Adressage et Routage point à point dans l’Internet

Bloc 3, INF 586

Walid Dabbous
INRIA Sophia Antipolis

Nommage et adressage

Outline

- Naming and Addressing
  - Names and addresses
  - Hierarchical naming
  - Addressing
  - Addressing in the Internet
  - Name resolution
  - Finding datalink layer addresses

Names and addresses

- Names and addresses both uniquely identify a host (or an interface on the host)
- `%nslookup`
  - `Default Server:  euryale101.inria.fr`
  - `Address:  138.96.80.222`

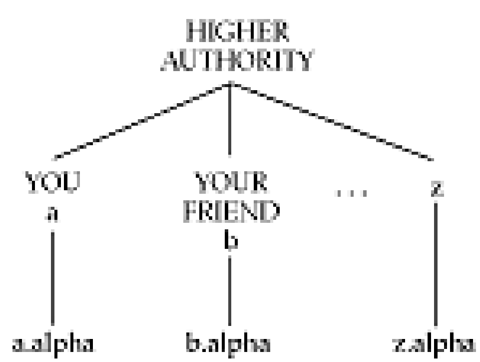
  - `> lix.polytechnique.fr`
  - `Name:    lix.polytechnique.fr`
  - `Address:  129.104.11.2`
- *Resolution*: the process of determining an address from a name

Why do we need both?

- Names are long and human understandable
  - wastes space to carry them in packet headers
  - hard to parse
- Addresses are shorter and machine understandable
  - if fixed size, easy to carry in headers and parse
- Indirection
  - multiple names may point to same address
  - can move a machine in same domain and just update the resolution table

Hierarchical naming

- Goal: give a globally unique name to each host
- Naïve approach: ask every other naming authorities before choosing a name
  - doesn't scale
  - not robust to network partitions
- Instead carve up *name space* (the set of all possible names) into mutually exclusive portions => hierarchy



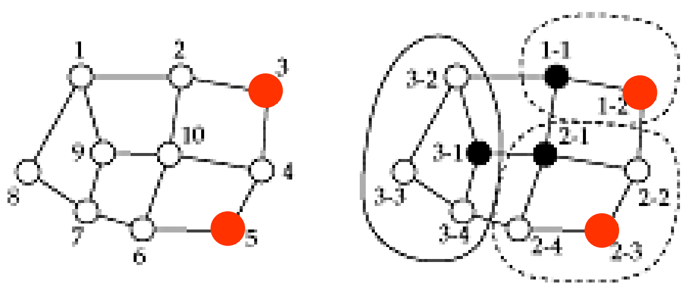
Top level domain

Hierarchy

- A wonderful thing!
  - simplifies distributed naming
  - guarantees uniqueness
  - scales arbitrarily
- Example: Internet names
  - use *Domain name system (DNS)*
  - global authority (Network Solutions Inc.) assigns top level domains to naming authorities (e.g. .edu, .net, .cz etc.)
  - naming authorities further carve up their space
  - all names in the same domain share a unique *suffix*

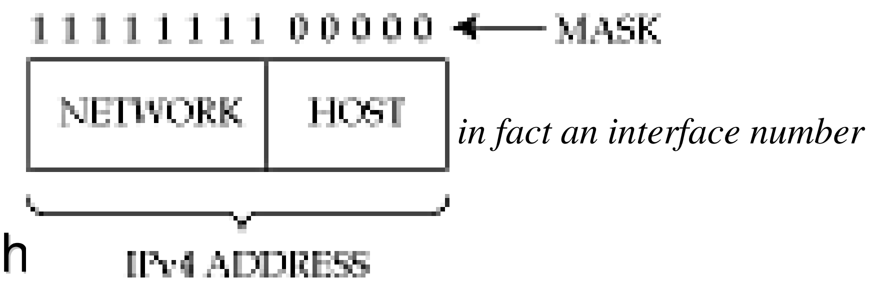
Addressing

- Addresses need to be globally unique, so they are also hierarchical
- Another reason for hierarchy: *aggregation*
  - reduces size of routing tables
    - impractical to have one entry per destination for the Internet
  - at the expense of longer routes



Addressing in the Internet

- Every *host interface* has its own IP address
- Routers have multiple interfaces, each with its own IP address
- Current version of IP is version 4, addresses are IPv4 addresses



- 4 bytes long, two-part h
  - network number and host number
  - boundary identified with a *subnet* mask
  - can aggregate addresses within subnets (network number based routing)

Address classes

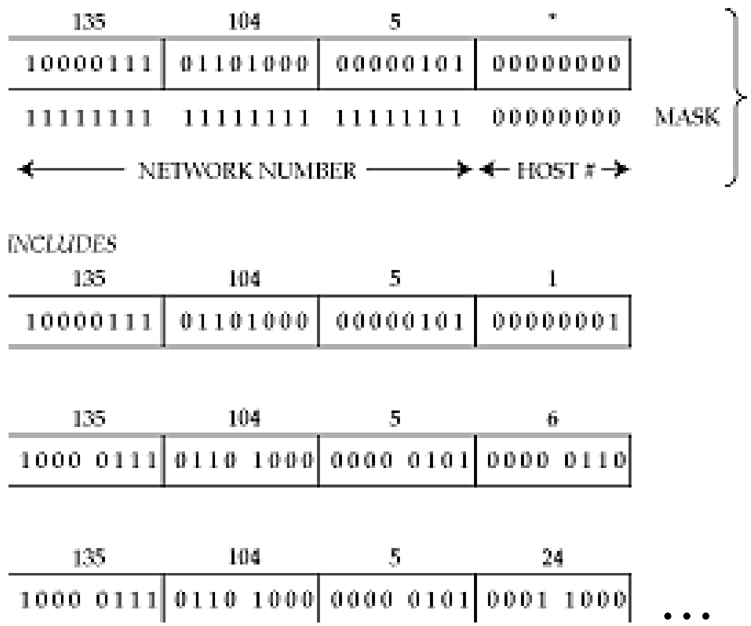
- First cut
  - fixed network-host partition, with 8 bits of network number
  - too few networks!
- Generalization
  - Class A addresses have 8 bits of network number
  - Class B addresses have 16 bits of network number
  - Class C addresses have 24 bits of network number
- Distinguished by leading bits of address
  - leading 0 => class A (first byte < 128)
  - leading 10 => class B (first byte in the range 128-191)
  - leading 110 => class C (first byte in the range 192-223)

Address evolution

- This scheme to allocate *scarce* resources was too *inflexible*
- Three extensions
  - subnetting
  - CIDR
  - dynamic host configuration

Subnetting

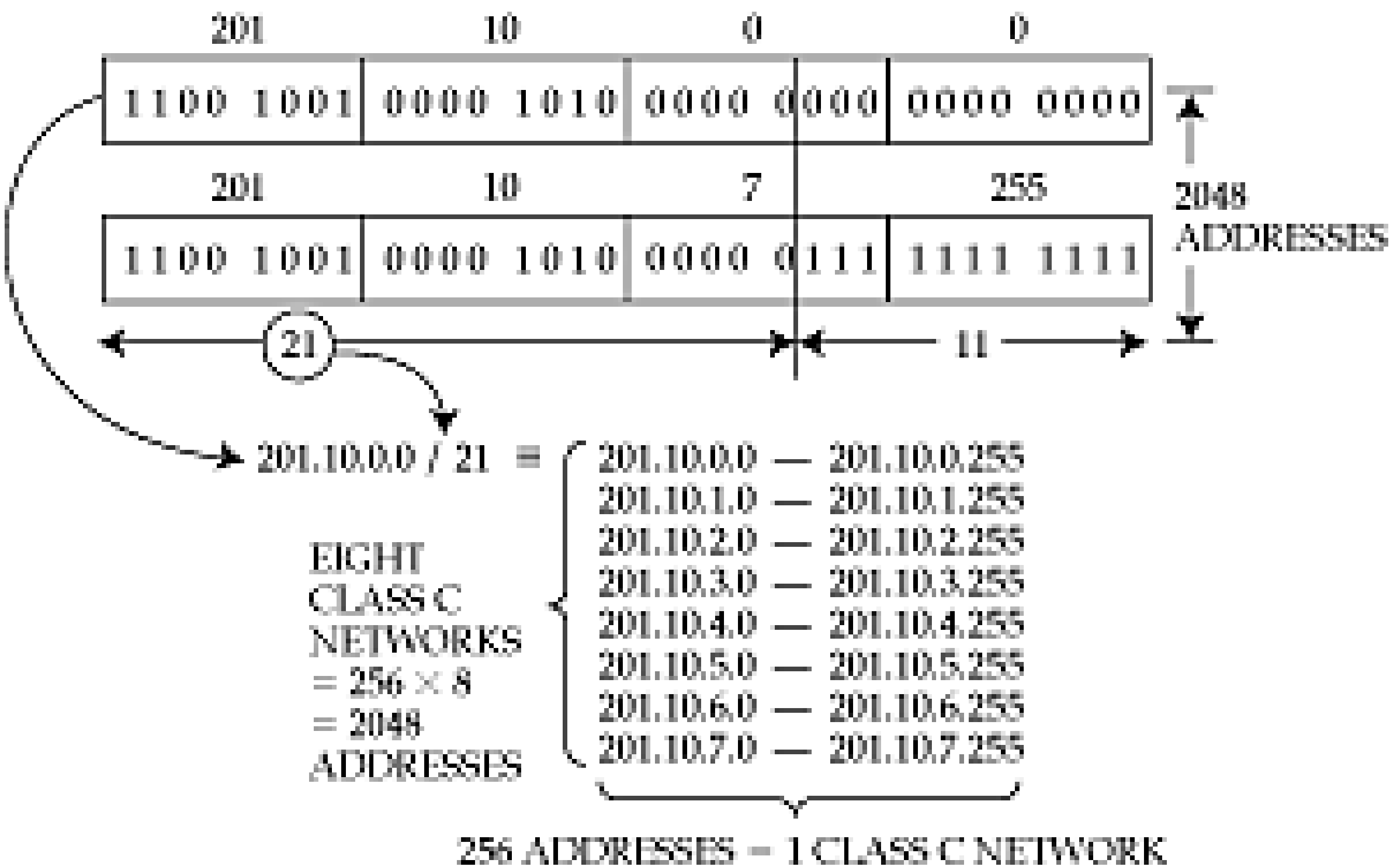
- Allows administrator to cluster IP addresses *within* its network
  - 256 subnet of 256 addresses (e.g. an Ethernet segment)
  - saves space and computation time in subnet routing tables
  - subnet masks are not visible outside the network



CIDR : Classless Interdomain Routing

- Scheme forced medium sized nets to choose class B addresses, which wasted space
- Address space exhaustion (2^{14} = 16382 class B addresses)
- Solution
  - allow ways to represent a contiguous set of class C addresses as a block, so that class C space can be used
  - use a CIDR mask
  - idea is very similar to subnet masks, except that all routers must agree to use it
  - carry a prefix indication: the number of bits of the network number part

CIDR (contd.)



201
10
0
0
1100 1001
0000 1010
0000 0000
0000 0000
201
10
7
255
1100 1001
0000 1010
0000 0111
1111 1111
2048 ADDRESSES
21
11
201.10.0.0 / 21 =
201.10.0.0 — 201.10.0.255
201.10.1.0 — 201.10.1.255
201.10.2.0 — 201.10.2.255
201.10.3.0 — 201.10.3.255
201.10.4.0 — 201.10.4.255
201.10.5.0 — 201.10.5.255
201.10.6.0 — 201.10.6.255
201.10.7.0 — 201.10.7.255
EIGHT CLASS C NETWORKS = 256 × 8 = 2048 ADDRESSES
256 ADDRESSES = 1 CLASS C NETWORK

Dynamic host configuration

- Allows a set of hosts to share a pool of IP addresses
- Dynamic Host Configuration Protocol (DHCP)
- Newly booted computer broadcasts *discover* to subnet
- DHCP servers reply with *offers* of IP addresses
- Host picks one and broadcasts a *request* with the name of a particular server
- All other servers “withdraw” offers, and selected server sends an *ack*
- When done, host sends a *release*
- IP address has a *lease* which limits time it is valid
- Server reuses IP addresses if their lease is over (LRU is wise)
- Similar technique used in *Point-to-point* protocol (PPP)
    - to allocate addresses by ISPs to dial-up hosts

IPv6

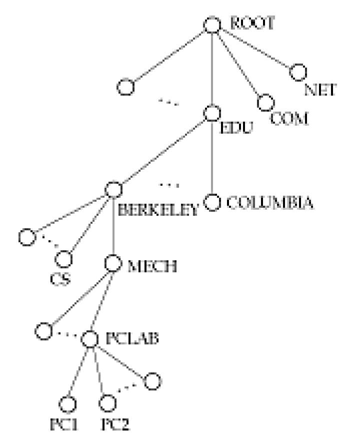
- 32-bit address space is likely to eventually run out
- IPv6 extends size to 128 bits (16 bytes)
- Main features
  - classless addresses (longest prefix match like CIDR)
  - multiple levels of aggregation are possible for *unicast* (IPv6 aggregatable global unicast address RFC[2374])
    - Top level aggregation
    - Next-level aggregation
    - Site-level aggregation
  - several flavors of *multicast*
  - *anycast* (e.g. for partial routes), same syntax as unicast
  - interoperability with IPv4

Name resolution

- Translation done by name servers
- Application send query to a name server:
  - essentially look up a name and return an address
- Centralized design
  - consistent
  - single point of failure
  - concentrates load
- Thus compose name servers from a set of distributed agents
  - that coordinate their action

DNS (Domain Name System)

- Distributed name server
- A name server is responsible (an *authoritative server)* for a set of domains (a subtree of the name space)
- May delegate responsibility for part of a domain to a child
  - things organized so that a name is correctly translated by at least one authoritative Server
- Query is sent to the root of name space
- Parses it and passes to the responsible server

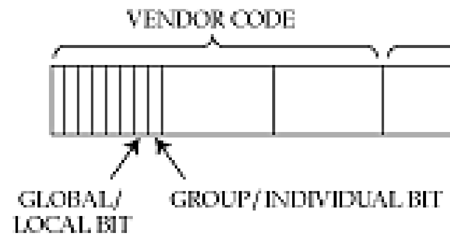


DNS

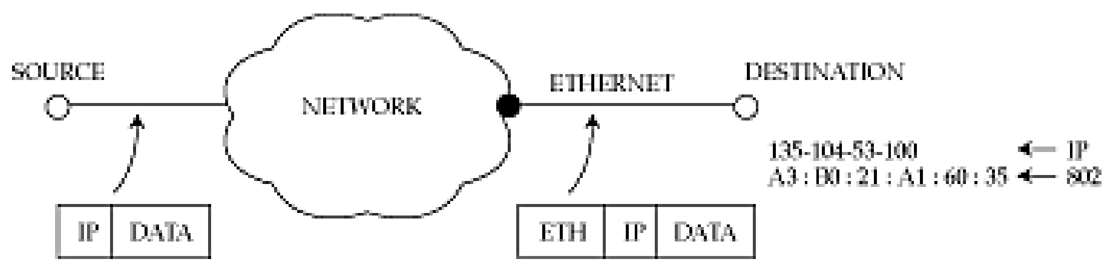
- Heavy load on root servers
  - Root servers are *replicated*
  - requires coordination among servers
  - name resolution query can be made to any replicated server
- *Caching* is also used to reduce load on root servers
- End systems cache and timed out
  - result of the query
  - address of authoritative servers for common domains
- If local server cannot answer a query, it asks root, which delegates reply

Finding datalink layer addresses

- Datalink layer address: most common format is IEEE 802



- Need to know datalink layer address typically for the last hop (in broadcast LANs)



ARP

- To get datalink layer address of a machine on the local subnet
- Broadcast a query with IP dest address onto local LAN
- Host that owns that address (or proxy) replies with address
- All hosts are required to listen for ARP requests and reply
  - including laser printers!
- Reply stored in an ARP cache and timed out
- In point-to-point LANs, need an ARP server
  - register translation with server
  - ask ARP server instead of broadcasting

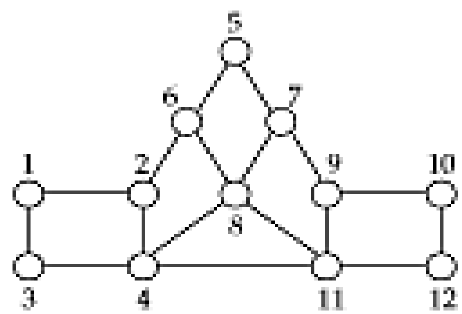
Le routage dans l'Internet

What is it?

- Process of finding a (the best?) 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?
- 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
  - hard to collect
  - inherently large
  - dynamic
- *A routing protocol must intelligently summarize relevant information*

Requirements

- Minimize routing table space
  - fast to look up
  - less to exchange (for some routing protocols)
- Minimize number and frequency of control messages
- Robustness: avoid
  - black holes
  - loops
  - oscillations
- Use optimal path (“best” may be SP, least delay, secure, balances load, lowest monetary cost)
- Trade-offs:
  - robustness vs number of control messages or routing table size
  - reduce table size for slightly “longer” path

Choices

- Centralized vs. distributed routing
  - centralized is simpler, but prone to failure and congestion
- Source-based vs. hop-by-hop (destination address based)
  - 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)
  - not on the Internet (path scarcity and routing table space)
- State-dependent or “dynamic” vs. state-independent
  - do routes depend on current network state (e.g. delay), but risk of oscillations

Outline

- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN

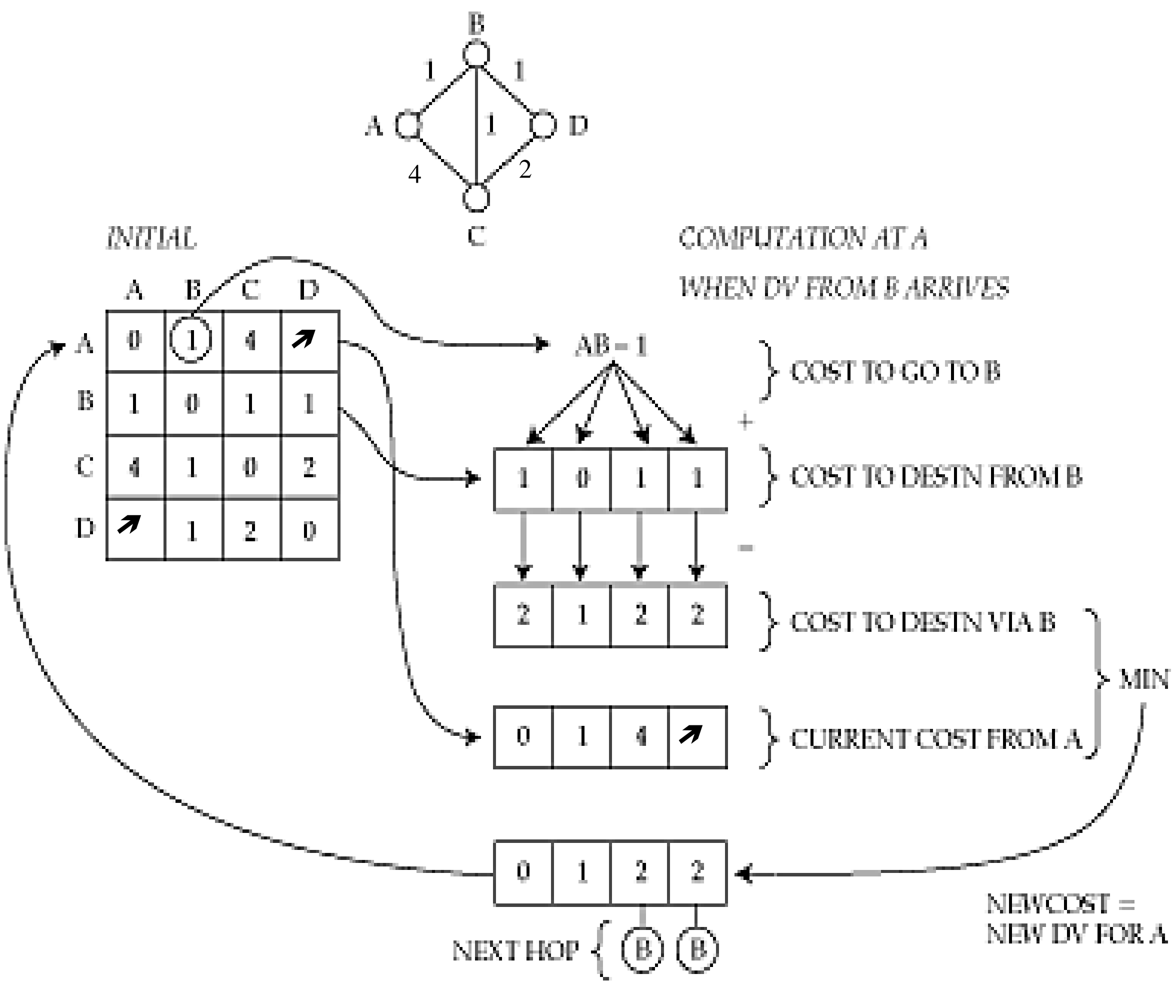
Distance vector routing

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

Basic idea for DV

- Node tells its neighbors its best idea of distance to *every* other node in the network (node identities considered known a priori)
- Node receives these *distance vectors* from its neighbors
  - DV: a list of [destination, cost]-tuples, (next hop info in table)
- 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

Example

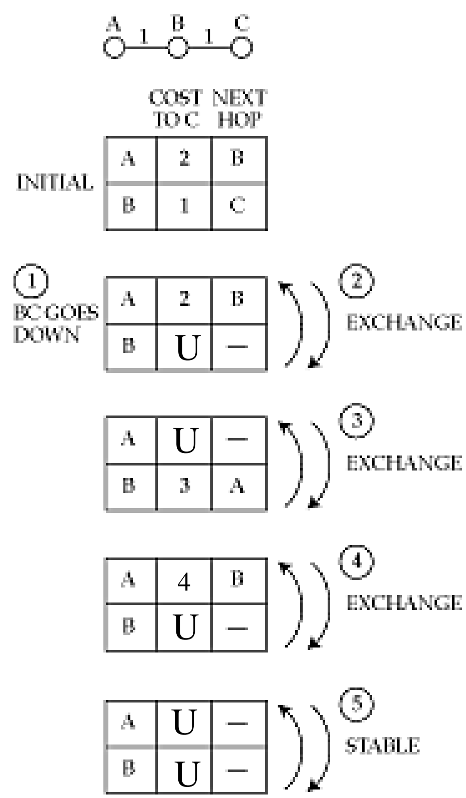

B
A
D
C
1
1
1
4
2
INITIAL
COMPUTATION AT A
WHEN DV FROM B ARRIVES
AB = 1
COST TO GO TO B
COST TO DESTN FROM B
COST TO DESTN VIA B
CURRENT COST FROM A
MIN
NEWCOST =
NEW DV FOR A
NEXT HOP

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
- Used in the Routing Information Protocol (RIP)

Problems with distance vector

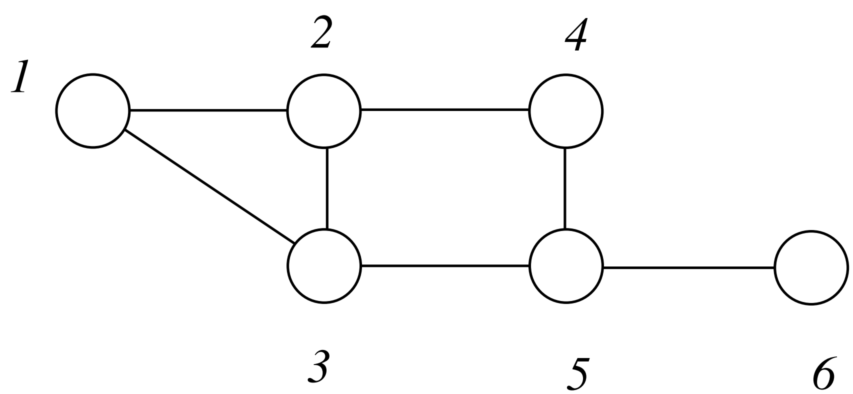
- Works well if nodes are always up
  - problems when links go down or come up
  - DV approach hides details to compute the vector
- Count to infinity



Dealing with the problem

- Path vector
  - DV carries path to reach each destination
  - Trade larger rtg table & extra control overhead **for** robustness
- Split horizon
  - never tell neighbor cost to X if neighbor is next hop to X
  - with poisonous reverse: tell neighbor cost is infinity (faster convergence in some cases)
  - doesn't work for 3-way count to infinity (assume BA then CA go down in slide 31)
- Triggered updates
  - exchange routes on link failure, instead of on timer
  - faster count up to infinity
- More complicated
  - source tracing (same information as path vector with little additional space)
  - DUAL (Distributed Update ALgorithm)

Source tracing



Destination	Next	Last
1	_	_
2	2	1
3	3	1
4	2	2
5	2	4
6	2	5

DUAL (Distributed Update ALgorithm)

- Avoids loops even in presence of rapid changes
- Router keeps a pared down *topology*
  - sorted union of DVs
- Upon reception of a DV
  - Updates table only if cost decreases (no loop may occur in this case)
  - If cost increases (link's cost or link failure)
    - check in topology table if shorter path exists
    - if not
      - freeze routing table
      - distribute new DV to all neighbors (recursively) (*expand* until all affected routers know of change)
    - unfreeze and inform "previous" router
    - contract until first router knows that all affected are aware
- Used in EIGRP (Cisco)

Outline

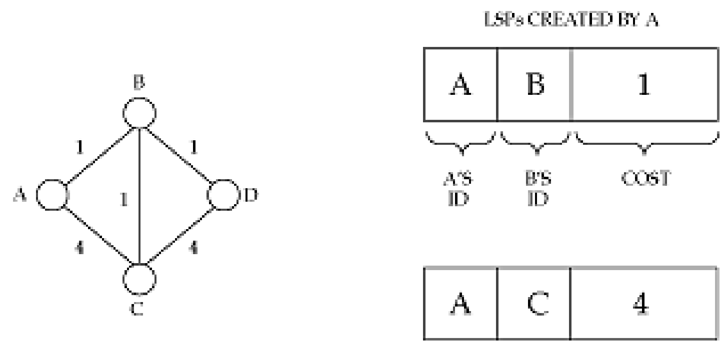
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN

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
  - loop free if same view of topology and same algorithm
- Key elements
  - topology dissemination
  - computing shortest routes

Topology dissemination

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



- 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

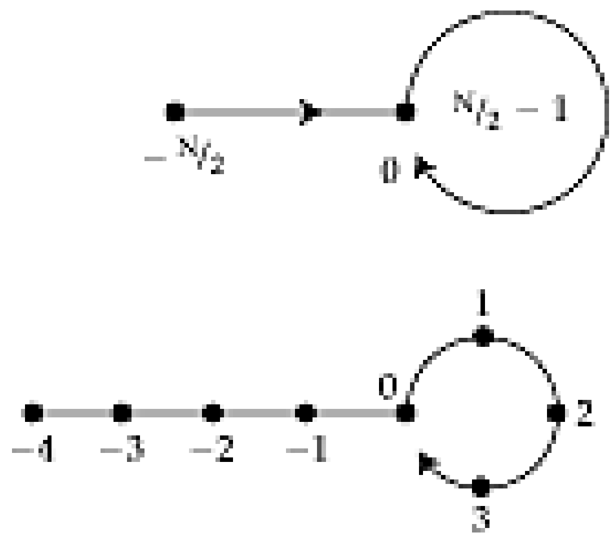
Sequence numbers

- How do we know an LSP is new?
  - Needed to purge “old” information (e.g. after a link failure)
- Use a sequence number in LSP header
- Greater sequence number is newer
- What if sequence number wraps around?
  - smaller sequence number is now newer!
  - Use a large sequence space + comparison on the circle
- But, on boot up, what should be the initial sequence number?
  - have to somehow purge old LSPs
  - two solutions
    - aging
    - lollipop-space sequence numbers

Aging

- Source 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
- So, on booting, router just has to wait for its old LSPs to be purged
- But what age to choose?
  - if too small
    - old LSP could be purged before new LSP fully flooded
    - needs frequent updates
  - if too large
    - router waits idle for a long time on rebooting

A better solution



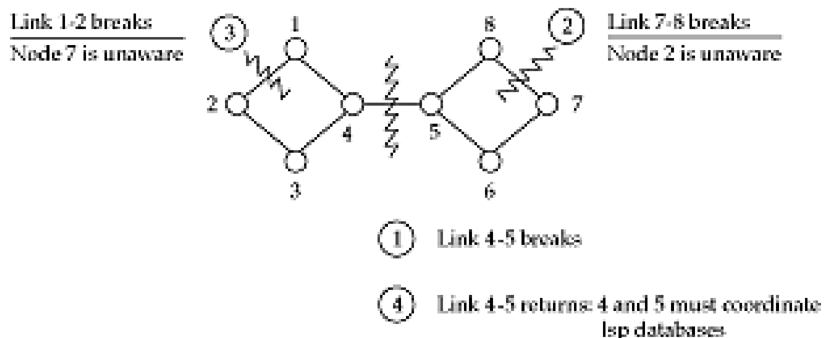
- Need a *unique* start sequence number
- a is older than b if:
  - a < 0 and a < b
  - a > 0, a < b, and b-a < N/4
  - a > 0, b > 0, a > b, and a-b > N/4

More on lollipops

- Additional rule: if a router gets an older LSP, it tells the sender about the newer LSP sequence number
- 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 (inconsistent)



- Databases described by database descriptor records
  - descriptor is link id + version number
- Routers on each side of a newly restored link exchange database descriptors to update databases (determine missing and out-of-date LSPs)

Link or router failure

- *Link* failure easy to detect and recover from
  - Router floods this information
- How to detect *router* failure?
  - HELLO protocol
  - Neighbor floods information about router failure if no response to *N* HELLO packet
- HELLO packet may be corrupted (dead router considered alive!)
  - so age anyway (even with lollipop-space sequence numbers)
  - on a timeout, flood the information

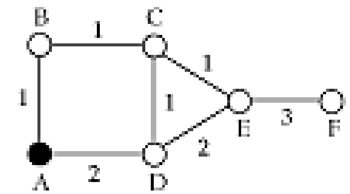
Securing LSP databases

- LSP databases *must* be consistent to avoid routing loops
- Malicious agent may inject spurious LSPs
- Routers must actively protect their databases
  - checksum LSPs even when stored in the database
    - detects corruption on *link* or *disk*
  - ack LSP exchanges
  - authenticate LSP exchanges using passwords

Computing shortest paths

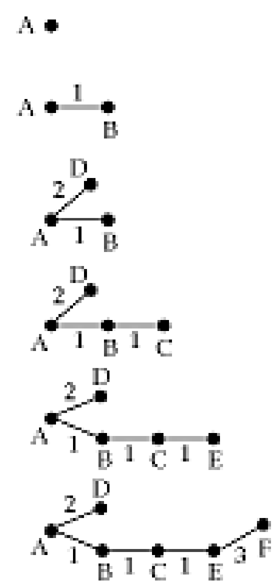
- Based on Dijkstra's shortest path algorithm
  - computes SP from a "root" to every other node
- Basic idea
  - maintain a set of nodes P to whom we know shortest path
  - initialize P to root
  - consider set {every node one hop away from nodes in P} = T
  - find every way in which to reach a given node in T from root, 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 and loop freeness (+LS)
    - in LS routers know entire topology, but transient loops can form (during topology changes flooding)
    - simple modification to vanilla DV algorithm can prevent loops
  - multiple routing metrics (+LS)
    - requires *all* routers agree to report same metrics
  - convergence time after a change (+LS)
    - DV with triggered updates + DUAL has also fast convergence
  - communication overhead (+DV)
    - Nodes are not required to independently compute consistent routes in DV (in LS high overhead to ensure database consistency)
  - memory overhead (+DV)
    - Advantage lost if we use path vector
- Both are evenly matched
- Both widely used (OSPF, BGP)

Outline

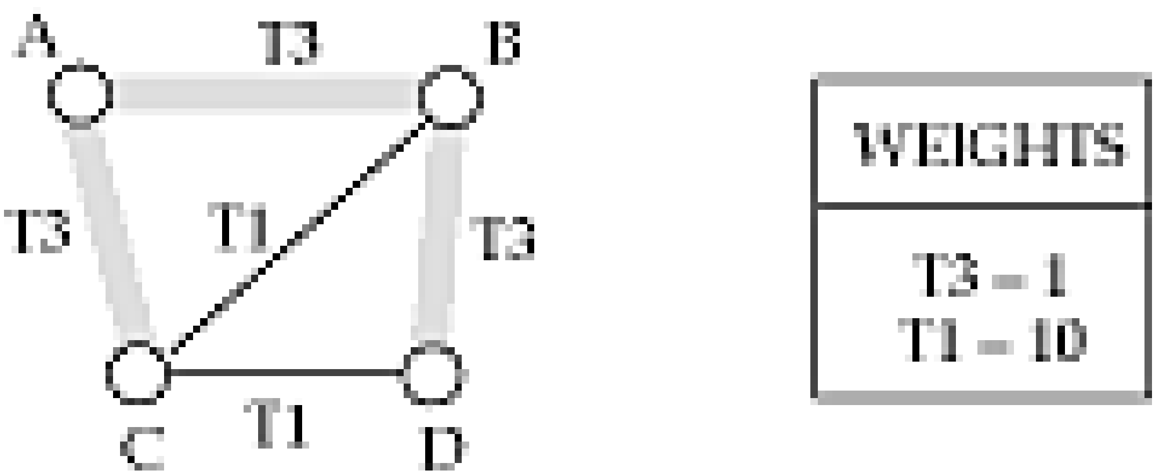
- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN

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

Static metrics

- Simplest: set all link costs to 1 => min hop routing
  - but 56K modem link is not the same as a T3!
- Enhancement: give links weight inversely proportional to capacity
- But therefore BC and CD are not used even if T3 are congested



Dynamic metrics

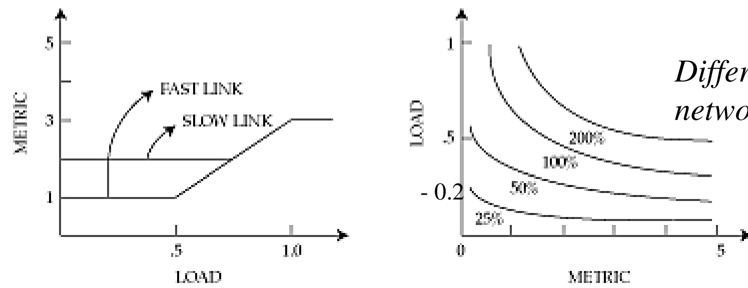
- A first cut (ARPAnet original)
- Cost proportional to length of router queue
  - independent of link capacity
- Unintended consequences of complex design!
  - Many problems when network is loaded
    - queue length averaged over a too small time (10 s) : transient spikes in queue length caused major rerouting
    - cost had wide dynamic range => network completely ignored paths with high costs
    - queue length assumed to predict future loads => opposite is true
    - no restriction on successively reported costs => large oscillations
    - all tables computed simultaneously => low cost links 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 (3:1), cost hop normalized
- cost also depends on intrinsic link capacity
  - on low load cost depends only on capacity
- restriction on successively reported costs (1/2 hop)
- attempt to stagger table computation

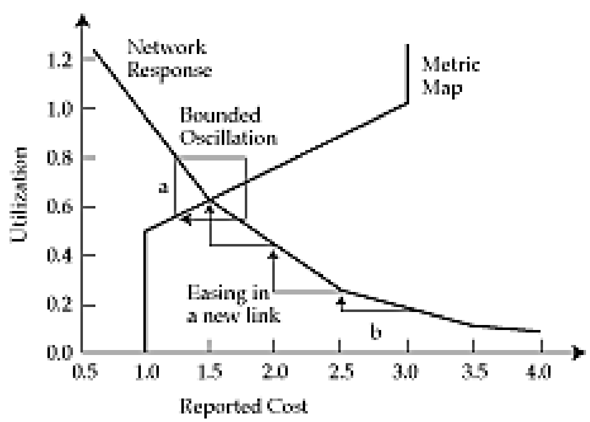
Routing dynamics



(a) METRIC MAP

(b) NETWORK RESPONSE MAP

Different overall network loads



Case a: stable state cost 1.25, U 0.8, m 1.75, U 0.6, m 1.25, …

Case b: new link with high metric, entering steady state

Are dynamic metrics used?

- Not widely used in today’s Internet
- hard to control amount of routing updates a priori
  - dependent on network traffic
- Still can cause oscillations

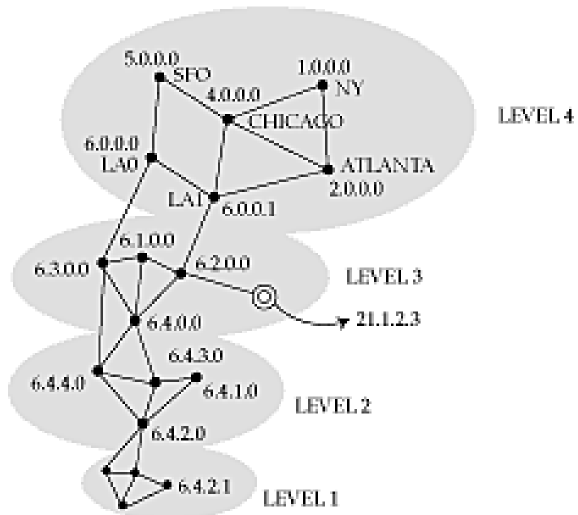
Outline

- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN

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 (both LA_i carry packets to 6.*)
  - gateways participate in multiple routing protocols
  - non-aggregable routes increase core table space (21.1.2.3)

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

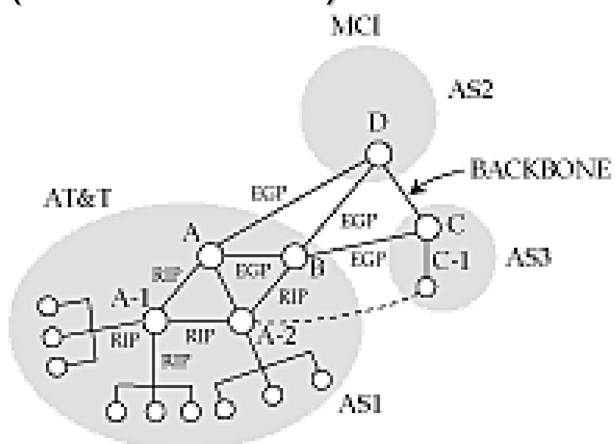
- If a domain has multiple gateways
  - *external* records tell hosts in a domain which one to pick to reach a host in an external domain
    - e.g allows 6.4.0.0 to discover shortest path to 5.* is through 6.0.0.0
  - *summary* records tell backbone which gateway to use to reach an internal node
    - 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

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 (A-D-B is not!)



- *Transit* over *backdoors* is a problem (A2-C1 should not be summarized)

Interior protocols

- Much easier to implement
  - free of administrative “problems” : no manual configuration
- Typically partition an AS into *areas*
- Exterior and summary records used between areas

Issues in interconnection EGPs and IGPs

- May use different schemes (DV vs. LS)
- Cost metrics may differ
  - 5 hops for an IGP $\neq$ 5 hops inter-AS
- Need to:
  - convert from one scheme to another
  - use the least common denominator for costs
    - Hop-count metric!
  - manually intervene if necessary

Outline

- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN

Common routing protocols

- Interior
  - RIP
  - OSPF
- Exterior
  - EGP
  - BGP

RIP

- Distance vector
- Cost metric is hop count
- Infinity = 16
- Exchange distance vectors every 30 s
- Split horizon with poisonous reverse
- 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 on a broadcast LAN

EGP

- Original exterior gateway protocol
- Distance-vector
- Costs are either 128 (reachable) or 255 (unreachable)
  - only propagates reachability information
  - ⇒ backbone must be structured as a tree to ensure loop free
- Allows administrators to pick neighbors to peer with
- Allows backdoors (by setting backdoor cost < 128)
  - not visible to outside systems
- No longer widely used
  - need for loop free topology

BGP

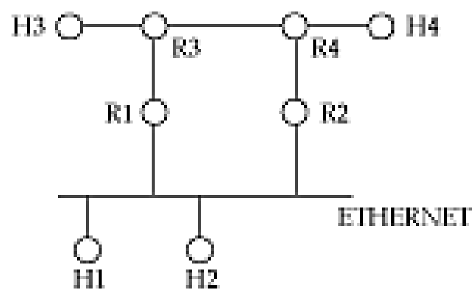
- Path-vector
  - distance vector annotated with entire path
  - also with policy attributes (no cost information)
  - guaranteed loop-free
- Can use non-tree backbone topologies
  - uses true cost (not like EGP)
- Uses TCP to communicate between routers
  - reliable
  - but subject to TCP flow control
- BGP provides the mechanisms to distribute path information
- But leaves (complex) policies to network administrator

Outline

- Distance-vector routing
- Link-state routing
- Choosing link costs
- Hierarchical routing
- Internet routing protocols
- Routing within a broadcast LAN

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
- Local destination’s datalink address determined using ARP
  - broadcast a request
  - owner of IP address replies
- To discover routers (default for non local packets)
  - routers periodically sends router advertisements
    - with preference level and time to live (typ. 30 min)
  - pick most preferred router
  - flush when TTL expires
  - can also force routers to reply with *solicitation message* (after a boot)

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