Routing

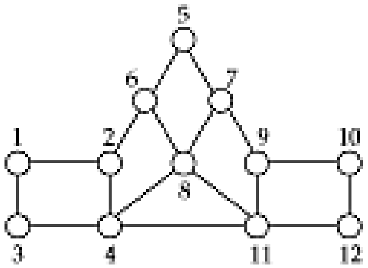
An Engineering Approach to Computer Networking

What is it?

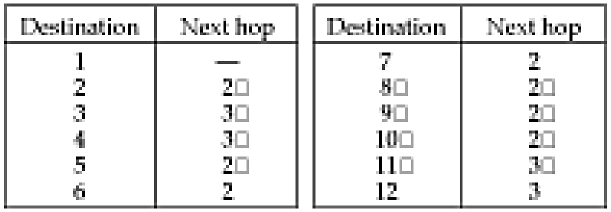
- Process of finding a path from a source to every destination in the network
- Suppose you want to connect to Antarctica from your desktop
  - what route should you take?
  - does a shorter route exist?
  - what if a link along the route goes down?
  - what if you’re on a mobile wireless link?
- Routing deals with these types of issues

Basics

- A *routing protocol* sets up a *routing table* in *routers* and *switch controllers*



ROUTING TABLE AT 1



Destination	Next hop	Destination	Next hop
1	—	7	2
2	2□	8□	2□
3	3□	9□	2□
4	3□	10□	2□
5	2□	11□	3□
6	2	12	3

- A node makes a *local* choice depending on *global* topology: this is the fundamental problem

Key problem

- How to make correct local decisions?
  - each router must know *something* about global state
- Global state
  - inherently large
  - dynamic
  - hard to collect
- *A routing protocol must intelligently summarize relevant information*

Requirements

- Minimize routing table space
  - fast to look up
  - less to exchange
- Minimize number and frequency of control messages
- Robustness: avoid
  - black holes
  - loops
  - oscillations
- Use optimal path

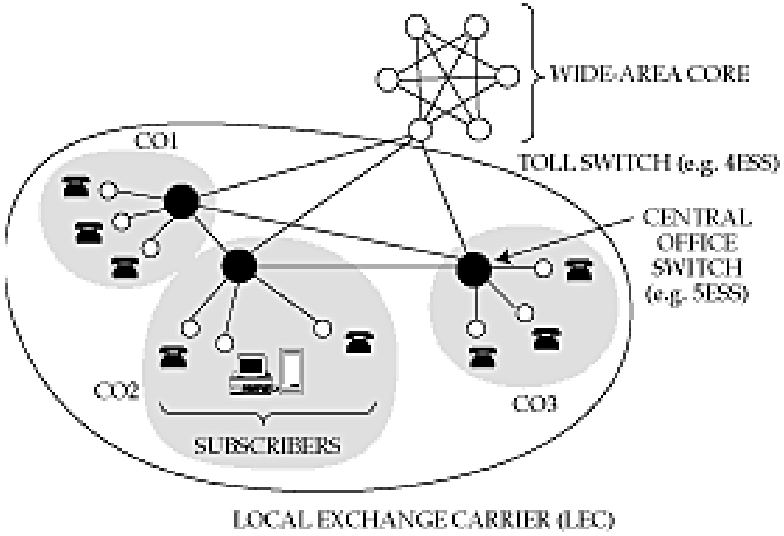
Choices

- Centralized vs. distributed routing
  - centralized is simpler, but prone to failure and congestion
- Source-based vs. hop-by-hop
  - how much is in packet header?
  - Intermediate: *loose source route*
- Stochastic vs. deterministic
  - stochastic spreads load, avoiding oscillations, but misorders
- Single vs. multiple path
  - primary and alternative paths (compare with stochastic)
- State-dependent vs. state-independent
  - do routes depend on current network state (e.g. delay)

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Telephone network topology


WIDE-AREA CORE
CO1
TOLL SWITCH (e.g. 4ESS)
CENTRAL OFFICE SWITCH (e.g. 5ESS)
CO2
CO3
SUBSCRIBERS
LOCAL EXCHANGE CARRIER (LEC)

- 3-level hierarchy, with a fully-connected core
- AT&T: 135 core switches with nearly 5 million circuits
- LECs may connect to multiple cores

Routing algorithm

- If endpoints are within same CO, directly connect
- If call is between COs in same LEC, use one-hop path between COs
- Otherwise send call to one of the cores
- Only major decision is at toll switch
  - one-hop or two-hop path to the destination toll switch
  - (why don’t we need longer paths?)
- Essence of problem
  - which two-hop path to use if one-hop path is full

Features of telephone network routing

- Stable load
  - can predict pairwise load throughout the day
  - can choose optimal routes in advance
- Extremely reliable switches
  - downtime is less than a few minutes per year
  - can assume that a chosen route is available
  - can’t do this in the Internet
- Single organization controls entire core
  - can collect global statistics and implement global changes
- Very highly connected network
- Connections require resources (but all need the same)

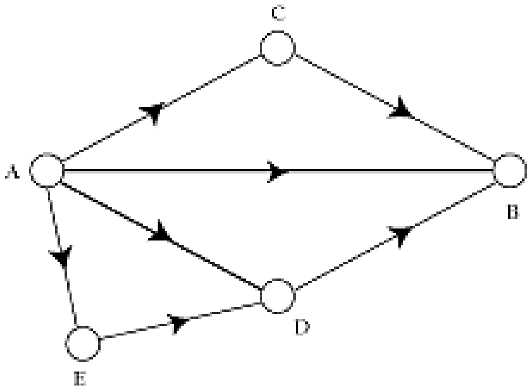
The cost of simplicity

- Simplicity of routing a historical necessity
- But requires
  - reliability in every component
  - logically fully-connected core
- Can we build an alternative that has same features as the telephone network, but is cheaper because it uses more sophisticated routing?
  - Yes: that is one of the motivations for ATM
  - But 80% of the cost is in the local loop
    - not affected by changes in core routing
  - Moreover, many of the software systems assume topology
    - too expensive to change them

Dynamic nonhierarchical routing (DNHR)

- Simplest core routing protocol
  - accept call if one-hop path is available, else drop
- DNHR
  - divides day into around 10-periods
  - in each period, each toll switch is assigned a primary one-hop path and a list of alternatives
  - can overflow to alternative if needed
  - drop only if all alternate paths are busy
    - *crankback*
- Problems
  - does not work well if actual traffic differs from prediction

Metastability



- Burst of activity can cause network to enter metastable state
  - high blocking probability even with a low load
- Removed by trunk reservation
  - prevents spilled traffic from taking over direct path

Trunk status map routing (TSMR)

- DNHR measures traffic once a week
- TSMR updates measurements once an hour or so
  - only if it changes “significantly”
- List of alternative paths is more up to date

Real-time network routing (RTNR)

- No centralized control
- Each toll switch maintains a list of lightly loaded links
- Intersection of source and destination lists gives set of lightly loaded paths
- Example
  - At A, list is C, D, E => links AC, AD, AE lightly loaded
  - At B, list is D, F, G => links BD, BF, BG lightly loaded
  - A asks B for its list
  - Intersection = D => AD and BD lightly loaded => ADB lightly loaded => it is a good alternative path
- Very effective in practice: only about a couple of calls blocked in core out of about 250 million calls attempted every day

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

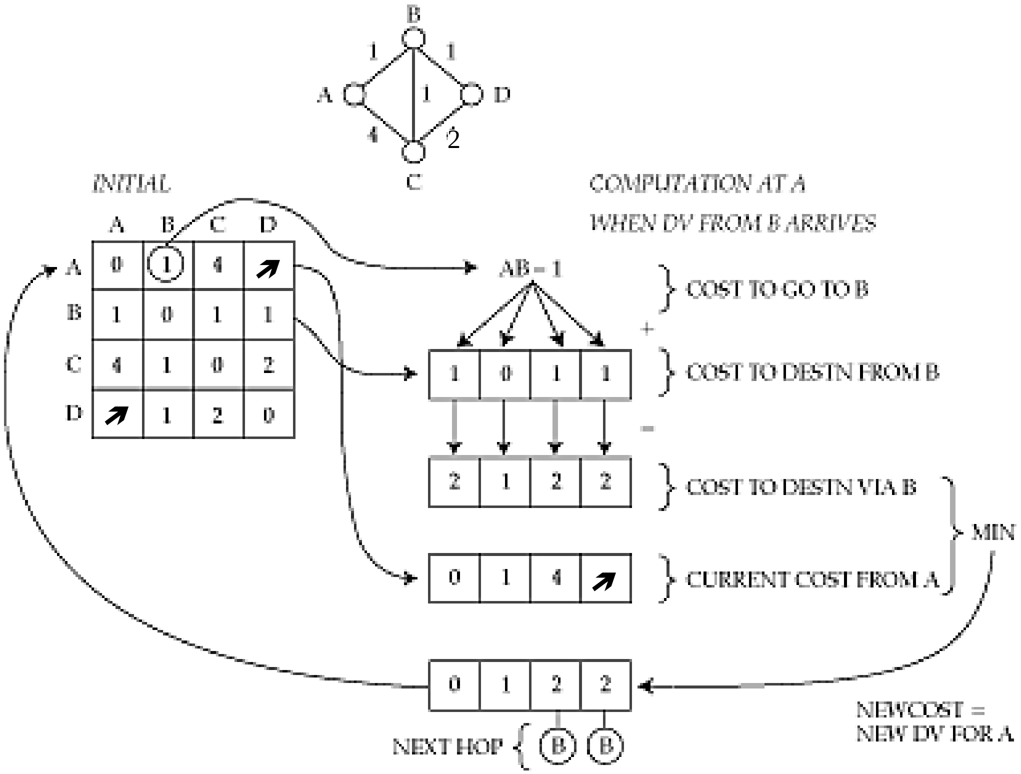
Distance vector routing

- Environment
  - links and routers unreliable
  - alternative paths scarce
  - traffic patterns can change rapidly
- Two key algorithms
  - distance vector
  - link-state
- Both assume router knows
  - address of each neighbor
  - cost of reaching each neighbor
- Both allow a router to determine global routing information by talking to its neighbors

Basic idea

- Node tells its neighbors its best idea of distance to *every* other node in the network
- Node receives these *distance vectors* from its neighbors
- Updates its notion of best path to each destination, and the next hop for this destination
- Features
  - distributed
  - adapts to traffic changes and link failures
  - suitable for networks with multiple administrative entities

Example



INITIAL

	A	B	C	D
A	0	1	4	↗
B	1	0	1	1
C	4	1	0	2
D	↗	1	2	0

COMPUTATION AT A WHEN DV FROM B ARRIVES

AB = 1 } COST TO GO TO B

+

1	0	1	1

} COST TO DESTN FROM B

=

2	1	2	2

} COST TO DESTN VIA B

0	1	4	↗

} CURRENT COST FROM A

MIN

0	1	2	2

NEXT HOP { B B

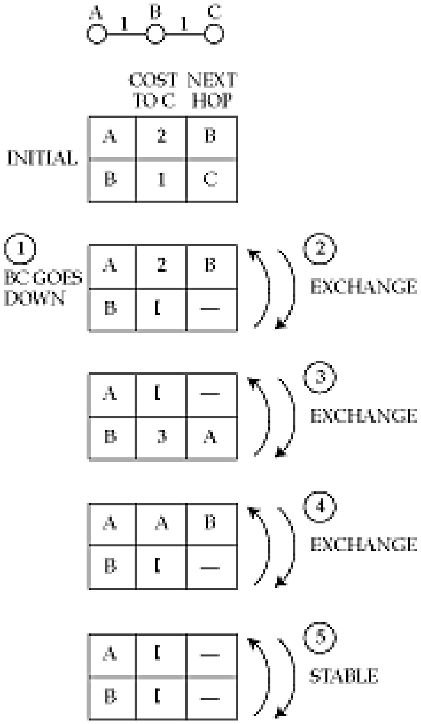
NEWCOST =
NEW DV FOR A

Why does it work

- Each node knows its true cost to its neighbors
- This information is spread to its neighbors the first time it sends out its distance vector
- Each subsequent dissemination spreads the truth one hop
- Eventually, it is incorporated into routing table everywhere in the network
- Proof: Bellman and Ford, 1957

Problems with distance vector

- Count to infinity



INITIAL

	COST TO C	NEXT HOP
A	2	B
B	1	C

① BC GOES DOWN

A	2	B
B	[	—

② EXCHANGE

A	[	—
B	3	A

③ EXCHANGE

A	A	B
B	[	—

④ EXCHANGE

A	[	—
B	[	—

⑤ STABLE

Dealing with the problem

- Path vector
  - DV carries path to reach each destination
- Split horizon
  - never tell neighbor cost to X if neighbor is next hop to X
  - doesn’t work for 3-way count to infinity (see exercise)
- Triggered updates
  - exchange routes on change, instead of on timer
  - faster count up to infinity
- More complicated
  - source tracing
  - DUAL

Outline

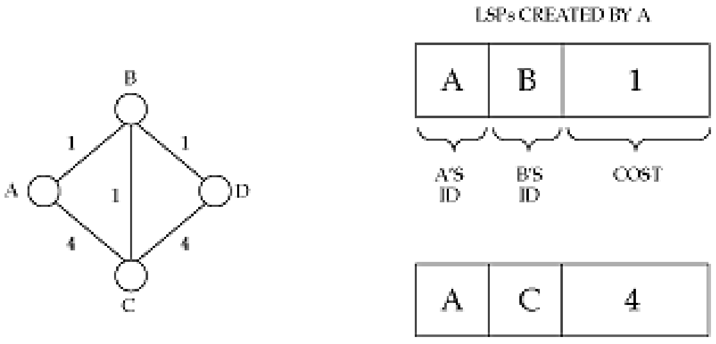
- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Link state routing

- In distance vector, router knows only *cost* to each destination
  - hides information, causing problems
- In link state, router knows entire network topology, and computes shortest path by itself
  - independent computation of routes
  - potentially less robust
- Key elements
  - topology dissemination
  - computing shortest routes

Link state: topology dissemination

- A router describes its neighbors with a *link state packet (LSP)*



LSPs CREATED BY A

A	B	1
A	C	4

- Use *controlled flooding* to distribute this everywhere
  - store an LSP in an *LSP database*
  - if new, forward to every interface other than incoming one
  - a network with E edges will copy at most 2E times

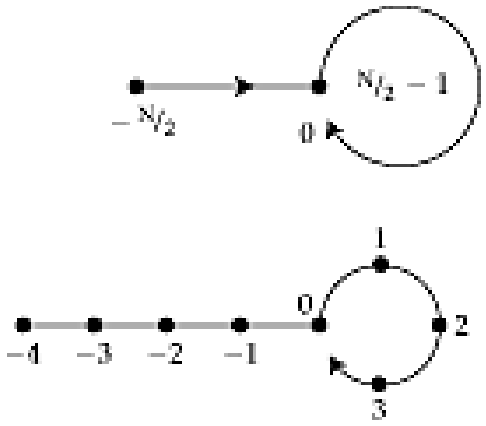
Sequence numbers

- How do we know an LSP is new?
- Use a sequence number in LSP header
- Greater sequence number is newer
- What if sequence number wraps around?
  - smaller sequence number is now newer!
  - (hint: use a large sequence space)
- On boot up, what should be the initial sequence number?
  - have to somehow purge old LSPs
  - two solutions
    - aging
    - lollipop sequence space

Aging

- Creator of LSP puts timeout value in the header
- Router removes LSP when it times out
    - also floods this information to the rest of the network (why?)
- So, on booting, router just has to wait for its old LSPs to be purged
- But what age to choose?
    - if too small
        - purged before fully flooded (why?)
        - needs frequent updates
    - if too large
        - router waits idle for a long time on rebooting

A better solution



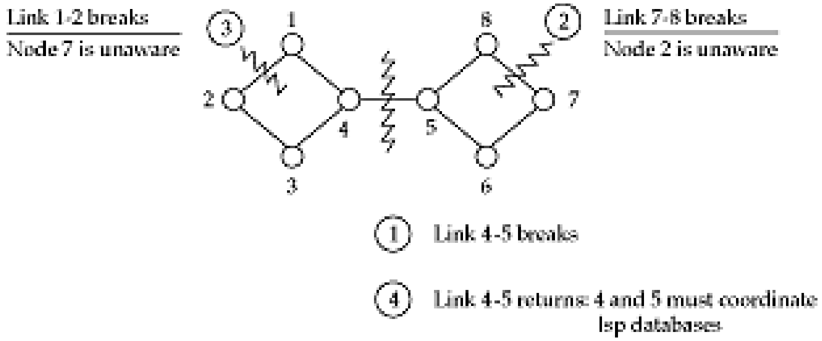
- Need a *unique* start sequence number
- a is older than b if:
  - a < 0 and a < b
  - a > o, a < b, and b-a < N/4
  - a > 0, b > 0, a > b, and a-b > N/4

More on lollipops

- If a router gets an older LSP, it tells the sender about the newer LSP
- So, newly booted router quickly finds out its most recent sequence number
- It jumps to one more than that
- -N/2 is a *trigger* to evoke a response from community memory

Recovering from a partition

- On partition, LSP databases can get out of synch



- Databases described by database descriptor records
- Routers on each side of a newly restored link talk to each other to update databases (determine missing and out-of-date LSPs)

Router failure

- How to detect?
  - HELLO protocol
- HELLO packet may be corrupted
  - so age anyway
  - on a timeout, flood the information

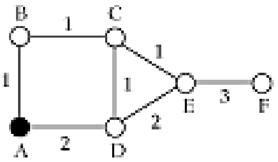
Securing LSP databases

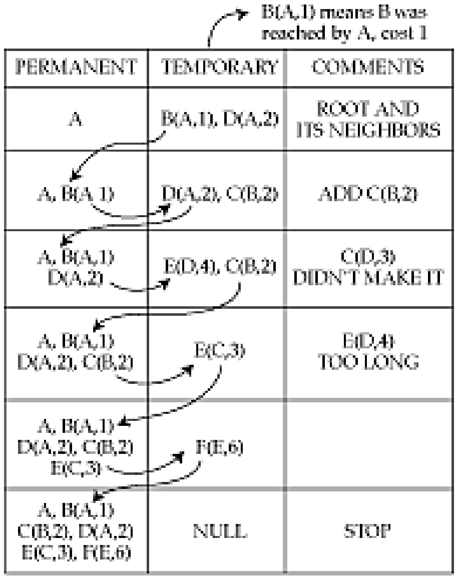
- LSP databases *must* be consistent to avoid routing loops
- Malicious agent may inject spurious LSPs
- Routers must actively protect their databases
  - checksum LSPs
  - ack LSP exchanges
  - passwords

Computing shortest paths

- Basic idea
  - maintain a set of nodes P to whom we know shortest path
  - consider every node one hop away from nodes in P = T
  - find every way in which to reach a given node in T, and choose shortest one
  - then add this node to P

Example

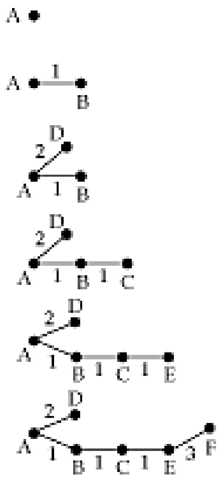



B(A,1) means B was reached by A, cost 1

PERMANENT	TEMPORARY	COMMENTS
A	B(A,1), D(A,2)	ROOT AND ITS NEIGHBORS
A, B(A 1)	D(A,2), C(B,2)	ADD C(B,2)
A, B(A,1) D(A,2)	E(D,4), C(B,2)	C(D,3) DIDN'T MAKE IT
A, B(A,1) D(A,2), C(B,2)	E(C,3)	E(D,4) TOO LONG

A, B(A,1)
D(A,2), C(B,2)
E(C,3)
	F(E,6)	

A, B(A,1)
C(B,2), D(A,2)
E(C,3), F(E,6)
	NULL	STOP



Link state vs. distance vector

- Criteria
  - stability
  - multiple routing metrics
  - convergence time after a change
  - communication overhead
  - memory overhead
- Both are evenly matched
- Both widely used

Outline

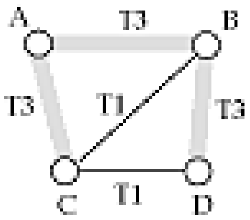
- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Choosing link costs

- Shortest path uses link costs
- Can use either static of dynamic costs
- In both cases: cost determine amount of traffic on the link
  - lower the cost, more the expected traffic
  - if dynamic cost depends on load, can have oscillations (why?)

Static metrics

- Simplest: set all link costs to 1 => min hop routing
  - but 28.8 modem link is not the same as a T3!
- Give links weight proportional to capacity



WEIGHTS

T3 = 1
T1 = 10

Dynamic metrics

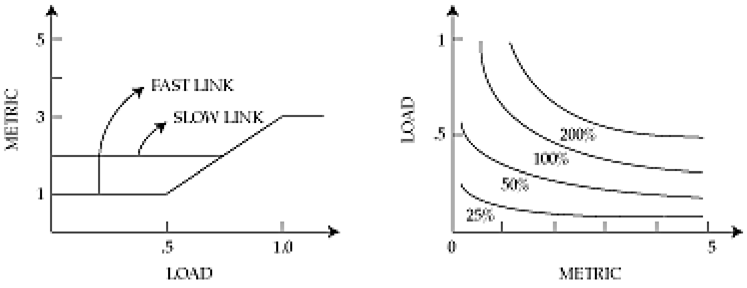
- A first cut (ARPAnet original)
- Cost proportional to length of router queue
  - independent of link capacity
- Many problems when network is loaded
  - queue length averaged over a small time => transient spikes caused major rerouting
  - wide dynamic range => network completely ignored paths with high costs
  - queue length assumed to predict future loads => opposite is true (why?)
  - no restriction on successively reported costs => oscillations
  - all tables computed simultaneously => low cost link flooded

Modified metrics

- queue length averaged over a small time
- wide dynamic range queue
- queue length assumed to predict future loads
- no restriction on successively reported costs
- all tables computed simultaneously

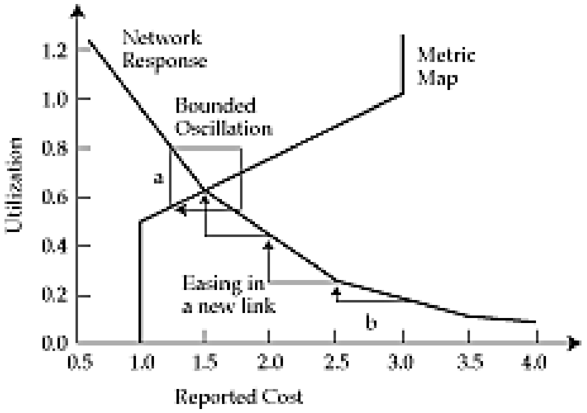
- queue length averaged over a longer time
- dynamic range restricted
- cost also depends on intrinsic link capacity
- restriction on successively reported costs
- attempt to stagger table computation

Routing dynamics



(a) METRIC MAP

(b) NETWORK RESPONSE MAP



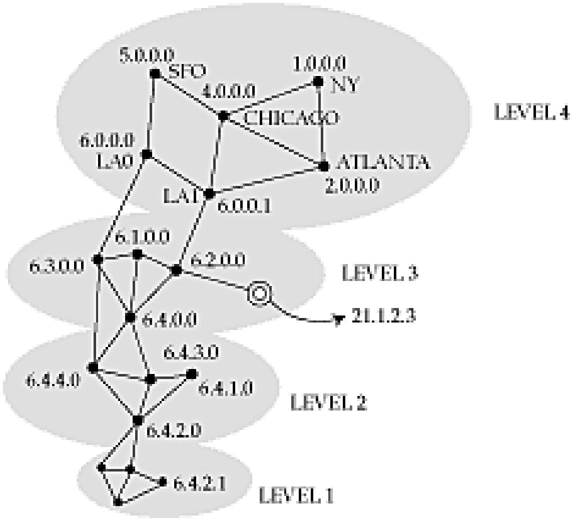
Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Hierarchical routing

- Large networks need large routing tables
  - more computation to find shortest paths
  - more bandwidth wasted on exchanging DVs and LSPs
- Solution:
  - hierarchical routing
- Key idea
  - divide network into a set of domains
  - gateways connect domains
  - computers within domain unaware of outside computers
  - gateways know only about other gateways

Example



- Features
  - only a few routers in each level
  - not a strict hierarchy
  - gateways participate in multiple routing protocols
  - non-aggregable routers increase core table space

Hierarchy in the Internet

- Three-level hierarchy in addresses
  - network number
  - subnet number
  - host number
- Core advertises routes only to networks, not to subnets
  - e.g. 135.104.*, 192.20.225.*
- Even so, about 80,000 networks in core routers (1996)
- Gateways talk to backbone to find best next-hop to every other network in the Internet

External and summary records

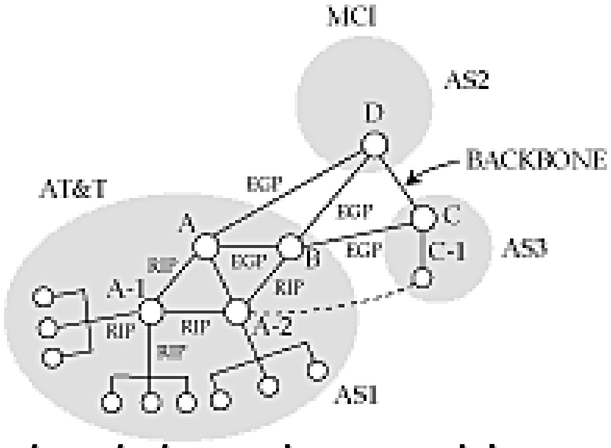
- If a domain has multiple gateways
  - *external* records tell hosts in a domain which one to pick to reach a host in an external domain
    - e.g allows 6.4.0.0 to discover shortest path to 5.* is through 6.0.0.0
  - *summary* records tell backbone which gateway to use to reach an internal node
    - e.g. allows 5.0.0.0 to discover shortest path to 6.4.0.0 is through 6.0.0.0
- External and summary records contain distance from gateway to external or internal node
  - unifies distance vector and link state algorithms

Interior and exterior protocols

- Internet has three levels of routing
  - highest is at *backbone* level, connecting *autonomous systems (AS)*
  - next level is within AS
  - lowest is within a LAN
- Protocol between AS gateways: exterior gateway protocol
- Protocol within AS: interior gateway protocol

Exterior gateway protocol

- Between untrusted routers
  - mutually suspicious
- Must tell a *border gateway* who can be trusted and what paths are allowed



- *Transit* over *backdoors* is a problem

Interior protocols

- Much easier to implement
- Typically partition an AS into *areas*
- Exterior and summary records used between areas

Issues in interconnection

- May use different schemes (DV vs. LS)
- Cost metrics may differ
- Need to:
  - convert from one scheme to another (how?)
  - use the lowest common denominator for costs
  - manually intervene if necessary

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Common routing protocols

- Interior
  - RIP
  - OSPF
- Exterior
  - EGP
  - BGP
- ATM
  - PNNI

RIP

- Distance vector
- Cost metric is hop count
- Infinity = 16
- Exchange distance vectors every 30 s
- Split horizon
- Useful for small subnets
    - easy to install

OSPF

- Link-state
- Uses areas to route packets hierarchically within AS
- Complex
  - LSP databases to be protected
- Uses *designated routers* to reduce number of endpoints

EGP

- Original exterior gateway protocol
- Distance-vector
- Costs are either 128 (reachable) or 255 (unreachable) => reachability protocol => backbone must be loop free (why?)
- Allows administrators to pick neighbors to peer with
- Allows backdoors (by setting backdoor cost < 128)

BGP

- Path-vector
  - distance vector annotated with entire path
  - also with policy attributes
  - guaranteed loop-free
- Can use non-tree backbone topologies
- Uses TCP to disseminate DVs
  - reliable
  - but subject to TCP flow control
- Policies are complex to set up

PNNI

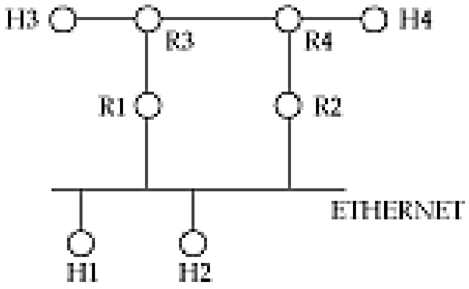
- Link-state
- Many levels of hierarchy
- Switch controllers at each level form a peer group
- Group has a group leader
- Leaders are members of the next higher level group
- Leaders summarize information about group to tell higher level peers
- All records received by leader are flooded to lower level
- LSPs can be annotated with per-link QoS metrics
- Switch controller uses this to compute source routes for call-setup packets

Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Routing within a broadcast LAN

- What happens at an endpoint?
- On a point-to-point link, no problem
- On a broadcast LAN
  - is packet meant for destination within the LAN?
  - if so, what is the datalink address ?
  - if not, which router on the LAN to pick?
  - what is the router’s datalink address?



Internet solution

- All hosts on the LAN have the same subnet address
- So, easy to determine if destination is on the same LAN
- Destination’s datalink address determined using ARP
  - broadcast a request
  - owner of IP address replies
- To discover routers
  - routers periodically sends router advertisements
    - with preference level and time to live
  - pick most preferred router
  - delete overage records
  - can also force routers to reply with *solicitation message*

Redirection

- How to pick the best router?
- Send message to arbitrary router
- If that router’s next hop is another router on the same LAN, host gets a *redirect* message
- It uses this for subsequent messages

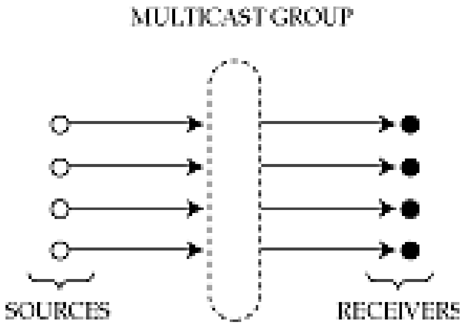
Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Multicast routing

- Unicast: single source sends to a single destination
- Multicast: hosts are part of a *multicast group*
  - packet sent by *any* member of a group are received by *all*
- Useful for
  - multiparty videoconference
  - distance learning
  - resource location

Multicast group

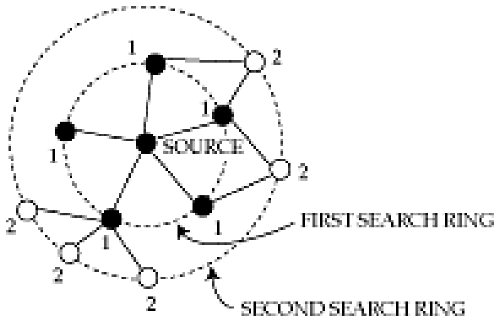


- Associates a set of senders and receivers with each other
  - but independent of them
  - created either when a sender starts sending from a group
  - or a receiver expresses interest in receiving
  - even if no one else is there!
- Sender does not need to know receivers' identities
  - *rendezvous point*

Addressing

- Multicast group in the Internet has its own Class D address
  - looks like a host address, but isn’t
- Senders send to the address
- Receivers anywhere in the world request packets from that address
- “Magic” is in associating the two: *dynamic directory service*
- Four problems
  - which groups are currently active
  - how to express interest in joining a group
  - discovering the set of receivers in a group
  - delivering data to members of a group

Expanding ring search

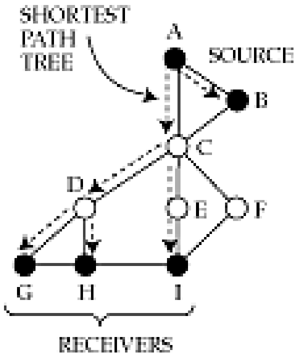


- A way to use multicast groups for resource discovery
- Routers decrement TTL when forwarding
- Sender sets TTL and multicasts
  - reaches all receivers <= TTL hops away
- Discovers local resources first
- Since heavily loaded servers can keep quiet, automatically distributes load

Multicast flavors

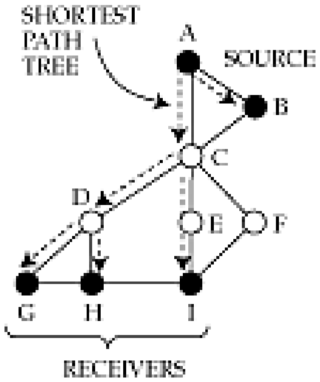
- Unicast: point to point
- Multicast:
  - point to multipoint
  - multipoint to multipoint
- Can simulate point to multipoint by a set of point to point unicasts
- Can simulate multipoint to multipoint by a set of point to multipoint multicasts
- The difference is efficiency

Example



- Suppose A wants to talk to B, G, H, I, B to A, G, H, I
- With unicast, 4 messages sent from each source
  - links AC, BC carry a packet in triplicate
- With point to multipoint multicast, 1 message sent from each source
  - but requires establishment of two separate multicast groups
- With multipoint to multipoint multicast, 1 message sent from each source,
  - single multicast group

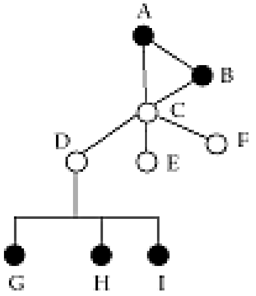
Shortest path tree



- Ideally, want to send exactly one multicast packet per link
  - forms a *multicast tree* rooted at sender
- Optimal multicast tree provides *shortest* path from sender to every receiver
  - *shortest-path* tree rooted at sender

Issues in wide-area multicast

- Difficult because
  - sources may join and leave dynamically
    - need to dynamically update shortest-path tree
  - leaves of tree are often members of broadcast LAN
    - would like to exploit LAN broadcast capability



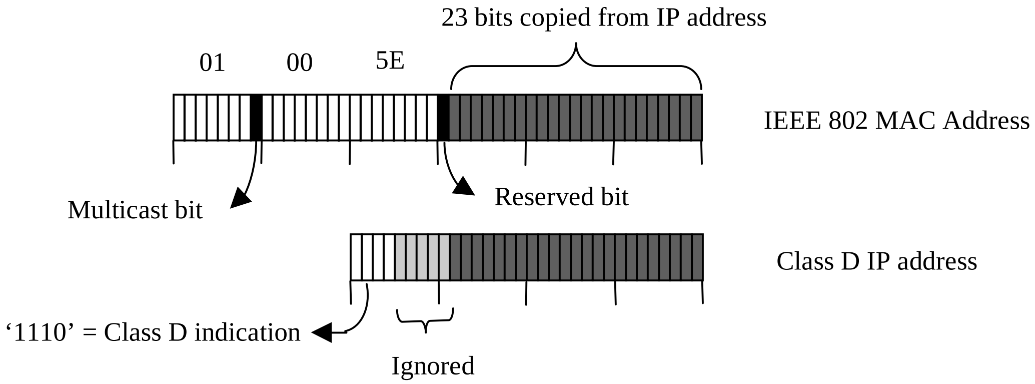
  - would like a receiver to join or leave without explicitly notifying sender
    - otherwise it will not scale

Multicast in a broadcast LAN

- Wide area multicast can exploit a LAN's broadcast capability
- E.g. Ethernet will multicast all packets with multicast bit set on destination address

- Two problems:
  - what multicast MAC address corresponds to a given Class D IP address?
  - does the LAN have contain any members for a given group (why do we need to know this?)

Class D to MAC translation



- Multiple Class D addresses map to the same MAC address
- Well-known translation algorithm => no need for a translation table

Internet Group Management Protocol

- Detects if a LAN has any members for a particular group
  - If no members, then we can *prune* the shortest path tree for that group by telling parent
- Router periodically broadcasts a *query* message
- Hosts reply with the list of groups they are interested in
- To suppress traffic
  - reply after random timeout
  - broadcast reply
  - if someone else has expressed interest in a group, drop out
- To receive multicast packets:
  - translate from class D to MAC and configure adapter

Wide area multicast

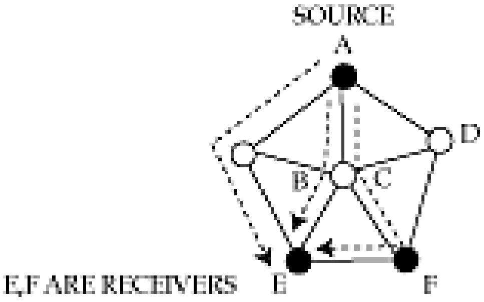
- Assume
  - each endpoint is a router
  - a router can use IGMP to discover all the members in its LAN that want to subscribe to each multicast group
- Goal
  - distribute packets coming from any sender directed to a given group to all routers on the path to a group member

Simplest solution

- Flood packets from a source to entire network
- If a router has not seen a packet before, forward it to all interfaces except the incoming one
- Pros
  - simple
  - always works!
- Cons
  - routers receive duplicate packets
  - detecting that a packet is a duplicate requires storage, which can be expensive for long multicast sessions

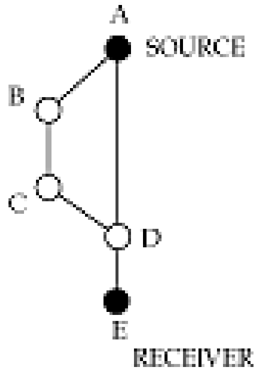
A clever solution

- *Reverse path forwarding*
- Rule
  - forward packet from S to all interfaces if and only if packet arrives on the interface that corresponds to the shortest path *to* S
  - no need to remember past packets
  - C need not forward packet received from D



Cleverer

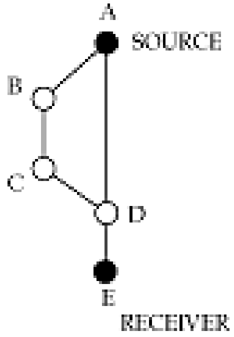
- Don’t send a packet downstream if you are not on the shortest path from the downstream router to the source
- C need not forward packet from A to E



- Potential confusion if downstream router has a choice of shortest paths to source (see figure on previous slide)

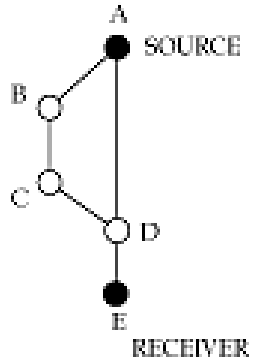
Pruning

- RPF does not completely eliminate unnecessary transmissions



- B and C get packets even though they do not need it
- Pruning => router tells parent in tree to stop forwarding
- Can be associated either with a multicast group or with a source *and* group
  - trades selectivity for router memory

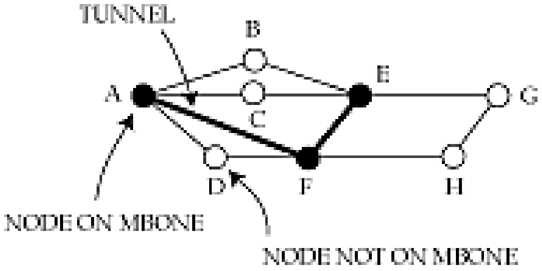
Rejoining



- What if host on C’s LAN wants to receive messages from A after a previous prune by C?
  - IGMP lets C know of host’s interest
  - C can send a *join(group, A)* message to B, which propagates it to A
  - or, periodically flood a message; C refrains from pruning

A problem

- Reverse path forwarding requires a router to know shortest path to a source
  - known from routing table
- Doesn’t work if some routers do not support multicast
  - *virtual links* between multicast-capable routers
  - shortest path to A from E is not C, but F

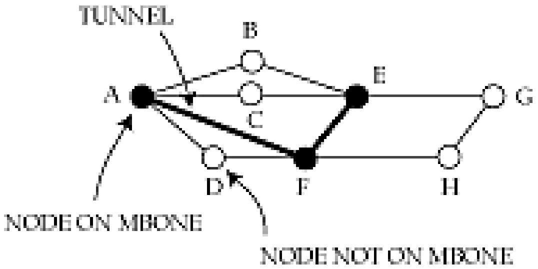


A problem (contd.)

- Two problems
  - how to build virtual links
  - how to construct routing table for a network with virtual links

Tunnels

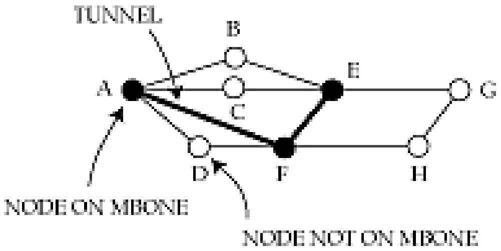
- Why do we need them?



- Consider packet sent from A to F via multicast-incapable D
- If packet’s destination is Class D, D drops it
- If destination is F’s address, F doesn’t know multicast address!
- So, put packet destination as F, but carry multicast address internally
- Encapsulate IP in IP => set protocol type to IP-in-IP

Multicast routing protocol

- Interface on “shortest path” to source depends on whether path is real or virtual



- Shortest path from E to A is not through C, but F
  - so packets from F will be flooded, but not from C
- Need to discover shortest paths only taking multicast-capable routers into account
  - DVMRP

DVMRP

- Distance-vector Multicast routing protocol
- Very similar to RIP
  - distance vector
  - hop count metric
- Used in conjunction with
  - flood-and-prune (to determine memberships)
    - prunes store per-source and per-group information
  - reverse-path forwarding (to decide where to forward a packet)
  - explicit join messages to reduce join latency (but no source info, so still need flooding)

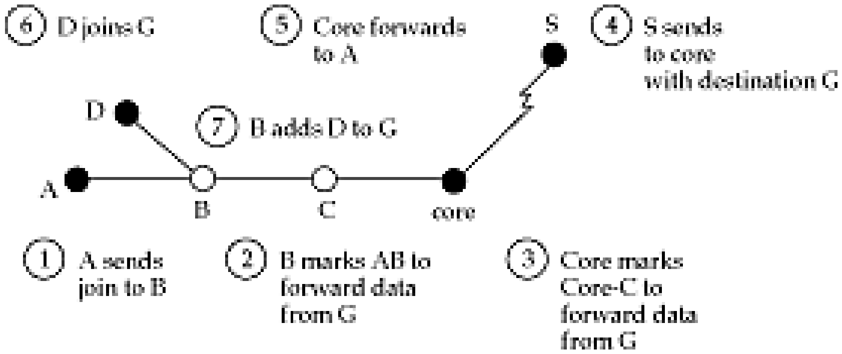
MOSPF

- Multicast extension to OSPF
- Routers flood group membership information with LSPs
- Each router independently computes shortest-path tree that only includes multicast-capable routers
  - no need to flood and prune
- Complex
  - interactions with external and summary records
  - need storage per group per link
  - need to compute shortest path tree per source and group

Core-based trees

- Problems with DVMRP-oriented approach
  - need to periodically flood and prune to determine group members
  - need to source per-source and per-group prune records at each router
- Key idea with core-based tree
  - coordinate multicast with a *core* router
  - host sends a join request to core router
  - routers along path mark incoming interface for forwarding

Example

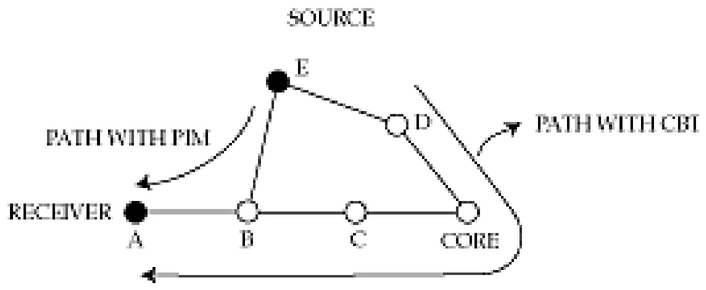

6 D joins G
5 Core forwards to A
S
4 S sends to core with destination G
D
7 B adds D to G
A
B
C
core
1 A sends join to B
2 B marks AB to forward data from G
3 Core marks Core-C to forward data from G

- Pros
  - routers not part of a group are not involved in pruning
  - explicit join/leave makes membership changes faster
  - router needs to store only one record per group
- Cons
  - all multicast traffic traverses core, which is a bottleneck
  - traffic travels on non-optimal paths

Protocol independent multicast (PIM)

- Tries to bring together best aspects of CBT and DVMRP
- Choose different strategies depending on whether multicast tree is *dense* or *sparse*
  - flood and prune good for dense groups
    - only need a few prunes
    - CBT needs explicit join per source/group
  - CBT good for sparse groups
- Dense mode PIM == DVMRP
- Sparse mode PIM is similar to CBT
  - but receivers can switch from CBT to a shortest-path tree

PIM (contd.)



- In CBT, E must send to core
- In PIM, B discovers shorter path to E (by looking at unicast routing table)
  - sends join message directly to E
  - sends prune message towards core
- Core no longer bottleneck
- Survives failure of core

More on core

- Renamed a *rendezvous point*
  - because it no longer carries all the traffic like a CBT core
- Rendezvous points periodically send “I am alive” messages downstream
- Leaf routers set timer on receipt
- If timer goes off, send a join request to alternative rendezvous point
- Problems
  - how to decide whether to use dense or sparse mode?
  - how to determine “best” rendezvous point?

Outline

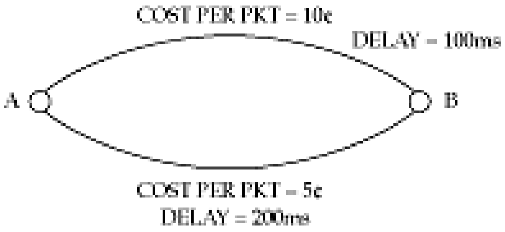
- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

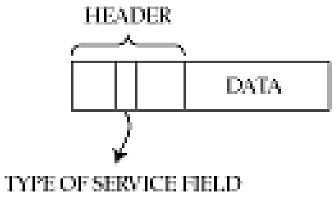
Routing vs. policy routing

- In standard routing, a packet is forwarded on the ‘best’ path to destination
  - choice depends on load and link status
- With policy routing, routes are chosen depending on *policy* directives regarding things like
  - source and destination address
  - transit domains
  - quality of service
  - time of day
  - charging and accounting
- The general problem is still open
  - fine balance between correctness and information hiding

Multiple metrics

- Simplest approach to policy routing
- Advertise multiple costs per link
- Routers construct multiple shortest path trees





Problems with multiple metrics

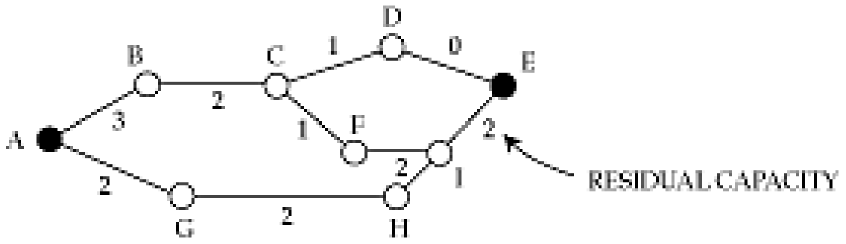
- All routers must use the same rule in computing paths
- Remote routers may misinterpret policy
  - source routing may solve this
  - but introduces other problems (what?)

Provider selection

- Another simple approach
- Assume that a single service provider provides almost all the path from source to destination
    - e.g. AT&T or MCI
- Then, choose policy simply by choosing provider
    - this could be dynamic (agents!)
- In Internet, can use a loose source route through service provider's access point
- Or, multiple addresses/names per host

Crankback

- Consider computing routes with QoS guarantees
- Router returns packet if no next hop with sufficient QoS can be found
- In ATM networks (PNNI) used for the call-setup packet
- In Internet, may need to be done for _every_ packet!
  - Will it work?



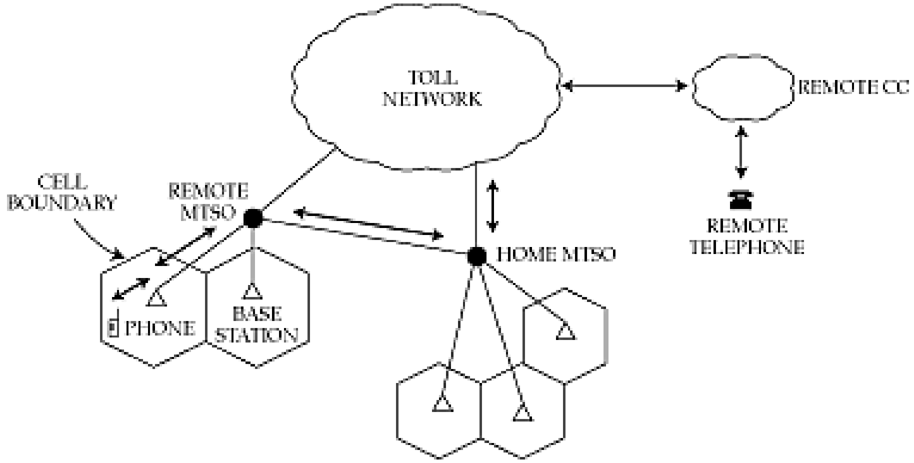
Outline

- Routing in telephone networks
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN
- Multicast routing
- Routing with policy constraints
- Routing for mobile hosts

Mobile routing

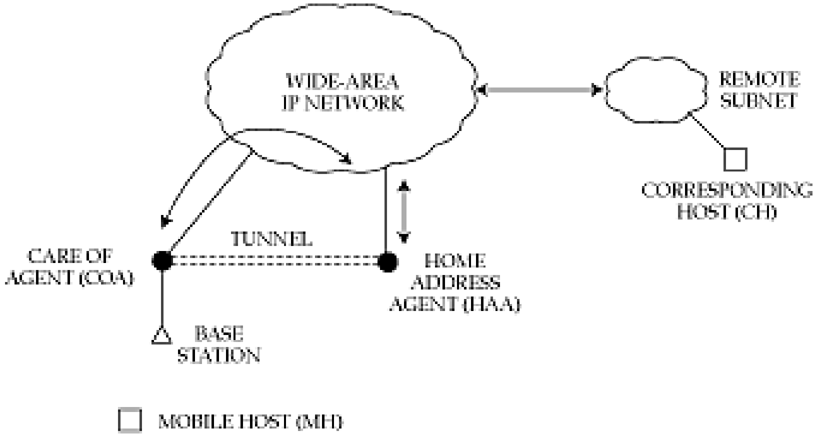
- How to find a mobile host?
- Two sub-problems
  - location (where is the host?)
  - routing (how to get packets to it?)
- We will study mobile routing in the Internet and in the telephone network

Mobile routing in the telephone network



- Each cell phone has a global ID that it tells remote MTSO when turned on (using slotted ALOHA up channel)
- Remote MTSO tells home MTSO
- *To* phone: call forwarded to remote MTSO to closest base
- *From* phone: call forwarded to home MTSO from closest base
- New MTSOs can be added as load increases

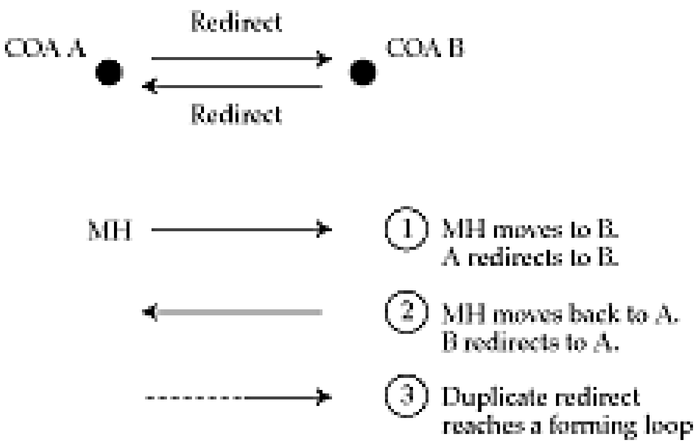
Mobile routing in the Internet



- Very similar to mobile telephony
  - but outgoing traffic does not go through home
  - and need to use tunnels to forward data
- Use *registration* packets instead of slotted ALOHA
  - passed on to home address agent
- Old care-of-agent forwards packets to new care-of-agent until home address agent learns of change

Problems

- Security
  - mobile and home address agent share a common secret
  - checked before forwarding packets to COA
- Loops


Redirect
COA A
COA B
Redirect
MH
1 MH moves to B.
A redirects to B.
2 MH moves back to A.
B redirects to A.
3 Duplicate redirect
reaches a forming loop