UiO : Faculty of Mathematics and Natural Sciences
University of Oslo

Department of Informatics
Networks and Distributed Systems (ND) group

INF 3190
The Network Layer

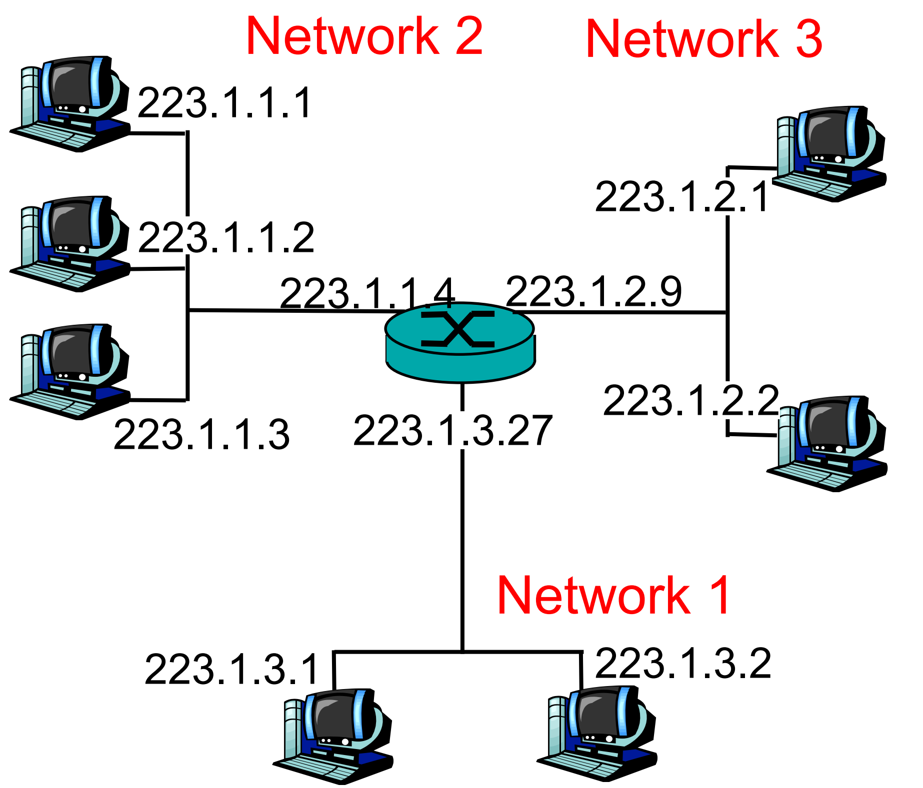
Michael Welzl

How to contact another Internet host?

- OS knows IP address of DNS server (preconfigured)
  - DNS used to be hosts.txt, now a distributed database of sorts
  - DNS query sent using UDP (User Datagram Protocol)
- UDP entity in the OS
  - adds UDP header, sends packet using IP (Internet Protocol)
- IP entity in the OS
  - adds IP header, sends packet using Ethernet Driver
- Ethernet Driver
  - queries local table for MAC (Medium Access Control) address
  - If not found, broadcast "who has IP address ... (DNS server)?" using ARP (Address Resolution Protocol) on LAN - answer is correct address
- DNS response = dest. IP address, repeat above procedure...

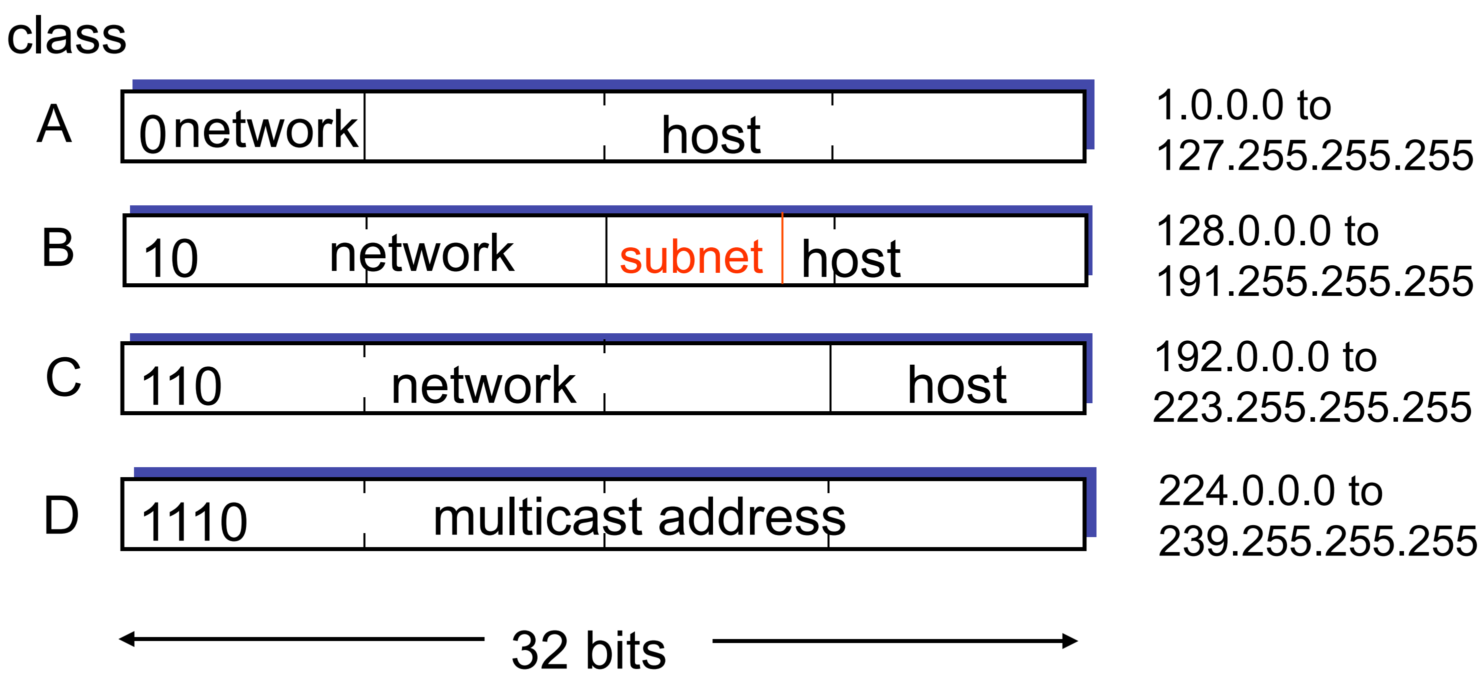
IP Addressing

- IP address: 32-bit identifier for host, router *interface*
  - network part (high order bits)
  - host part (low order bits)
- *interface:* connection between host, router and physical link
  - routers typically have multiple interfaces
  - host may have multiple interfaces
- *What's a network ?*
  - device interfaces with same network part of IP address
  - can physically reach each other without intervening router



network consisting of 3 IP networks (for IP addresses starting with 223, first 24 bits are network address)

Classful (traditional) Addressing



- Inefficient use of address space, address space exhaustion
- e.g., class B net allocated enough addresses for 65K hosts, even if only 2K hosts in that network; also, routing table with 65k entries = inefficient
  - subnetting: take highest bits from host to implement a subnet number
  - split defined via subnet mask (known to local routers but not outside)

Classless InterDomain Routing (CIDR)

network part ⟵ ⟶ host part

11001000 00010111 00010000 00000000

200.23.16.0/23

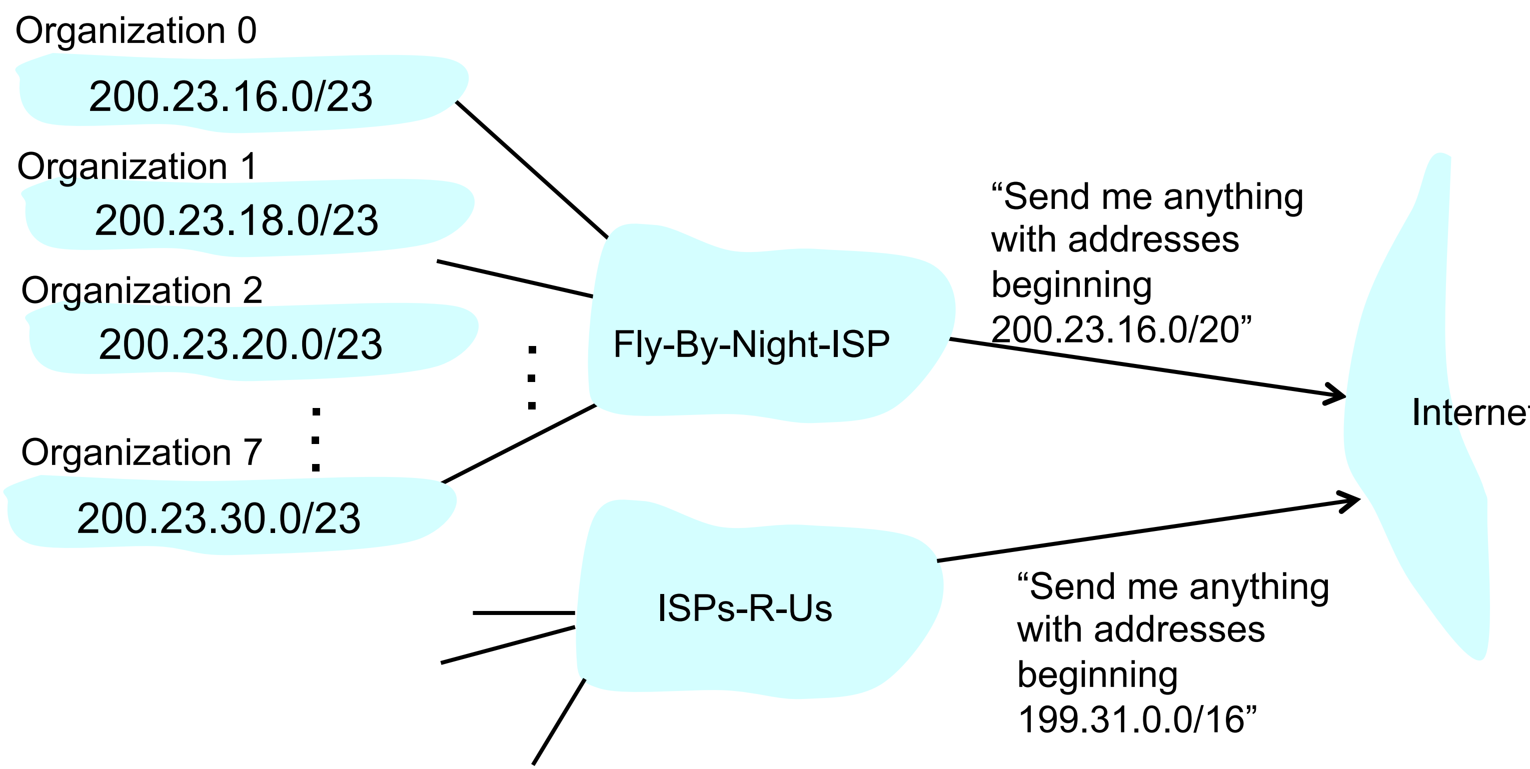
- More efficient use of address space
- Address format: a.b.c.d/x, where x is # bits in network portion
- network portion: arbitrary length
  - get allocated portion of ISP's address space

assigned by ICANN: Internet Corporation for Assigned Names and Numbers

ISP's block	11001000 00010111 00010000	00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000	00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010	00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100	00000000	200.23.20.0/23
...	.....	....	....
Organization 7	11001000 00010111 00011110	00000000	200.23.30.0/23

Hierarchical addressing: route aggregation

Hierarchical addressing allows efficient advertisement of routing information:



Getting a datagram from source to dest.

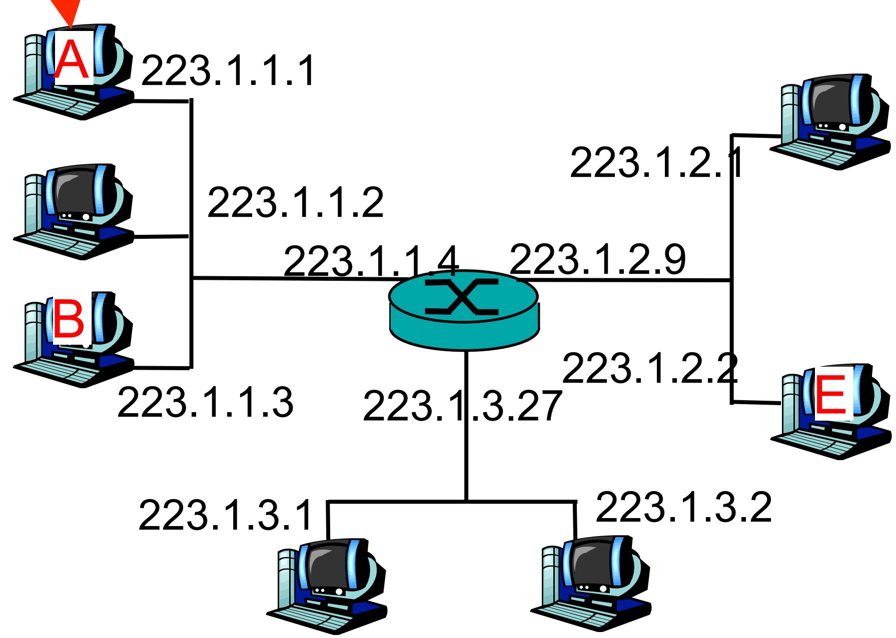
routing table in A

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

IP datagram:

misc fields	source IP addr	dest IP addr	data

- datagram remains unchanged, as it travels source to destination
- addr fields of interest here



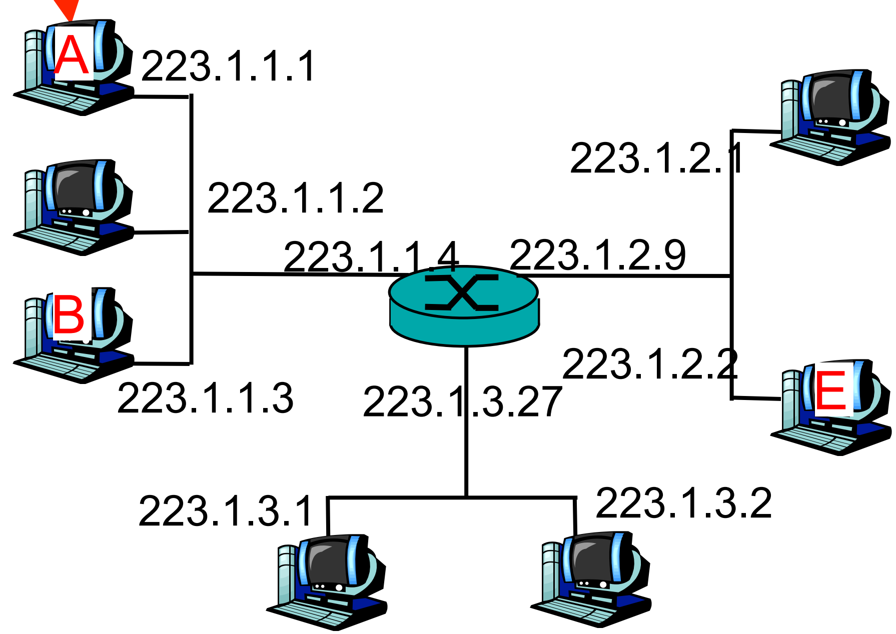
Getting a datagram from source to dest. /2

misc fields	223.1.1.1	223.1.1.3	data

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

Starting at A, given IP datagram addressed to B:

- look up net. address of B
- find B is on same net. as A
- link layer will send datagram directly to B inside link-layer frame
  - B and A are directly connected



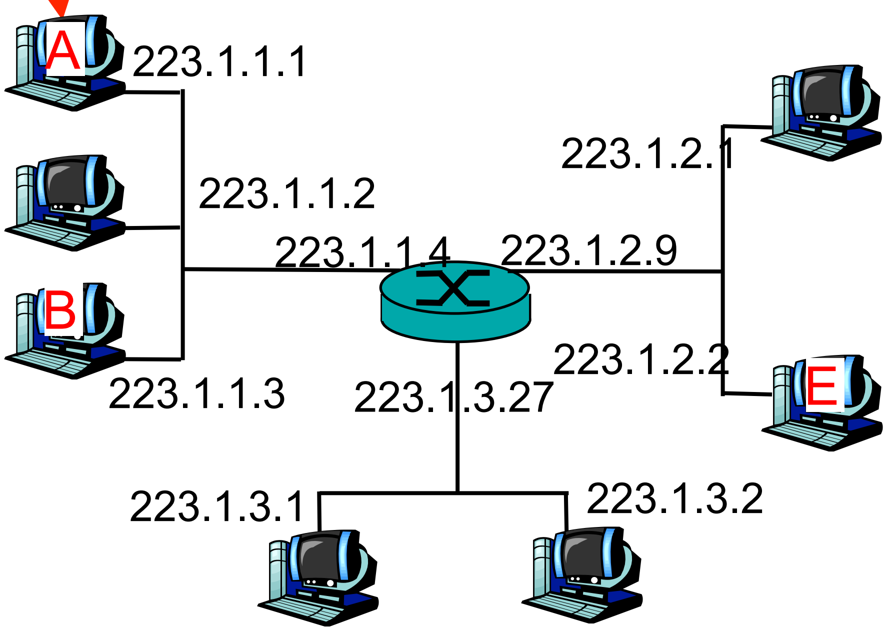
Getting a datagram from source to dest. /3

misc fields	223.1.1.1	223.1.2.2	data

Dest. Net.	next router	Nhops
223.1.1		1
223.1.2	223.1.1.4	2
223.1.3	223.1.1.4	2

Starting at A, dest. E:

- look up network address of E
- E on *different* network
  - A, E not directly attached
- routing table: next hop router to E is 223.1.1.4
- link layer sends datagram to router 223.1.1.4 inside link-layer frame
- datagram arrives at 223.1.1.4
- continued…..



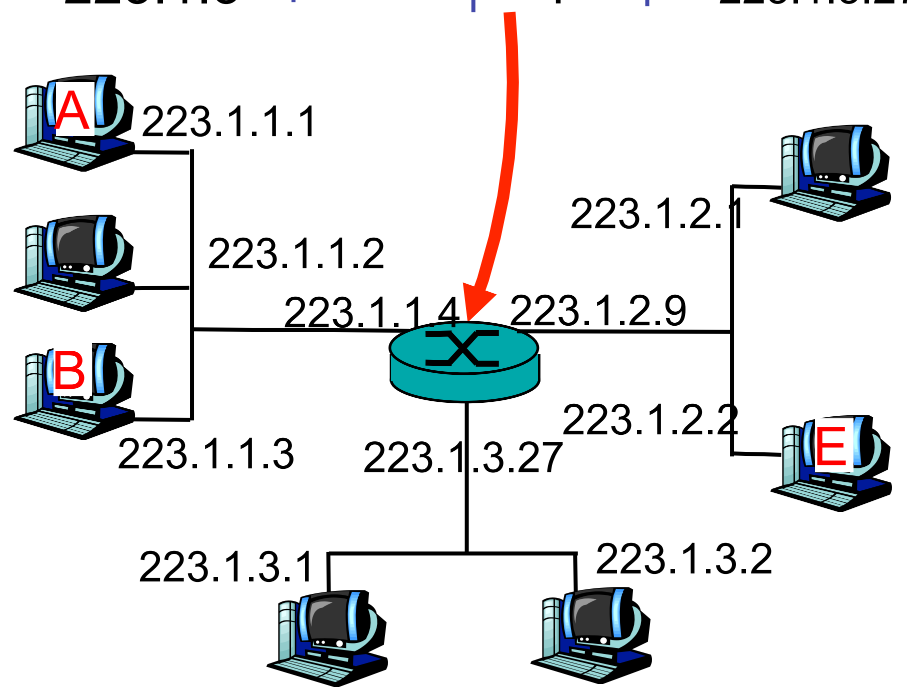
Getting a datagram from source to dest. /3

misc fields	223.1.1.1	223.1.2.2	data

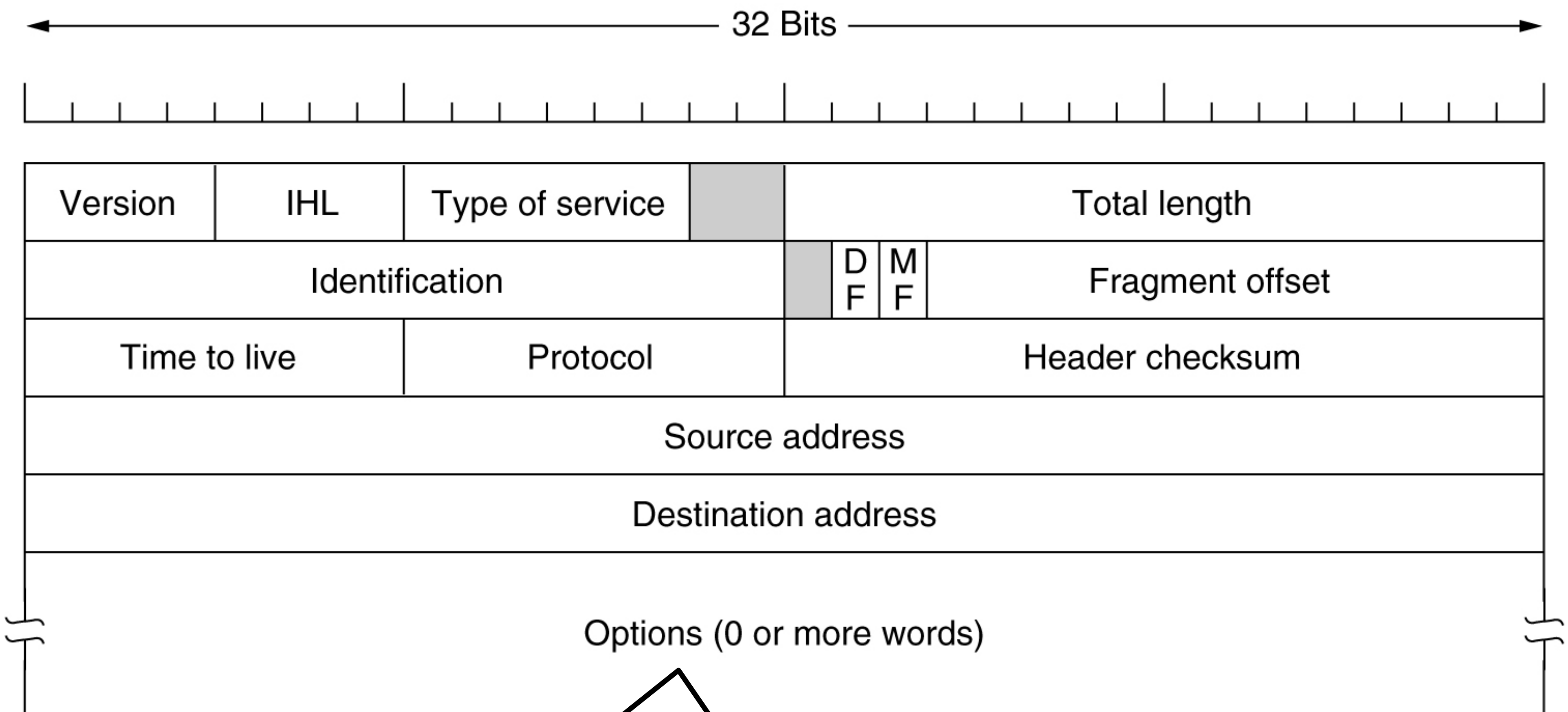
Dest. network	next router	Nhops	interface
223.1.1	-	1	223.1.1.4
223.1.2	-	1	223.1.2.9
223.1.3	-	1	223.1.3.27

Arriving at 223.1.4, destined for 223.1.2.2

- look up network address of E
- E on *same* network as router's interface 223.1.2.9
  - router, E directly attached
- link layer sends datagram to 223.1.2.2 inside link-layer frame via interface 223.1.2.9
- datagram arrives at 223.1.2.2!!! (hooray!)



IPv4 Header



Examples: Loose / Strict Source Routing, Record Route, Timestamp, Router Alert

Internet Protocol Version 6 (IPv6, IPNG)

- Initial motivation: 32-bit address space completely allocated by 2008.
  - Internet growth immense - MANET scenarios etc. etc.
- Additional motivation:
  - header format helps speed processing/forwarding
  - header changes to facilitate QoS
  - new “anycast” address: route to “best” of several replicated servers
- Some header fields changed / removed
  - no checksum → reduce per-hop processing time!
- Multicast - IGMP now part of ICMP
- Mobility - functionality relocated into routers

Transition From IPv4 To IPv6

- Not all routers can be upgraded simultaneous
  - no “flag days”
  - How will the network operate with mixed IPv4 and IPv6 routers?

- Several approaches:
  - Dual Stack: some routers with dual stack (v6, v4) can “translate” between formats (example path: v6 → v6_2_v4 → v4_2_v6 → v6)
  - Tunneling: IPv6 carried as payload in IPv4 datagram among IPv4 routers
  - IPv6/IPv4 network address and protocol translation: serious recent efforts

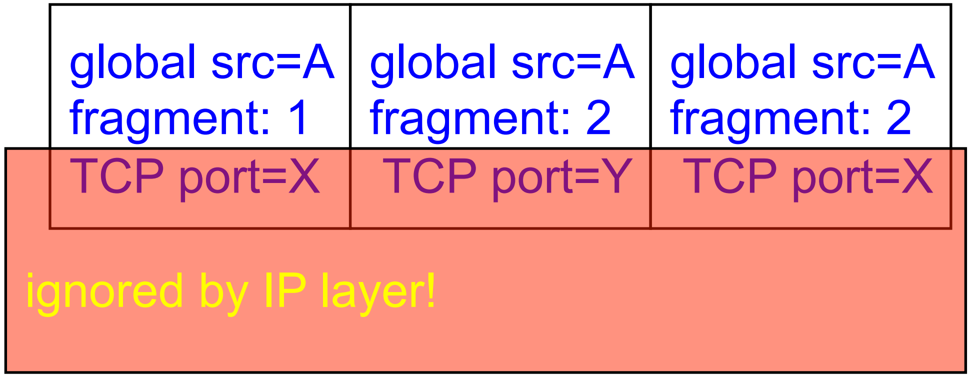
- Short-term solution: Network Address Translator (NAT)
  - assumption: at time t, only x out of y hosts communicate with outside world → use unique translation tables (= state!) to map x local to x global addresses

- NAT extension: NAPT (Network Address / Port Translator)
  - map local ip addr. / (tcp or udp) port no. pair to globally unique ip address / port no.
  - advantage: 1 globally unique ip address can be used by several local hosts at once
  - disadvantage: problems with specific port numbers

Some NAT trouble (there is more!)

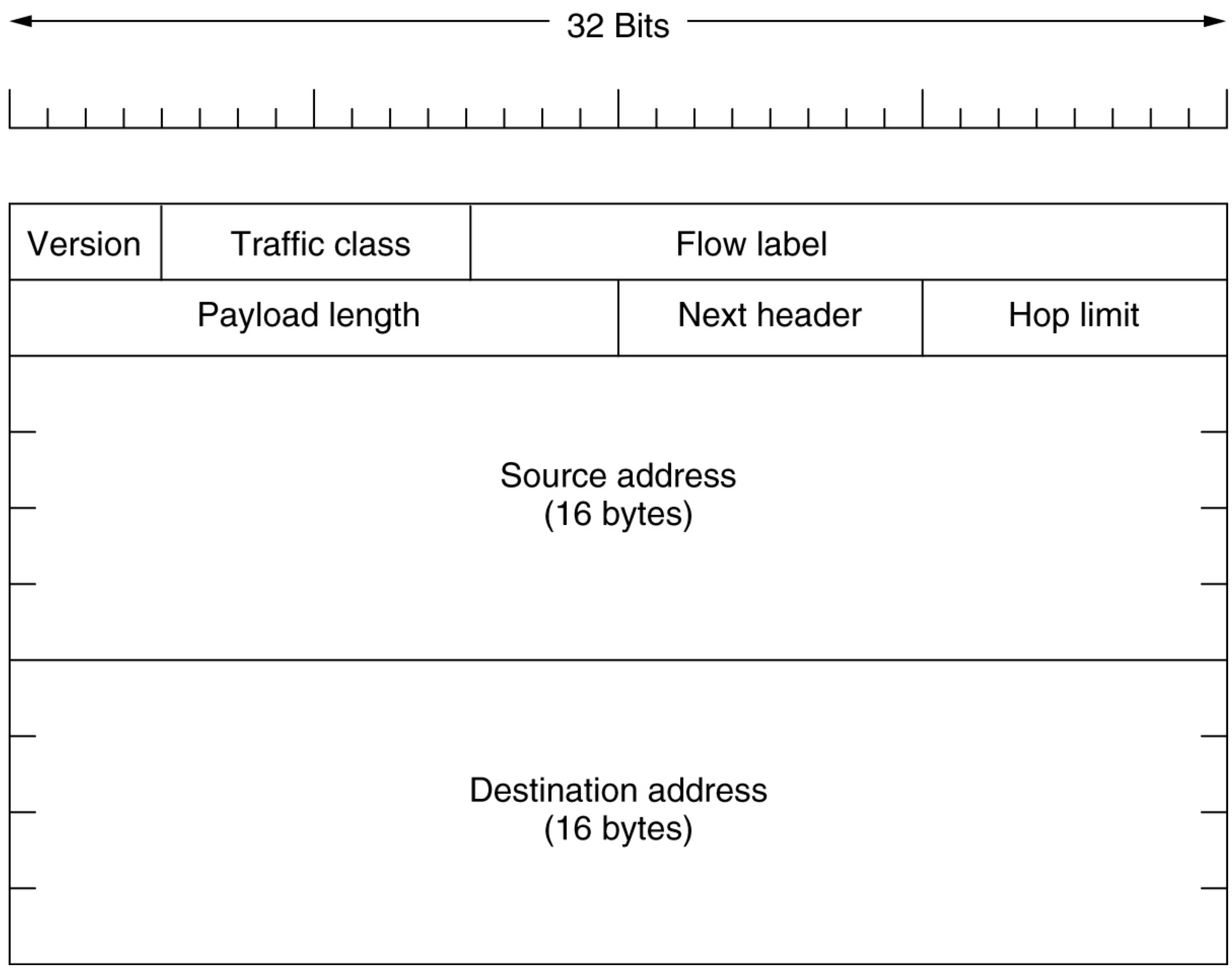
- Problem: realm-specific IP address in payload
  - Solution: per-app treatment by Application Level Gateway (ALG)
    - Problem: [IPsec] encryption at lower layer (transparent to the app)
      - Solution: implement lower-layer decryption + encryption in ALG
        - ...

- IPv4 fragmentation (routers split packets which are too large): semantics of IP address / port number pair lost at IP layer → wrong reassembly!



- NATs break the end-to-end model:
  - complicated functions (state, ..) in the network
  - special per-application functionality at lower layer
  - not possible to contact machines behind NAT directly

Main IPv6 Header



Fragmentation: only in hosts!

Optional: extension headers

Routing: CONS vs. CLNS

Pros + Cons of CONS (vs. CLNS):

+ routing decisions at conn. establishment (more sophisticated?)
+ max. packet size: at time of conn. establishment
  (no fragmentation at intermediate routers needed)
+ packets stay in order
+ congestion control by buffer allocation / conn. refusal
- router / link down: connections break
- no dynamic adaption to load (rare for CLNS, too!)
- routers need "memory" per link

Inverse arguments apply for CLNS over CONS

Note: distinguish a) route finding, b) packet forwarding

Virtual circuits

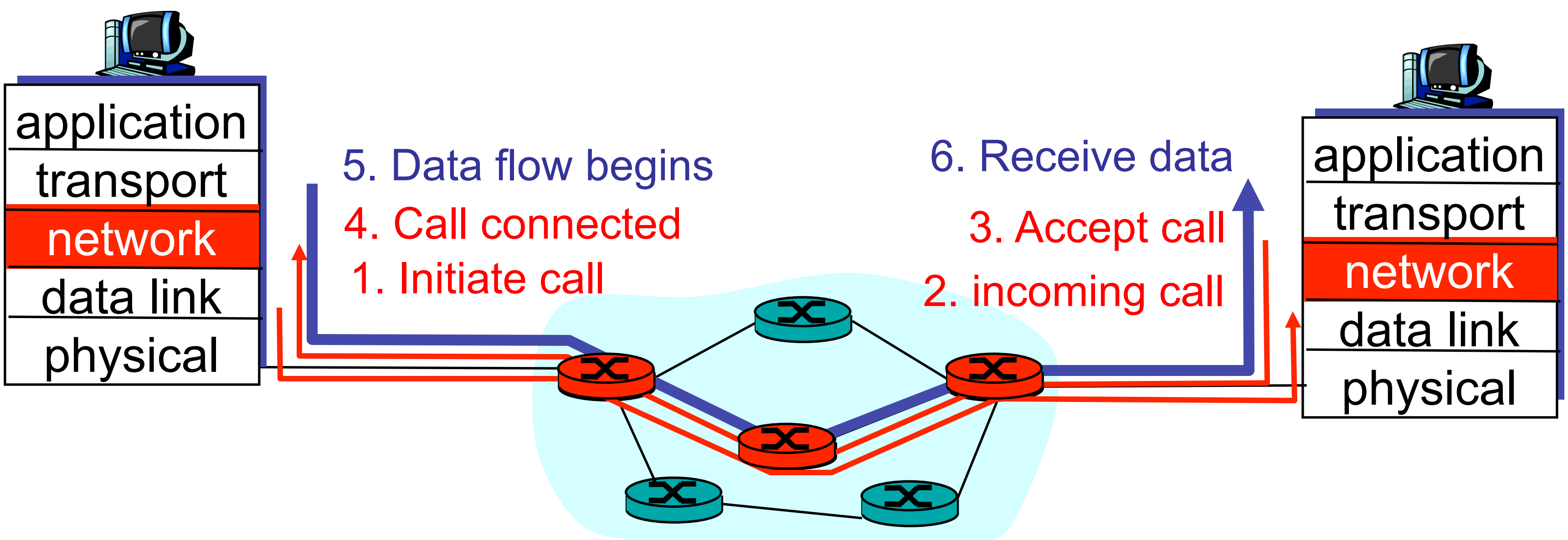
"source-to-dest path behaves much like telephone circuit"

- performance-wise
- network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (not destination host OD)
- *every* router on source-dest path s maintain "state" for each passing connection
  - transport-layer connection only involved two end systems
- link, router resources (bandwidth, buffers) may be *allocated* to VC
  - to get circuit-like perf.

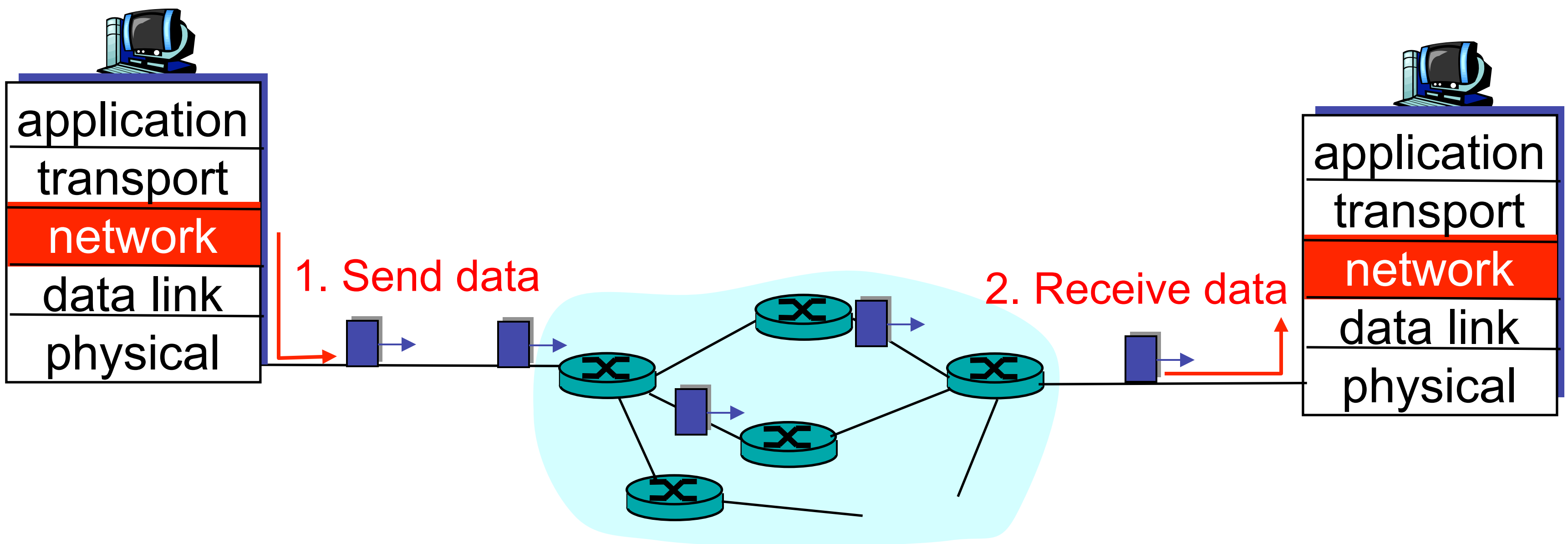
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, frame-relay, X.25
- not used in today's Internet

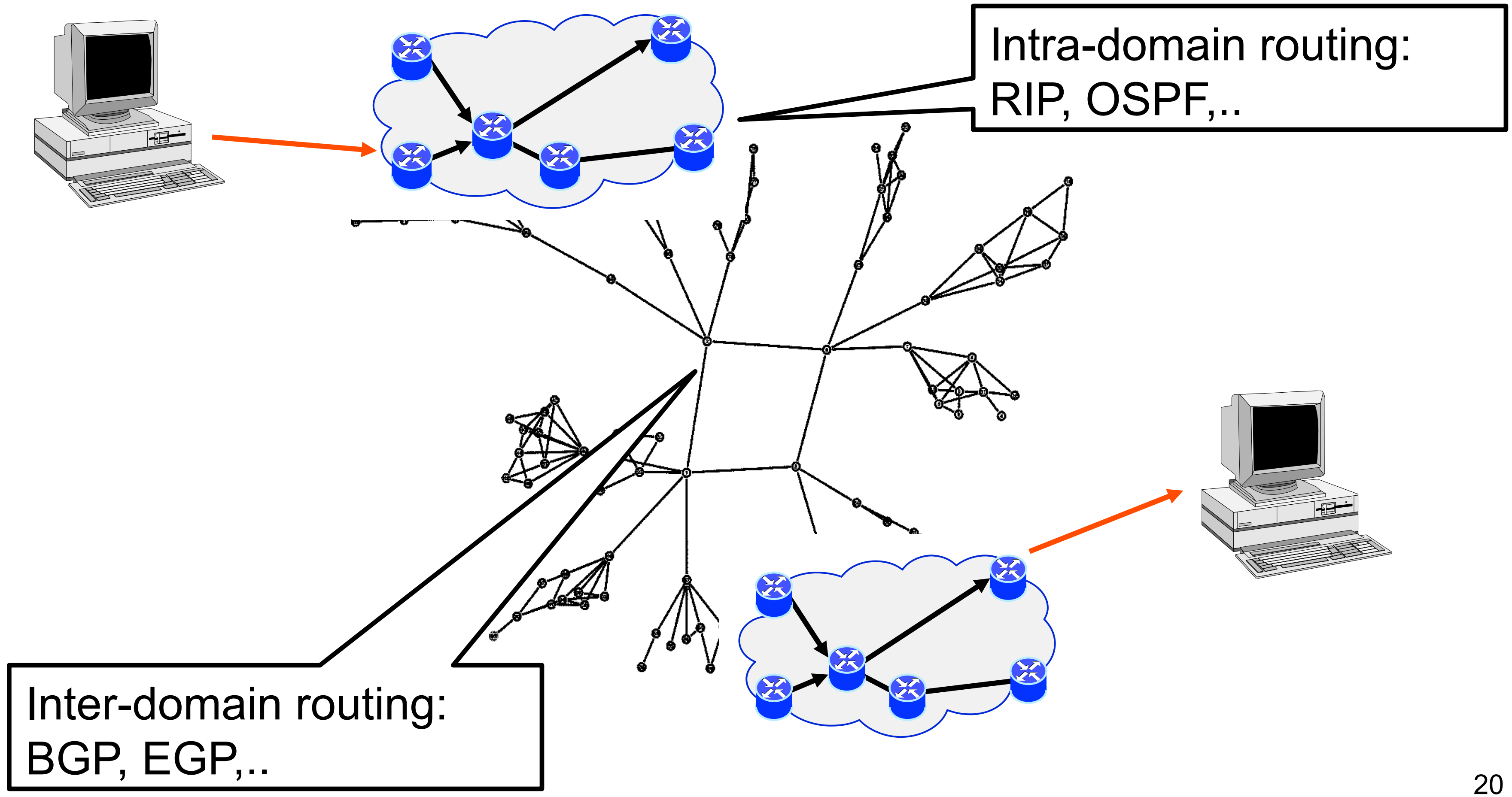


Datagram networks: the Internet model

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets typically routed using destination host ID
  - packets between same source-dest pair may take different paths



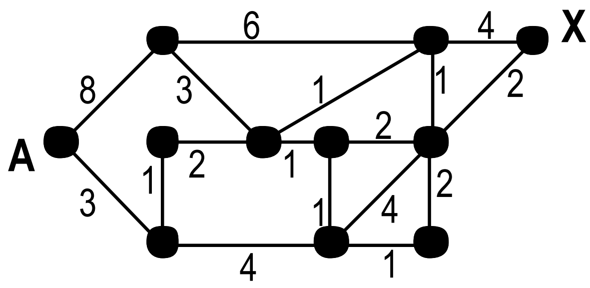
Routing in the Internet



Internet routing is (somewhat) dynamic

- Generated 1998 using “GeoBoy“ from NDG Software
- 10/2003 analysis shows ACONET ⇒ Sprint ⇒ Verio ⇒ wwoz

Routing: L.3 Protocol Function



Optimal Route?

- network considered as graph with nodes and links
- links are weighted by "cost":
  e.g.: distance, mean queue length, service time [1/capacity], $$-cost, …)
- optimal route from A to X: the one with minimal cost
- Usually "min. hops": set all cost to 1

Categories:

- stateful (table-based) / stateless (ad-hoc)
- connection-time / forward-time
- centralized / decentralized
- dynamic / static (how much? usually: adaptation to link/node up/down)
- hierarchical (Internet) or not
- unicast / multicast

ad hoc networks!

Routing: Flooding, Hot Potato

Flooding (stateless / ad-hoc, decentralized)

- send all incoming packets on each outgoing line → some copies will make it
- silly? a) useful for „route discovery"; b) can be improved:
  - don't send packet back or in "wrong direction": needs *some* knowledge (e.g., east/west)
  - identify packets (create unique ID or keep entire copy at routers) + add a sequence counter (increased per router): "I have seen *this* packet, with lower seq.no." → discard
    note: this (best) version requires "states", "memory" (scalability problem)
  - stateless enhancement: Time To Live (TTL) counter (decr. per router), 0 → discard

Hot Potato (stateless / ad-hoc, decentralized)

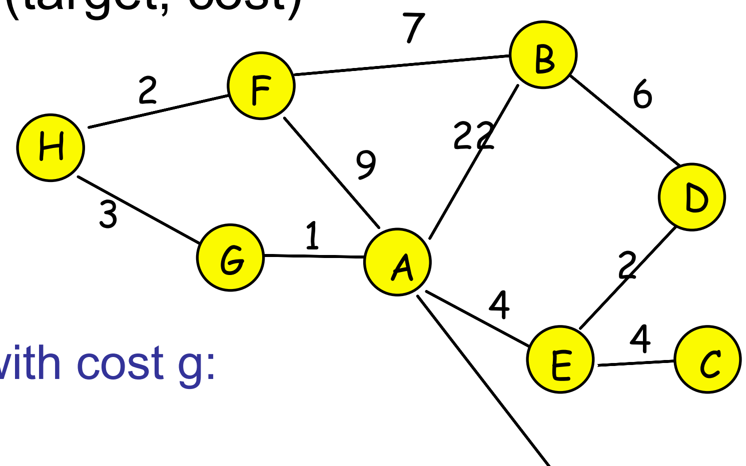
- route packet on link(s) with shortest queue (except incoming)
- good for highly connected net's, today hardly used at all

Centralized Routing (obsolete):

- "Routing Control Center" RCC collects info, computes optimum, broadcasts results
  - dense routing-related traffic "close to" RCC
  - remote routers tend to have older info then those close to RCC
  - single point of failure + scalability problem: frequent changes in large nets!

Distance Vector Routing

- Build local routing table based on info from adjacent routers
- table does not hold entire path, but triples [target, cost, via]
- distance vector = transmitted list of pairs (target, cost)



to	length	via
A	0	-
B	22	B
C	∞	?
D	∞	?
E	4	E
F	9	F
G	1	G
H	∞	?

Bellman-Ford Algorithm:

- each node knows cost for neighbours
- „start“: send table to neighbours

  upon receipt of table from sender s via path with cost g:

  for all rows i in table

  if (msg_table.cost[i]+g < my_table.cost[i]) or (s = my_table.via[i]):

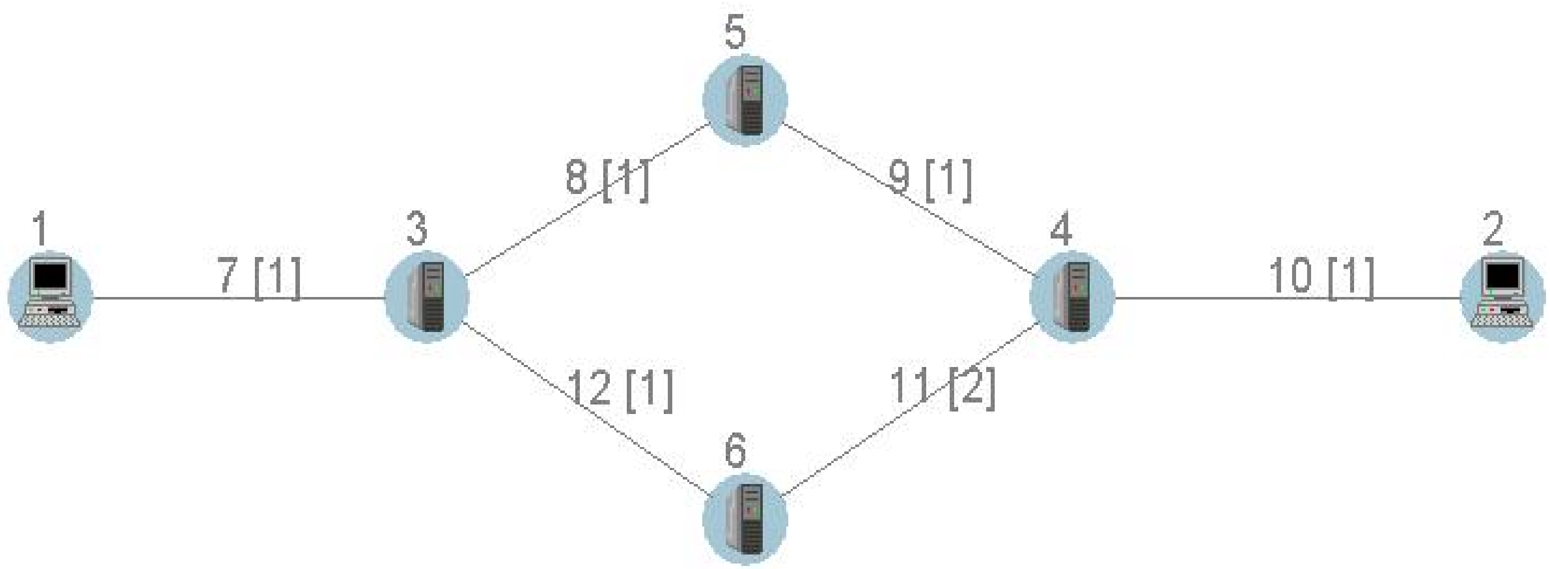
  replace for row i: length := msg_table.cost[i]+g; via := s

  if there was a change:

  send new table to all neighbours except sender

- repeat periodically
- Note: real implementation *discovers* network topology (new entries)

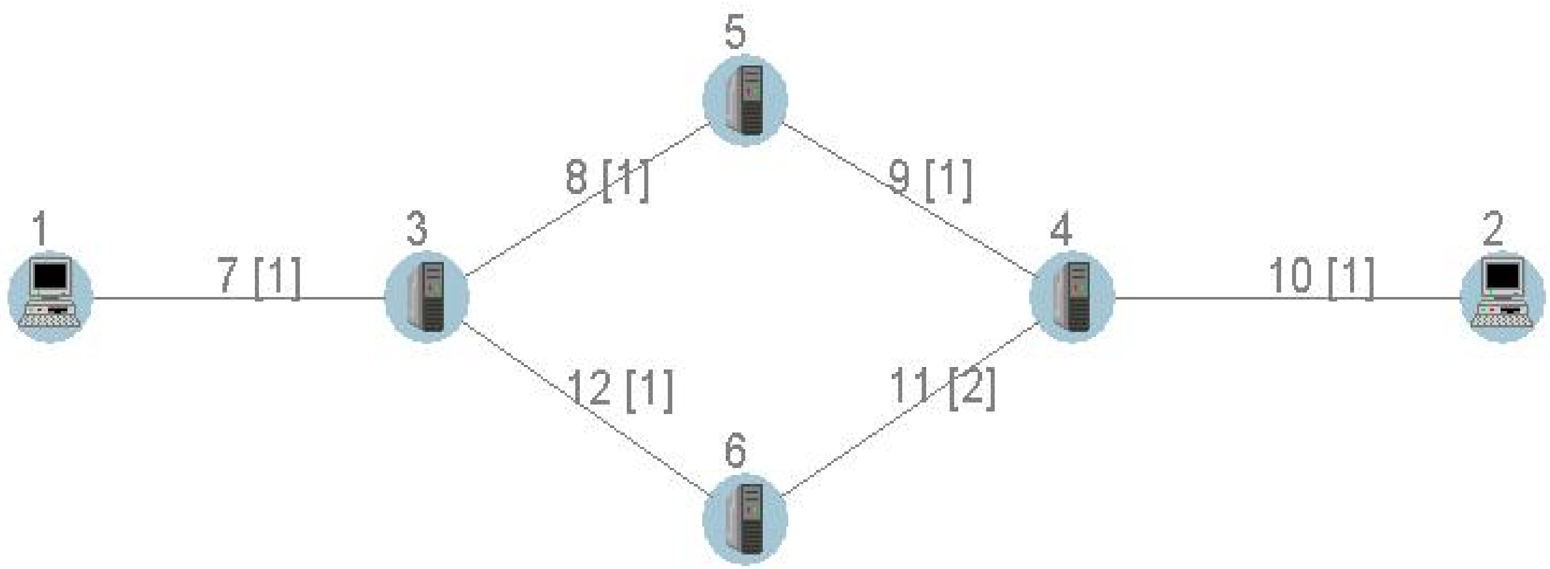
Distance Vector Routing Example: init



Router 3			Router 4			Router 5			Router 6		
dest	cost	gate	Dest	cost	gate	dest	cost	gate	dest	cost	gate
3	0	local	4	0	local	5	0	local	6	0	local
1	1	7	5	1	9	3	1	8	4	2	11
5	1	8	2	1	10	4	1	9	3	1	12
6	1	12	6	2	11	-	-	-	-	-	-

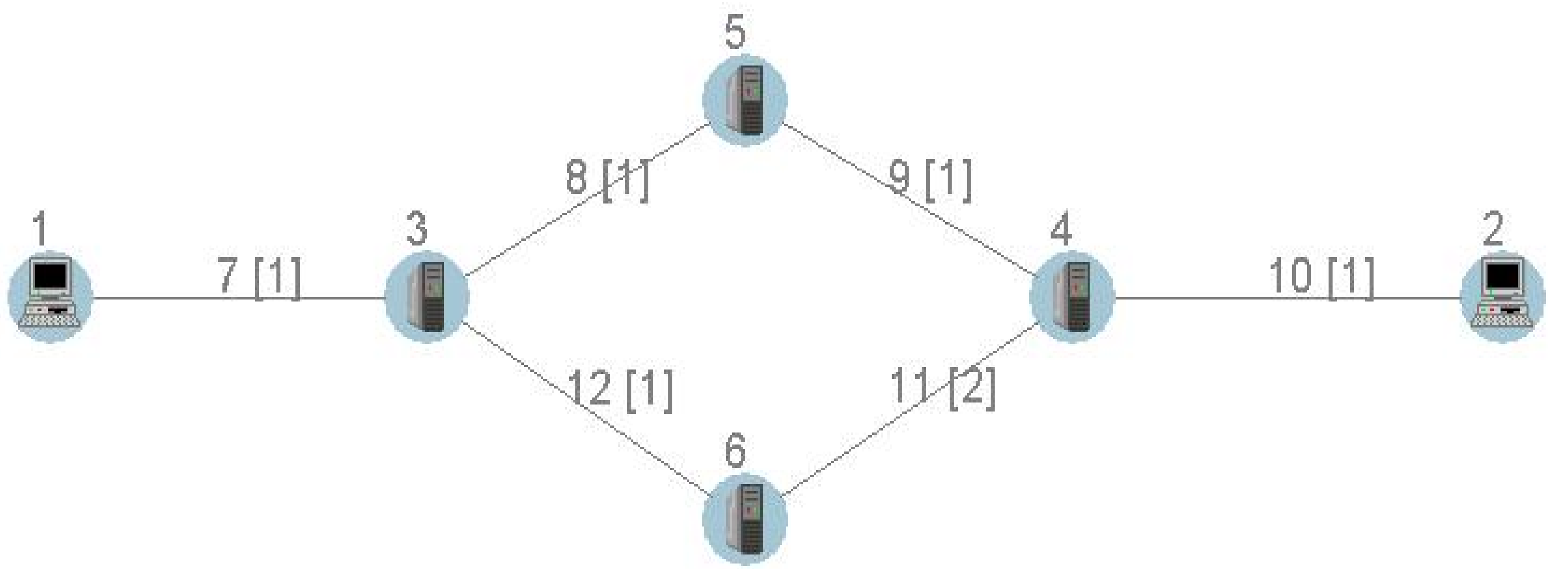
Generated with IRVTool: http://heim.ifi.uio.no/michawe/research/tools/irvtool/

Distance Vector Routing Example: Iteration 1



Router 3			Router 4			Router 5			Router 6		
dest	cost	gate	dest	cost	gate	dest	cost	gate	dest	cost	gate
3	0	local	4	0	local	5	0	local	6	0	local
1	1	7	5	1	9	3	1	8	4	2	11
5	1	8	2	1	10	4	1	9	3	1	12
6	1	12	6	2	11	1	2	8	5	2	12
4	2	8	3	2	9	6	2	8	2	3	11
-	-	-	-	-	-	2	2	9	1	2	12

Distance Vector Routing Example: Iteration 2



Router 3			Router 4			Router 5			Router 6		
dest	cost	gate	dest	cost	gate	dest	cost	gate	dest	cost	gate
3	0	local	4	0	local	5	0	local	6	0	local
1	1	7	5	1	9	3	1	8	4	2	11
5	1	8	2	1	10	4	1	9	3	1	12
6	1	12	6	2	11	1	2	8	5	2	12
4	2	8	3	2	9	6	2	8	2	3	11
2	3	8	1	3	9	2	2	9	1	2	12

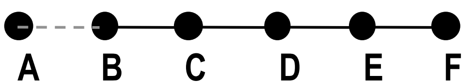
(Distance Vector) Routing Problems

- No globally known point of convergence
  - convergence can take long
- Good news overwrites bad news
  - black hole detection: packets go to /dev/null
  - e.g.: router with wrong table - all destinations cost 0 via local
- Reliance on updates
  - routing messages can be lost!
  - Requirement: timer per entry; typical timer problems again (long = more robust, but takes long to converge in case of link outage)
- Security
  - False updates can lead to trouble! Source must be authenticated

Distance Vector Routing Problems /2

Problem: good news spreads fast, bad news slowly
(count2infinity problem)

Example (cost=1 for all links for simplicity):



1. all tables are correct (e.g., F has cost 5 to A)
2. then, link A-B goes down
3. B recomputes with table initialized to "infinite" for A, receives (A,2) as part of msg. from C (outdated) → table-entry [A,C,3]
   ... but C routes via B! → bouncing effect
4. when C recomputes, it receives (A,3) from B, sets [A,B,4]
5. etc.etc., similar for other routers

course of action may be different (distributed/concurrent algorithm!)
but in any case "infinity" spreads very slowly

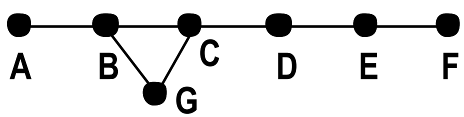
count2infinity occours only in total isolation:

e.g., extra node connected to A and F: high cost, but eventually reroute via F

Split Horizon

Solution: split horizon

- at item 3 above, C omits (A,2) from its message to B
  - More aggressive variant: split horizon with poisoned reverse: instead of omitting (A,2), transmit (A,∞) to B
  - but: problem basically remains if topology is, e.g. as follows:



- why: G tells B about route via G, G tells B about route via C, …
- Note: all these problems are caused by wrong updates
  - enhancement: triggered updates - send updates immediately when table changes
- Many more problems identified / solutions proposed, nothing „perfect“
  → Link state routing has become more common again

Link State Routing

- All routers have "full" information about net (at hierarchy level) [from, to, link, distance]
- each router can compute optimal routes locally
  - typically: Dijkstra's "Shortest Path First" algorithm
- more traffic for spreading out info about net, but no traffic for spreading out "optimum"
- no broadcast of optimum → info more up-to-date, "nearby" changes known/reflected fast: these are most important!
- less scalable than distance vector routing, may generate lots of traffic: hierarchy!

Updating the Distributed Map

- Flooding upon change (link going down, ..) - messages contain:
  - single update line (from, to, link, distance)
  - timestamp or message number N to distinguish between old and new information
  - number: updated slowly (link transition or timer); danger: wraparound!

Algorithm:

```
look for msg_table.line in my_table

if (line not present)
    add line;
    broadcast (everywhere except back to sender);
else if (my_table.N < msg_table.N)
    replace my_table.line by msg_table.line;
    broadcast (everywhere except back to sender);
else if (my_table.N > msg_table.N)
    transmit my_table.line in a new message via incoming interface;

(both numbers equal: do nothing)
```

Dijsktra's Algorithm

```
Initialization:
  N = {A}
  for all nodes v
    if v adjacent to A
      then D(v) = c(A,v)
      else D(v) = infty

Loop
  find w not in N such that D(w) is a minimum
  add w to N
  update D(v) for all v adjacent to w and not in N:
     D(v) = min( D(v), D(w) + c(w,v) )
  /* new cost to v is either old cost to v or known
   shortest path cost to w plus cost from w to v */
until all nodes in N
```

Notation:

c(i,j): link cost from node i to j. cost infinite if not direct neighbors

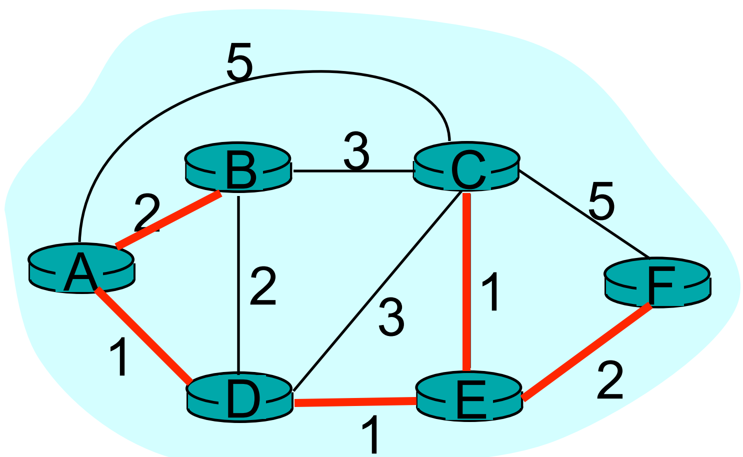
D(v): current value of cost of path from source to dest. V

p(v): predecessor node along path from source to v, that is next v

N: set of nodes whose least cost path definitively known

Dijkstra's algorithm: example

Step	start N	D(B),p(B)	D(C),p(C)	D(D),p(D)	D(E),p(E)	D(F),p(F)
0	A	2,A	5,A	1,A	infinity	infinity
1	AD	2,A	4,D		2,D	infinity
2	ADE	2,A	3,E			4,E
3	ADEB		3,E			4,E
4	ADEBC					4,E
5	ADEBCF					



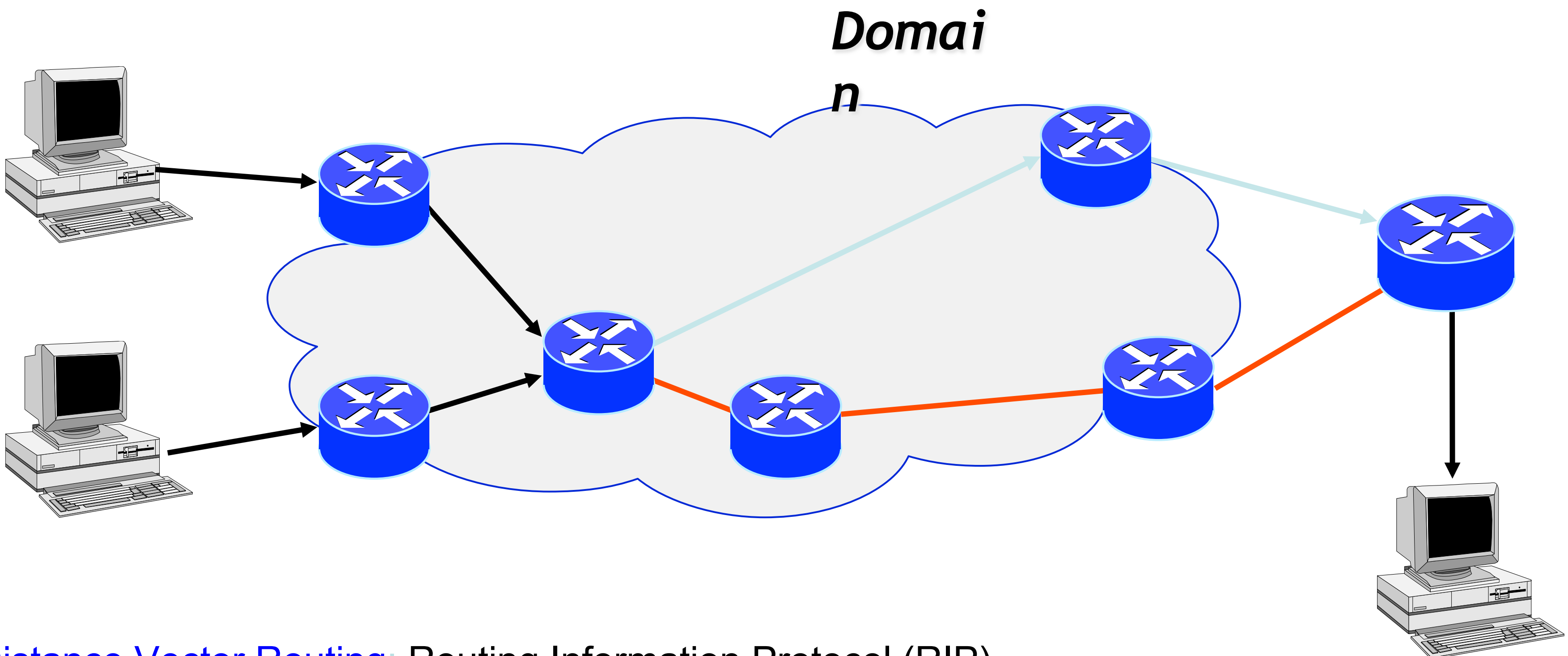
Link State Routing Conclusion

- Internet standard protocol: Open Shortest Path First (OSPF)
  - why? SPF is recommended, but not necessary for compatibility!
    - complexity: check all nodes w not in N → n*(n+1)/2 comparisons (n = number of nodes): $O(n^2)$; more efficient implementations possible: O(n log n)
  - OSPF designed in an open fashion

- Additional advantage of SPF: support of multiple paths
  - minor algorithm change, usually called Equal-Cost Multi-Path (ECMP)
  - idea: send traffic across several paths of equal length (cost)
  - better capacity utilization, less congestion (queuing delay)
  - but: out-of-order delivery, problem with RTT estimation (timeout control)

- Link State Routing also has issues - consider:
  - network split in two parts by link going down
    → two networks with two different maps
  - link up again
    → single link update insufficient, full table update inefficient
    (“bringing up adjacencies“)

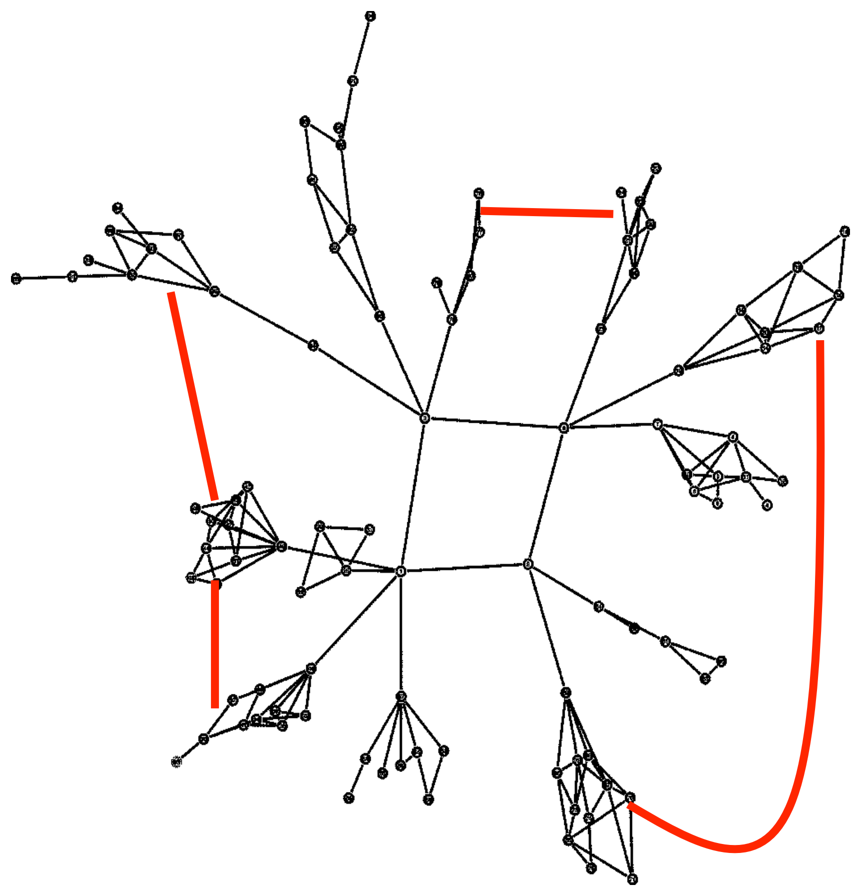
IP Routing: Domain level (IGP's)



- Distance Vector Routing: Routing Information Protocol (RIP), now RIPv2 - several features (authentication, ..), Interior Gateway Routing Protocol (IGRP) - Cisco proprietary, uses TCP instead of UDP
- Link State Routing: Open Shortest Path First (OSPF), consists of 3 subprotocols: Hello, Exchange, Flooding.
- OSI: „Intra-Domain Intermediate System to Intermediate System Routeing Protocol“ (IS-IS) - link state routing protocol similar to OSPF

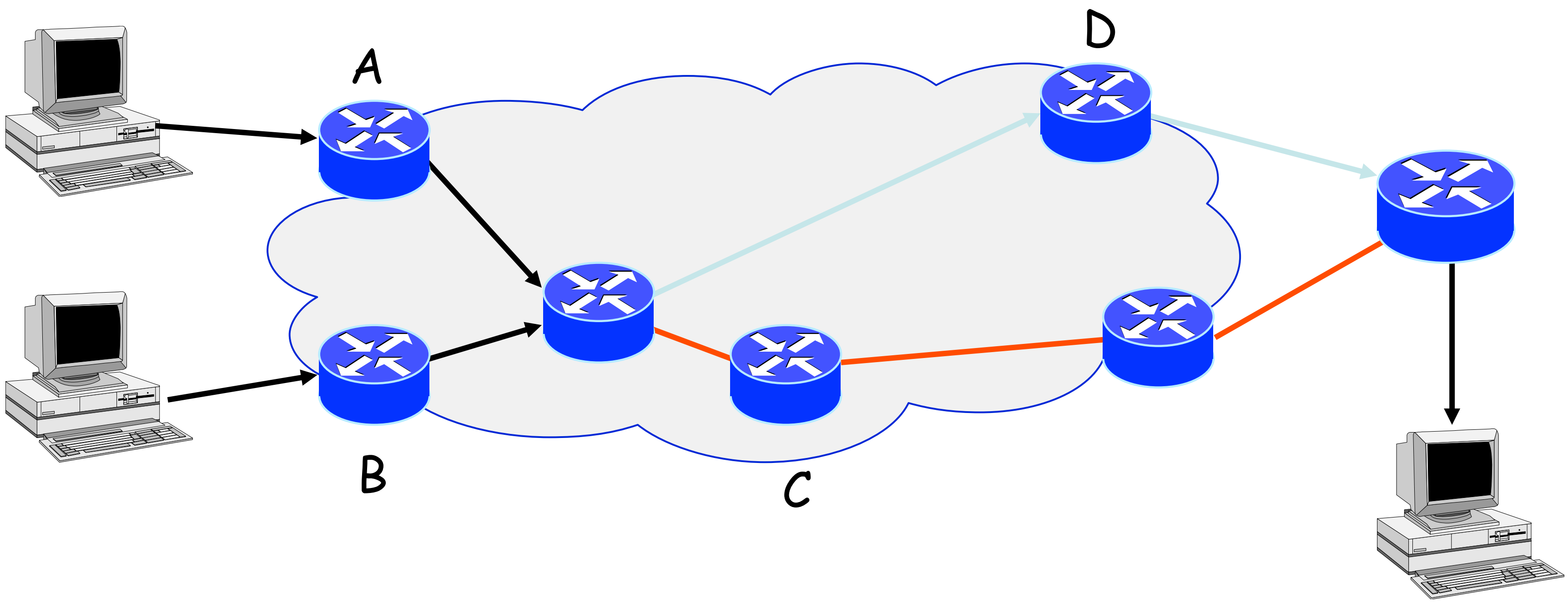
Interdomain routing (EGP's)

- Border Gateway Protocol (BGP):
  Roughly: Distance Vector Routing between "Autonomos Systems" (registered unique AS number)
  - Actually: Path Vector
- Metrics: often configured manually according to costs
- Peering relationships:
  usually cheaper than the backbone
- Implementation: filter
  only accept routes from specific sources



Traffic Engineering

- Static configuration: administrators want to move some traffic
- based on long-term measurements



- IP-in-IP tunneling example:
- B encapsulates packets (new src=B, dst=C), C removes new header

From traffic engineering to MPLS

- Layer violation:
  - If you are tunneling along a fixed path and your network is ATM, you could just as well set up a VC for the path - faster forwarding!
- Automatic variant: Ipsilon IP Switching
  - Switches identify data flow, establish ATM-VC "Short-Cut"
  - Does not scale well - fine granularity
- Better: Multiprotocol Label Switching (MPLS)
  - Not just (but mostly) ATM, even LANs!
  - based upon separation of forwarding and control functionality in routers
  - Label: put short info. (from layer 2) in front of IP (like IP encapsulation)
  - Label Switching Routers (LSR) forward on Label Switched Path (LSP)
  - At destination: remove label, forward IP packet normally

Multi Protocol Label Switching (MPLS)

- Forwarding Equivalence Class (FEC)
  - Group of packets with similar expected treatment (usually: same label)
  - Various forms of classification (choose *which* data flow?) possible

- What if labeled packets are labeled again?
  - Labels are stacked (push, pop, swap [= pop+push, connects two LSPs])

20 bit	Label	Label	Label	IP Header	Data
3 bit	Experi-mental	Experi-mental	Experi-mental		
1 bit	S=0	S=0	S=1		
8 bit	TTL	TTL	TTL		

Stack top — Stack bottom

MPLS details

- Label designed for speed:
  - 32 bit
  - S=1: “this is the last label“
  - TTL is the only IP header field that MUST be treated at each hop
- Labels distributed via Label Distribution Protocol (LDP)
- MPLS applications:
  - Traffic engineering (like IP-in-IP tunneling)
  - Split load by establishing more LSPs for one FEC
  - Virtual Private Networks (VPNs)
  - First aid if links go down (switch to different LSP)
  - QoS support

Multi Protocol Lambda Switching (MPλS)

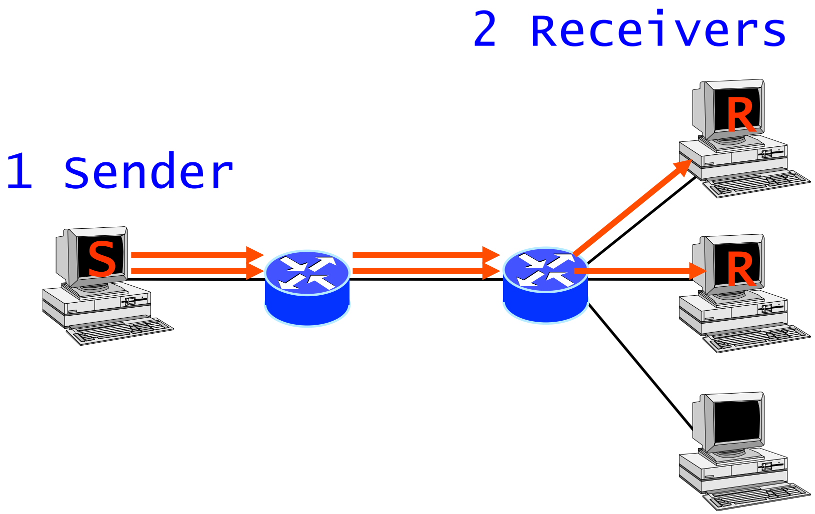
- Wavelength Division Multiplexing (WDM)
  - frequency multiplexing for optical tranmission media
  - optical packet switches - switches based on colours
  - problems: contention (signals with same colour overlap), ...

- IP over WDM: difficult to realize
  easier if connection oriented

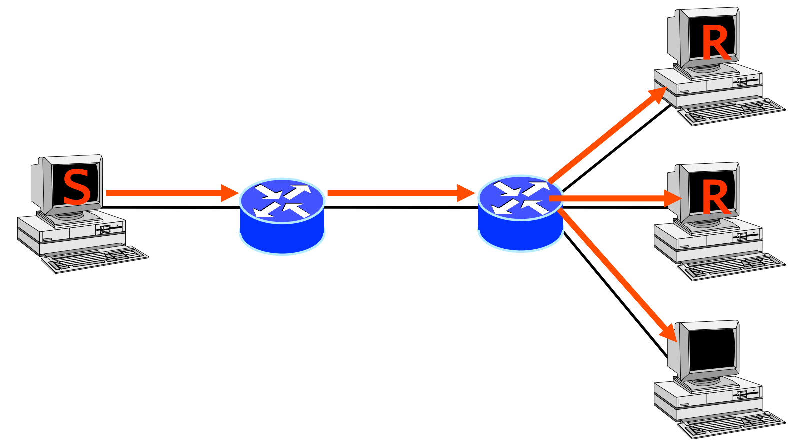
- Achieved via MPLS (Wavelength = label) -> MPλS

- MPλS switches can be connected via λ's for default-routing and signaling (similar to MPLS <=> ATM VC)

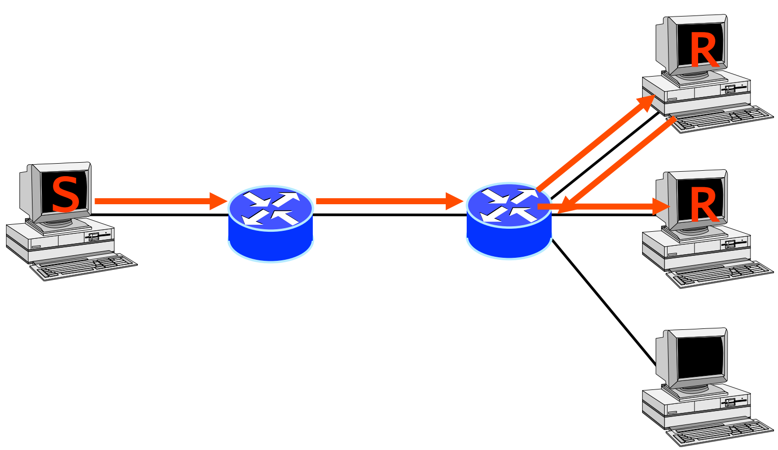
Unicast / Broadcast / (overlay) Multicast



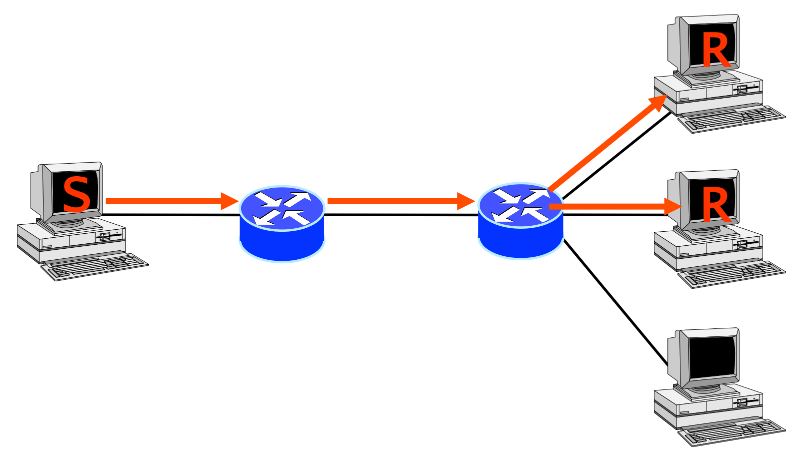
Unicast



Broadcast



MBone



IP Multicast

IP Multicast

- Required for applications with multiple receivers only
  - video conferences, real-time data stream transmission, ..

- Interested receivers + relevant routers regarded as a spanning tree
  - packet distribution algorithm called multicast routing

- Issues:
  - group management
    - protocol required to join / leave group dynamically: Internet Group Management Protocol (IGMP)
    - state in routers: hard / soft (lost unless refreshed)?
    - who initiates / controls group membership?
  - error control (congestion control, flow control) problematic: ACK implosion!
  - Internet: inter-domain and intra-domain protocols necessary
  - address allocation, requirement of router support, modest business motivation (customer / provider / ISP roles): deployment cumbersome

Mobility

- Vision: anywhere, anytime (Internet) access

- Possibility: obtain address dynamically (“plug-and-play“)
  - addresses hard-coded by system admin in a file
  - obtained via Dynamic Host Configuration Protocol (DHCP):
    - host broadcasts “DHCP discover” msg, DHCP server responds with “DHCP offer” msg, host requests IP address: “DHCP request” msg, DHCP server sends address: “DHCP ack” msg

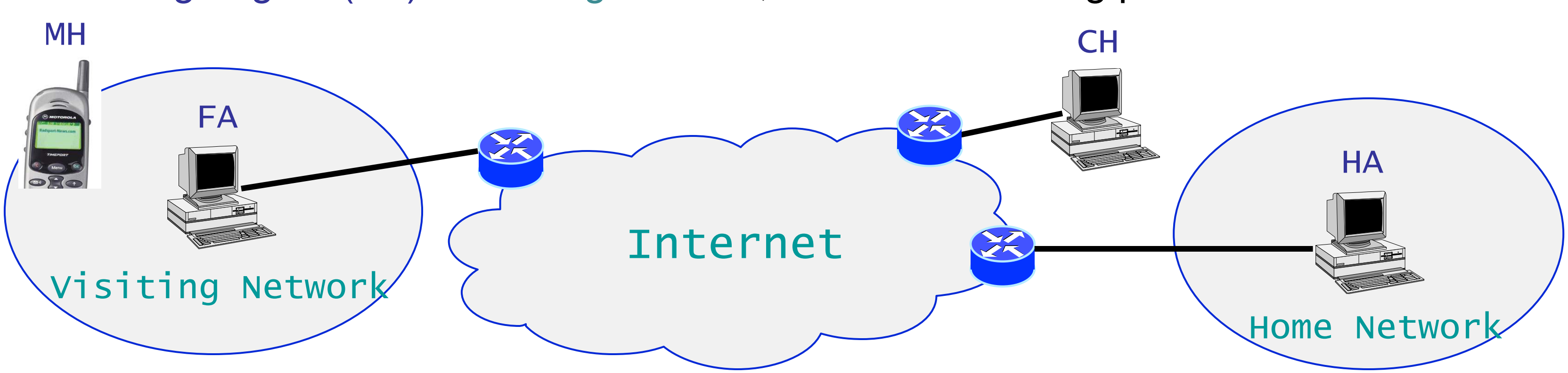
  - DHCP, DNS etc. perhaps still too troublesome for emerging IP based networks: home, automobile and airplane areas, ubicomp
    → alternative: IETF ZEROCONF effort - goals:
    - Allocate addresses without a DHCP server (use MAC address, ser.no., ..)
    - Translate between names and IP addresses without a DNS server
    - Find services, like printers, without a directory server
    - Allocate IP Multicast addresses without a MADCAP server

Mobility /2

- IP address = host identifier AND location
  - address changes: identifier changes => e.g. TCP connections are interrupted
  - connections should persist when users move
    ...no problem within (W)LAN, but what if user moves from LAN 1 to LAN 2 ?
  - what if a user wants to run a server?

- Solution: Mobile IP
  - mobile host (MH): address does not change
  - corresponding host (CH): wants to contact MH
  - home agent (HA): represents MH when MH not in home network
  - foreign agent (FA): in visiting network, forwards incoming packets to MH

always knows location of HA !



Mobility /3

- Two relevant addresses per MH:
  - Home address: permanent MH address, belongs to home subnet
  - Care-of-address: used by MH in visiting network - two types:
    - Foreign-Agent-Care-of-Address: FA forwards incoming packets to MH; several MH's can share the same address (not used in IPv6)
    - Collocated-Care-of-Address: assigned to MH in visiting network - no FA! address must be different for each MH in visiting network

- Operation:
  - Agent Discovery: passive (agent advertisement msgs from HA / FA) or active (agent solicitations query for advertisements) detection of HA or FA
  - Registration: MH sends care-of-address to HA + registers (request-reply); HA keeps table of home address - care-of-address entries → can reach MH
  - Tunneling: data flow: CA → HA → FA → MH → CA

tunnel (HA → FA)

source = home address! (MH → CA)