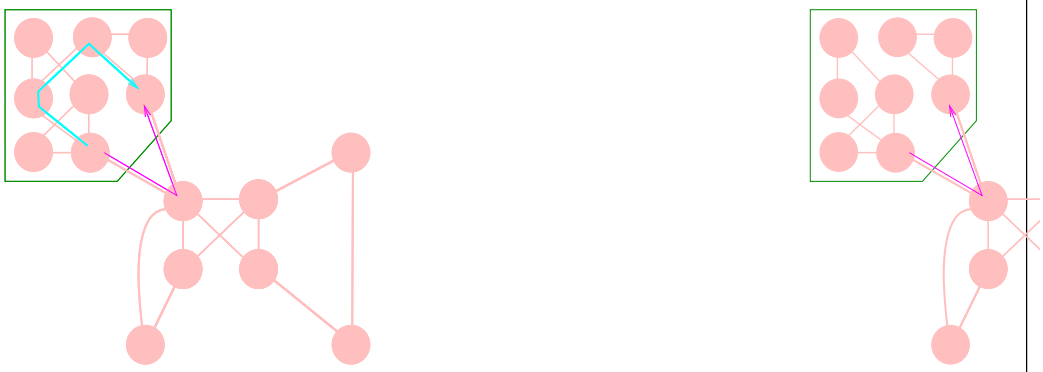
- Internal paths require care



- We cannot allow this path
  - Why ?
- Consider this second possibility
  - What prevents it ?

# Hierarchical Routing Issues (5)

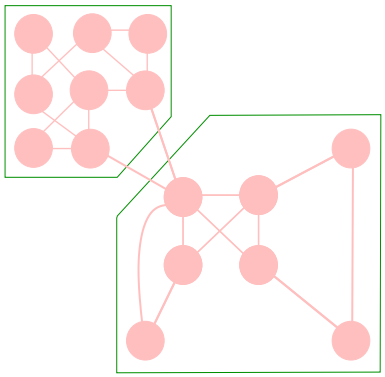
- Must require paths
  - Between nodes in the region
  - Remain entirely inside the region



- This is required
  - Even when there is no internal path

# Default Route

- Inside a local region
  - Often few paths to the outside
  - Usually want no knowledge
    - of anything out there
  - Instead prefer just
    - this link goes everywhere else
  - Treat entire outside as a big region



Packets for anywhere
sent to the
default region

Anywhere includes nowhere
local region
does not know
what exists
and can be reached
in outside world

# Longest Match Routing

- Routing algorithms all use addresses
  - How do they represent regions or anywhere ?
- Addresses used are actually
  - prefix
  - length
- The leftmost length bits of prefix
  - are the network address
  - (the rest is the ignored host number)
- Region made by making length smaller
  - For example:
    - instead of 10.11.12.0/24
    - we take 10.11.0.0/16
  - This means
    - 10.11.0.0 10.11.1.0 10.11.2.0
    - 10.11.3.0 10.11.4.0 ....
    - 10.11.252.0 10.11.254.0 10.11.255.0

# Longest Match Routing

- Then if two entries exist
  - 10.11.3.0/24
  - 10.11.0.0/16
- and packet for 10.11.3.9 arrives
  - Use 10.11.3.0/24
    - Both entries match
    - This has the longer (bigger) prefix
      - Longest Match Routing (forwarding)
- if packet for 10.11.4.9 arrives
  - Use 10.11.0.0/16
    - only this entry matches address
- Route to anywhere
    - Default Route
  - Is route to 0.0.0.0/0
    - Prefix length 0
    - Always shortest match

# Contents



# Exterior Routing

- Routing requires consistent algorithm
  - All routers must produce consistent paths
- OK when there is one management
    - company network
    - campus network
- But what about the Internet
  - Who decides on routing protocol?
  - Who assigns costs to links?
    - Who is the manager?
- There is no-one !
  - How does routing work then?

# Exterior Routing Protocols

- Need routing method
  - Tolerant of different configurations
  - Able to express policies
- Exterior Routing
  - Contrast Interior routing
    - and Interior Routing Protocols
      - Interior Gateway Protocols IGP
    - Routing inside a local region
      - Under common management
      - At least co-operating managements
  - with Exterior Routing & Protocols
    - Exterior Gateway Protocols EGP
- Border Gateway Protocol
  - an EGP
  - Connects borders of regions
    - and the network core
- A Path Vector Protocol

# BGP & Path Vector

- Path Vector Protocol
  - Similar to Distance Vector
  - Except routers exchange full paths
    - A-B-C-D-E
      - From A to E go via B, C and D
  - Cost is length of path
    - Aim to minimise length
  - Because entire path is known
    - Loops can be avoided
    - Never use a path that returns to me
- Some policies can be implemented
  - Because paths known
    - Can refuse certain sites
      - No paths including Q allowed
    - Or can prefer particular paths
      - Treat as lower cost
      - Even if have longer path
    - Loops cannot occur
      - Path check prevents them all

# BGP Routing (one extra bit)

- Uses Autonomous Systems (AS) as path elements
  - Identify collections of networks
    - ISP
    - Large organisations
  - Named by integers (AS number)
    - assigned sequentially
  - Path lists AS's and minimizes number of AS's
    - Only inside AS are actual net paths known
- BGP uses TCP
  - reliable transport
  - configured endpints
  - list all nets connected to this AS
  - list all AS paths advertised
  - updates only when changes occur
    - if TCP connection breaks, peer AS unreachable
  - TCP connections configured (by people)
    - policy for what routes (and data) is exchanged