Rechnernetze - Computer Networks

Lecture 8: Internetworking
Routing

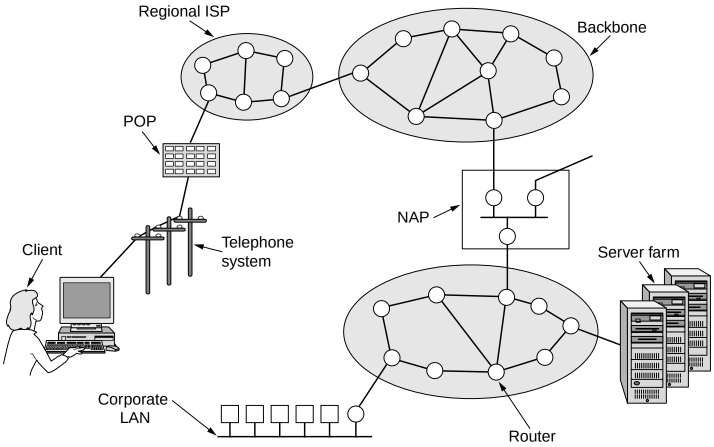
Prof. Dr.-Ing. Markus Fidler

Institute of Communications Technology
Leibniz Universität Hannover

June 7, 2024

Outline

End systems vs. intermediate systems



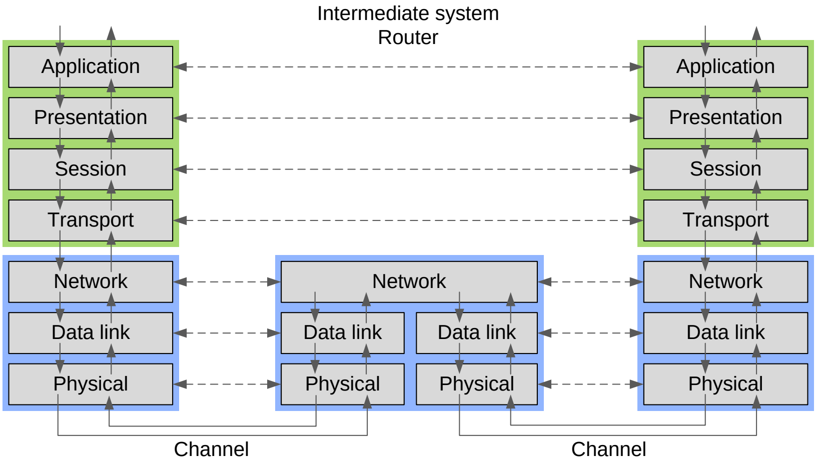
[Source: Tanenbaum, Computer Networks]

Network layer overview

Data transfer between end systems

- requires cooperation of several intermediate systems
- network layer (layer 3) functionality
- can involve several heterogeneous
  - links, i.e. physical and data link layer technologies
  - subnetworks, e.g. LANs, switches, bridges
- layered architecture: network layer
  - implemented on intermediate systems, i.e. routers
  - typically highest layer implemented on intermediate systems
    - network layer services are provided by the carrier
    - transport layer services and above are implemented only on end systems, i.e. under control of the user

Intermediate system: Router

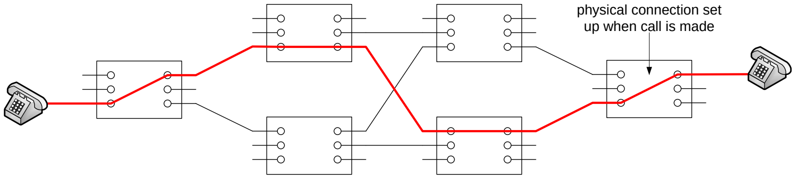


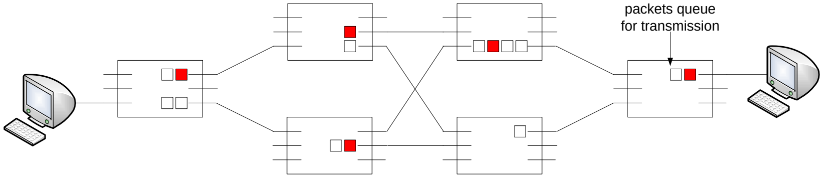
Network layer functions

Tasks of the network layer

- addressing
- internetworking
  - transitions between (sub-) networks
- routing
  - determine paths from source to destination through the net
- switching
  - forwarding of data/packets
- resource allocation
  - reservation of capacity
  - allocation of buffers
- multiplexing of data/packet flows
- fragmentation and reassembly
  - heterogeneous links with different maximum frame size

Circuit switching vs. packet switching





Circuit switching

Principle

- connection exists for the duration of communication
- switching centers establish end-to-end connections
- connections between switching centers, e.g. TDM, FDM

Example: Telephony

- operators at manual switching boards
- automated mechanical dialers
- digital networks (ISDN)

Properties

- connection has to be established before data transmission
- resource allocation during connection establishment
- rigid resource allocation
  - supports quality of service
  - possibly wastes resources

Packet switching

Principle

- packets of fixed or variable but limited size
- route selection on per-packet basis
- no dedicated path from source to destination

Example: Internet

- routers perform packet switching

Properties

- no reserved resources
- possibility of congestion
- no (intrinsic) support of quality of service
- efficient, high resource utilization

Virtual circuit switching

Principle

- dedicated path, i.e. virtual circuit (VC), from source to destination for the duration of the transmission
- intermediate systems store state information for each VC
- virtual circuits are, however, not physical connections
- data are forwarded as packets along the dedicated VC

Examples

- Asynchronous Transfer Mode (ATM)
- Multi-protocol Label Switching (MPLS)
- Integrated Services, Resource Reservation Protocol (RSVP)

Properties

- all packets use the same path, in-sequence delivery
- efficient resource utilization, no resource waste
- can support resource reservation and quality of service

Network layer services

Major design decision

- connection-oriented, reliable vs. connectionless, unreliable
- recall network layer is implemented by intermediate systems

Connection-oriented network layer: Typically

- connection establishment/release before/after data transfer
- error control, capacity guarantees, quality of service
- favorable for real-time communications, e.g., telephony, video
- used by telephone companies

Connectionless network layer: Typically

- immediate data transfer possible, no connection handling
- unreliable, no error control (left to higher layers)
- favorable for short data exchanges
- used by the Internet community

Connection-oriented vs. connectionless

Connection-oriented network layer: Characteristics and pros

- can provide sophisticated network layer services
- simplifies the design of higher layers
- places intelligence into the network, allows dumb end systems

Connectionless network layer: Characteristics and pros

- keeps the network simple and flexible
  - network layer service typically is best-effort, i.e., no guarantees
  - higher layers at the end systems have to recover from errors
  - dumb network requires intelligent end systems
- scalable
  - intermediate systems do not have to handle lots of connections
  - instead transport layer connections maintained by end systems

Implementation options

Network layer (layer 3)

- connection-oriented service
  - circuit switching (continuous data stream)
  - virtual circuit switching (uses packets)
- connectionless service
  - packet switching (also datagram service)

Transport layer (layer 4)

- in case of connection-oriented network layer
  - little functionality required at end-systems since the network typically provides essential services
- in case of connectionless network layer
  - most functionality beyond routing and switching shifted to the end systems that enhance the best-effort network service

Note the difference between connection-oriented

- network layer: involves all routers, i.e., the network itself
- transport layer: involves only hosts, i.e., end-systems

Outline

Overview

Routing
- Dijkstra's algorithm
- Link state routing
- Internetworking
- Open shortest path first (OSPF)
- Distance vector routing
- Routing information protocol (RIP)
- Border gateway protocol (BGP)

Routing basics

Task: basic network layer functionality

- find the best route from a source to a destination across a network of routers
- determine which outgoing interface to use at each router in order to reach the destination
- implemented by routing algorithms

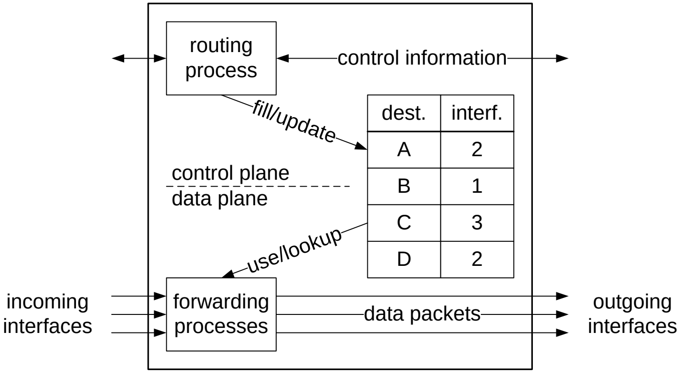
Route determination

- packet switching, datagram service
  - individual decision on a per-packet basis
- circuit switching, virtual circuit switching
  - joint decision for all data (packets) of the same flow
  - the route is determined during connection establishment

Routing and forwarding

Differentiate between routing and forwarding

- control plane: routing to determine paths to destinations
- data plane: process and forward received packets



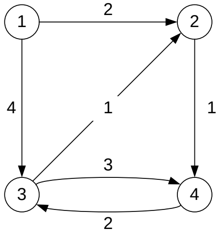
dest.	interf.
A	2
B	1
C	3
D	2

Network graph

Approach: weighted network graph $G = \langle V, E, C \rangle$

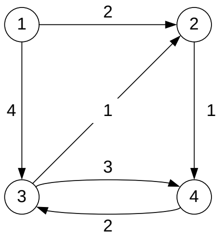
- $V =$ vector of vertices, i.e., routers v_i where $i = 1 \dots n$
- $E =$ matrix of edges, i.e., links where
  - $e_{i,j} = 1$: a directed link from node i to node j exists
  - $e_{i,j} = 0$: otherwise
- $C =$ (non-negative) costs associated with links, e.g.,
  - link delay
  - 1/link capacity
  - cost of usage

$$E = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 0 & 2 & 4 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$



Reachability

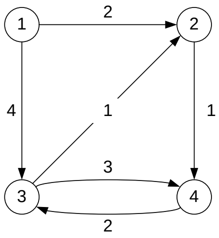
$$E = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



- E denotes the nodes that are reachable in one step

Reachability

$$E = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

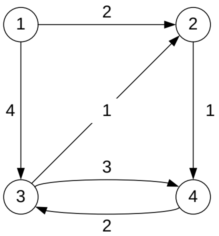


- E denotes the nodes that are reachable in one step
- $E \cdot E$ denotes the number of two step paths
- $E \cdot E \cdot E$ denotes the number of three step paths ...

$$E^2 = \begin{pmatrix} 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}, \quad E^3 = \begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}, \quad E^4 = \begin{pmatrix} 0 & 2 & 1 & 2 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 0 & 1 & 1 & 1 \end{pmatrix}$$

Reachability continued

$$E = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$



Given N nodes: node j is unreachable from node i if the respective entry (i,j) in $\sum_{n=1}^{N} E^n$ is zero

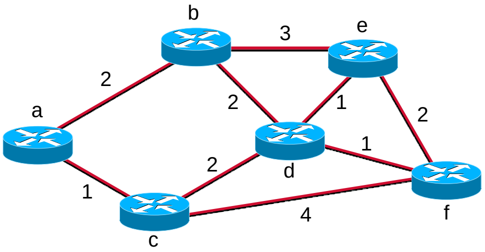
$$E^1 + E^2 + E^3 + E^4 = \begin{pmatrix} 0 & 4 & 4 & 5 \\ 0 & 1 & 2 & 3 \\ 0 & 3 & 3 & 5 \\ 0 & 2 & 3 & 3 \end{pmatrix}$$

i.e., node 1 is not reachable from any other node.

Routing

Routing determines the best path between the two nodes:

- least-cost path: path with the smallest cost among all paths
- shortest path: path with the smallest number of hops



Dijkstra's algorithm is an implementation, it

- computes least-cost paths from a source to all destinations
- list of least-cost paths is the routing table
- iterative greedy algorithm: after k steps least-cost paths to at least k nearest nodes are known

Notation

Denote

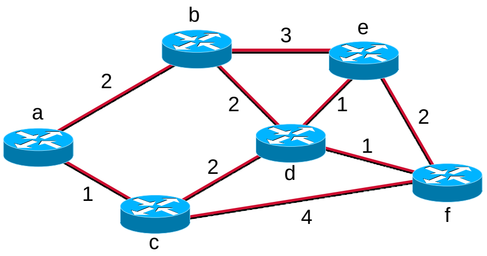
- $c(i,j)$ the cost of link $e_{i,j}$ from node i to node j
  - $c(i,j) = \infty$ if link $e_{i,j}$ does not exist
- $d(i)$ cost (distance) from the source to node i
- $p(i)$ predecessor of node i on the least-cost path from the source to node i
- M set of nodes for which least-cost paths are uniquely determined

Dijkstra's algorithm

```
1: $M = \{a\}$
2: **for** all nodes $i$ **do**
3:     **if** link $e_{a,i}$ exists **then**
4:         $d(i) = c(a,i)$
5:     **else**
6:         $d(i) = \infty$
7:     **end if**
8: **end for**
9: **repeat**
10:     find a node $j$ not in $M$ such that $d(j)$ is minimal
11:     add $j$ to $M$
12:     **for** all nodes $k$ not in $M$ **do**
13:         **if** $d(k) > d(j) + c(j,k)$ **then**
14:             $d(k) = d(j) + c(j,k)$
15:             $p(k) = j$
16:         **end if**
17:     **end for**
18: **until** all nodes in $M$
```

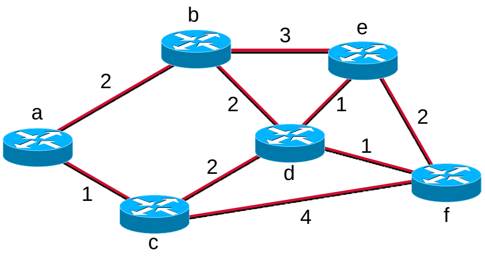
Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)



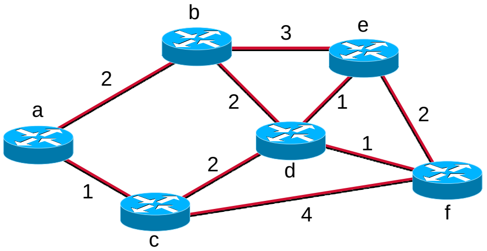
Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)
a	2	a	1	a	∞	-	∞	-	∞	-



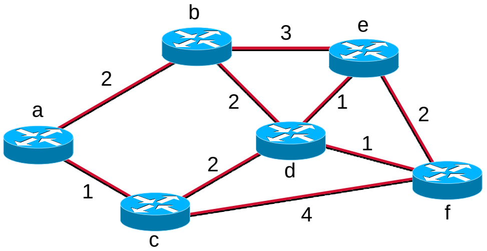
Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)
a	2	a	**1**	**a**	∞	-	∞	-	∞	-
ac	2	a			3	c	∞	-	5	c



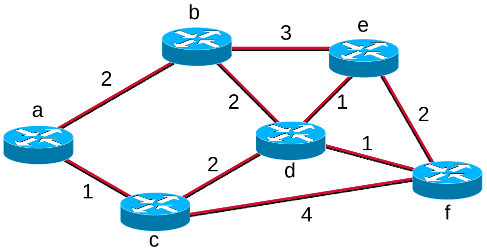
Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)
a	2	a	**1**	**a**	∞	-	∞	-	∞	-
ac	**2**	**a**			3	c	∞	-	5	c
acb					3	c	5	b	5	c



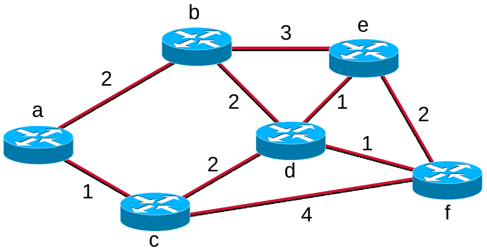
Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)
a	2	a	**1**	**a**	∞	-	∞	-	∞	-
ac	**2**	**a**			3	c	∞	-	5	c
acb					**3**	**c**	5	b	5	c
acbd							4	d	4	d



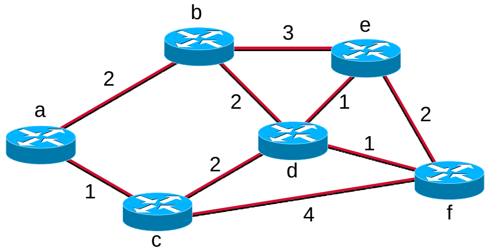
Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)
a	2	a	**1**	**a**	∞	-	∞	-	∞	-
ac	**2**	**a**			3	c	∞	-	5	c
acb					**3**	**c**	5	b	5	c
acbd							**4**	**d**	4	d
acbde									4	d



Dijkstra's algorithm: example

M	d(b)	p(b)	d(c)	p(c)	d(d)	p(d)	d(e)	p(e)	d(f)	p(f)
a	2	a	**1**	**a**	∞	-	∞	-	∞	-
ac	**2**	**a**			3	c	∞	-	5	c
acb					**3**	**c**	5	b	5	c
acbd							**4**	**d**	4	d
acbde									**4**	**d**
acbdef										



Dijkstra's algorithm: result

After executing Dijkstra's algorithm we know for each destination

- the cost of the least-cost path to the destination
- all nodes on the least-cost path to the destination

Complexity of the algorithm, given N nodes

- step 1: check $N-1$ destinations
- step 2: check $N-2$ destinations
- $\dots$
- step $N-1$: check 1 destination
- overall: $(N-1)+(N-2)+\dots+1 = N(N-1)/2$ steps
- complexity: $\mathcal{O}(N^2)$

Remark: there exist better implementations of the algorithm in $\mathcal{O}(N\log(N))$

Outline

Routing protocols

Routing protocols acquire information about the network graph

- centralized: routing control center distributes routing tables
- distributed: each router generates its own routing tables

- global information: each router knows the entire matrix of links, i.e., link state routing using Dijkstra's algorithm
- local information: routers exchange information with their neighbors only, i.e., distance vector routing

- static link weights: static routing
- dynamic link weights: adaptive routing
  link weights can change quickly, e.g., due to link failures etc.
  - load-dependent: can cause oscillations
  - load-independent: typically used today

  need measurements to estimate current link weights

Link state routing

Basic principle

- routers measure the cost metric to direct neighbors, distribute information, and calculate optimal routes

Procedure (executed periodically)

1. determine the addresses of adjacent routers
2. measure the cost metric to neighbor routers
3. organize the local link state information in a packet
4. distribute the information to all routers
5. calculate least-cost paths using all received link state packets

Examples

- IS-IS (Intermediate System to Intermediate System)
- OSPF (Open Shortest Path First), RFC 1247, RFC 2328

Link state routing continued

Step 1

- gather information about directly adjacent routers
- routers send HELLO messages over each connected link
- adjacent routers respond with their unique router id

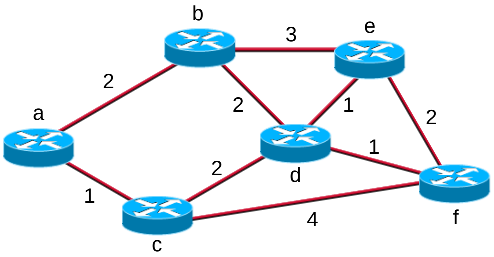
Step 2

- measure/estimate the cost metric of connected links
- typically the delay of the link is used as cost metric
- measured by transmission of ECHO messages that are reflected by the receiver
- multiple transmissions improve average values
- payload data can be used to measure transmission delays

Link state routing continued

Step 3

- organize the information as link state packet
- packet includes
  - own address
  - sequence number
  - age
  - distances to neighbors



a	
seqno	
age	
b	2
c	1

b	
seqno	
age	
a	2
d	2
e	3

...

Link state routing continued

Step 4

- distribute the local information to all routers
- broadcast link state packets, aka. flooding
  - sequence number in packets
  - age counter in packets
- link state packets time out and are invalidated after age sec.

Step 5

- compute new routes
- each router for itself
- uses Dijkstra's algorithm

Flooding

Flooding principle

- router transmits received link state packets to all adjacent routers except over the path it came in
- must avoid broadcast storm (infinitely many packets)

Methods to circumvent broadcast storm

- packets are transmitted over a limited number of hops only
  - hop counter in the packet header (here age)
  - each router decrements the hop counter
  - routers discard packets with hop counter $= 0$
  - need to know the maximum path length, respectively, maximum diameter of the network
- duplicated packets are filtered and discarded
  - packets are labeled with source id and sequence number
  - routers remember packets that have already been transferred
  - routers discard duplicated packets
  - routers need to keep track of sequence numbers of each other

Outline

Overview

Routing
- Dijkstra's algorithm
- Link state routing
- Internetworking
- Open shortest path first (OSPF)
- Distance vector routing
- Routing information protocol (RIP)
- Border gateway protocol (BGP)

Internetworking

So far

- routing for entire networks including all routers and links
- all routers are treated uniformly
- the network is flat, i.e. non-hierarchical

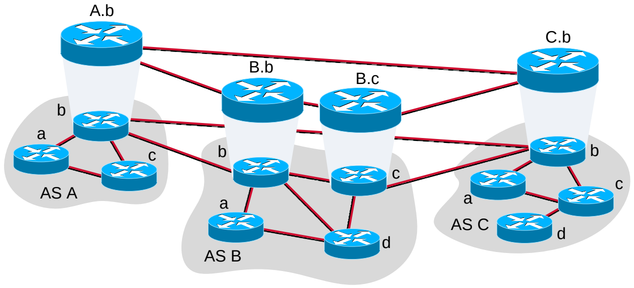
The Internet is a network of networks

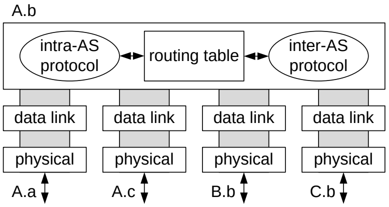
- many administratively independent regions called autonomous systems (ASs)
- managing/using global routing tables is not possible due to
  - different administrative authorities
  - the immense number of hosts and routes

Solution

- form a hierarchy of networks
- routing is performed
  - between networks
  - within networks

Autonomous systems: example





Hierarchical routing

Routing within ASs (interior gateway protocols)

- routers within an AS exchange routing information only locally within the AS
- routers use an intra-AS routing protocol, e.g.,
  - routing information protocol (RIP)
  - open shortest path first (OSPF)
  - enhanced interior gateway routing protocol (EIGRP) (proprietary protocol by CISCO)
- different ASs can run different intra-AS routing protocols

Routing between ASs (exterior gateway protocols)

- ASs maintain one or more gateway or border routers
- border routers perform inter-AS routing using the
  - border gateway protocol (BGP)

Outline

Open shortest path first (OSPF)

Specification

- Version 2: RFC 2328, April 1998
    - 'open' as opposed to proprietary solutions e.g. by Cisco

Characteristics

- based on link state routing
- OSPF advertisements are sent as broadcasts in the entire AS
- each router has full topological information of the AS
- shortest paths are computed using Dijkstra's algorithm
- link costs can be configured manually in routers

Broadcast of OSPF advertisements

- transmitted directly over IP
- periodically every 30 minutes or in case of changes
- each advertisement has one entry for each neighboring router

OSPF features

Some features

- security
  - authentication of messages
  - trusted routers
- load balancing if several paths have equal cost
- link costs can be set individually for different types of service (TOS field in the IP header)
  - different routes support different quality of service
- supports intra-AS hierarchies of routers
- extension supports multicast (MOSPF)

OSPF hierarchical routing

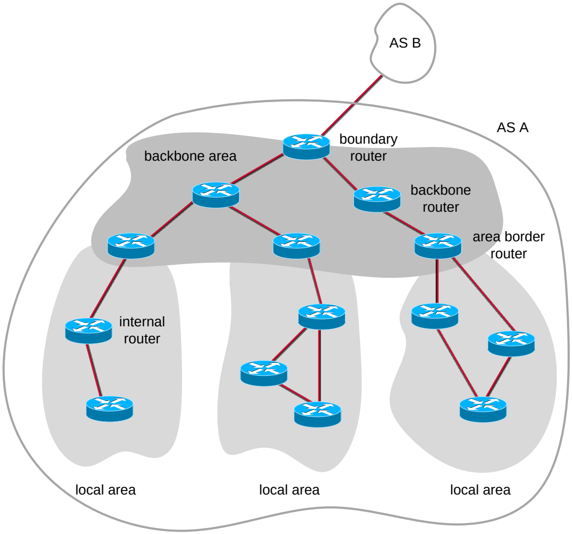
Two-level hierarchy within an AS

- several local areas
- one backbone area
  - forwards IP datagrams between local areas
  - forwards IP datagrams from and to other ASs
- OSPF advertisements are broadcast only within an area
- each router knows the entire topology of its area

Depending on area and function routers are one out of four

- **internal router** within a local area
- **area border router** within both local and backbone area
  - compose routing information about networks of its area
  - transmit advertisements for other area border routers
- **backbone router** within the backbone area
- **boundary router** within the backbone area, connects to other ASs

OSPF hierarchical routing continued



Outline

Overview of distance vector routing

Characteristics

- each node determines paths and costs to all other nodes
- decentralized approach, routing information is distributed
- each node communicates only with its direct neighbors
    - nodes notify only their direct neighbors about changes
    - asynchronous, uncoordinated information exchange at any time
    - information are exchanged only if changes occur

Approach

- each node maintains two tables
    - distance table: specifies the cost of reaching any node x via any neighboring node y
    - forwarding table: minimal cost to reach a destination node x and next hop of the path y

Notation

Denote

- D^x distance table of node x
  - example: $D^x(y,z) = 4$ route to destination y has cost 4 via next hop z
- F^x forwarding table of node x
  - example: $F^x(y) = (4, z)$ shortest path to destination y via next hop z has cost 4

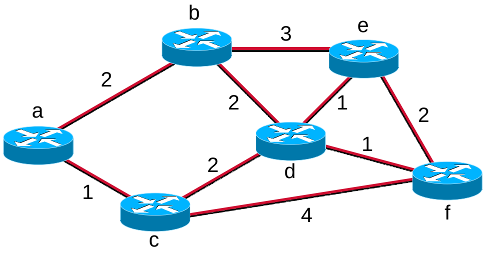
Mode of operation

- node z computes the shortest path to destination y as $c(z,y) = \min_w\{D^z(y,w)\}$
- node z stores the next hop w that achieves the minimum in its forwarding table $F^z(y) = (c(z,y), w)$
- node z sends its forwarding table to it's neighboring node x
- node x computes $D^x(y,z) = c(x,z) + c(z,y)$

Distance and forwarding tables: example

D^c	a	d	f
a	1	5	8
b	3	4	7
d	4	2	5
e	5	3	6
f	5	3	4

F^c	
a	1,a
b	3,a
d	2,d
e	3,d
f	3,d



Distance and forwarding tables: updates

Each iteration at a router is caused by

- changes of costs of connected links
- changes in the forwarding table of direct neighbors

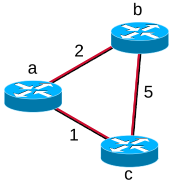
Two modes for exchanging forwarding tables between neighboring nodes exist

- fetching forwarding tables
  - selectively only from direct neighbors
  - only if costs on the path to the direct neighbor change
- sending forwarding tables
  - after each (complete) computation that changes the forwarding table
  - transmission of the forwarding table to all direct neighbors

Distance vector algorithm

```
1:  for all neighbors $y$ do
2:      $D^x(y, y) = c(x, y)$
3:  end for
4:  for all destinations $z$ do
5:      send $\min_w\{D^x(z, w)\}$ to all neighbors $y$
6:  end for
7:  loop
8:      if $c(x, y)$ for neighbor $y$ changes by $d$ then
9:          update $D^x(z, y) = D^x(z, y) + d$ for all destinations $z$
10:     else if updated $c(y, z)$ received from neighbor $y$ then
11:         recompute $D^x(z, y) = c(x, y) + c(y, z)$
12:     end if
13:     if a new $\min_w\{D^x(z, w)\}$ exists for any destination $z$ then
14:         send new $\min_w\{D^x(z, w)\}$ to all neighbors $y$
15:     end if
16: end loop
```

Distance vector algorithm: example



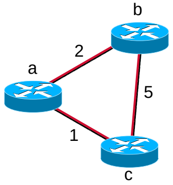
D^a	b	c
b	**2**	∞
c	∞	**1**

D^b	a	c
a	**2**	∞
c	∞	**5**

D^c	a	b
a	**1**	∞
b	∞	**5**

initialization

Distance vector algorithm: example



D^a	b	c
b	**2**	∞
c	∞	**1**

D^b	a	c
a	**2**	∞
c	∞	**5**

D^c	a	b
a	**1**	∞
b	∞	**5**

initialization

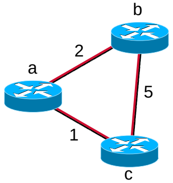
D^a	b	c
b	**2**	6
c	7	**1**

D^b	a	c
a	**2**	6
c	**3**	5

D^c	a	b
a	**1**	7
b	**3**	5

1. step

Distance vector algorithm: example



D^a	b	c
b	**2**	∞
c	∞	**1**

D^b	a	c
a	**2**	∞
c	∞	**5**

D^c	a	b
a	**1**	∞
b	∞	**5**

initialization

D^a	b	c
b	**2**	6
c	7	**1**

D^b	a	c
a	**2**	6
c	**3**	5

D^c	a	b
a	**1**	7
b	**3**	5

1. step

D^a	b	c
b	**2**	4
c	5	**1**

D^b	a	c
a	**2**	6
c	**3**	5

D^c	a	b
a	**1**	7
b	**3**	5

2. step

Routing information protocol (RIP)

Specification

- Version 1: RFC 1058, June 1988
  - experimental implementations in BSD UNIX already in 1982
  - implemented as `routed`
- Version 2: RFC 2453, November 1998, Internet standard 56

Characteristics

- based on distance vector routing
- metric is hop count, i.e. all links have cost 1
- maximum distance is 15 hops, i.e. paths within an AS span at most 14 routers

Exchange of distance vectors resp. RIP advertisements

- uses UDP (RIP is an application layer protocol)
- periodically every 30 seconds
- contain at most 25 networks (subnets) within the AS

RIP fault handling

Fault detection and recovery

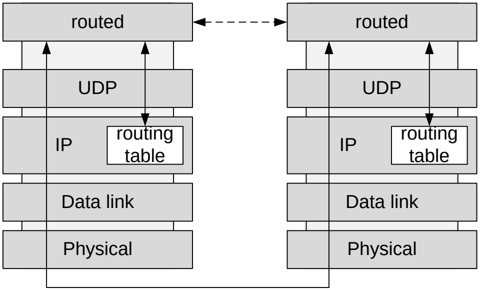
- at each router RIP maintains a timer for each neighboring router
- timeout interval is 180 seconds
- the timer is reset each time a RIP advertisement is received
- RIP advertisements are sent every 30 seconds
- if the timer expires 6 RIP advertisements in a row are missing; the neighbor is declared unreachable (link or router failure)
    - the router updates its routing table and sends it immediately to each of his neighbors
    - each neighbor updates its table and sends it immediately to its neighbors etc.

$\Rightarrow$ failures propagate quickly through the network

RIP implementation

Implementation of RIP as application layer protocol

- UNIX application process `routed`
- exchange of RIP advertisements using datagram sockets, i.e. UDP (port 520)
- `routed` is, however, a special process that can access and manipulate the local routing tables of a router



Border gateway protocol (BGP)

Specification

- Version 4: RFC 4271
- additionally RFC 1772 application and RFC 1773 experience

Characteristics

- de-facto standard for inter-AS routing in the Internet
- external BGP routing ends at the boundary router of the target AS
  - within the AS routing is done separately by intra-AS routing
  - different ASs can use different intra-AS routing protocols
- internal BGP is used to advertise paths to subnets in other ASs within an AS

BGP implementation

Implementation

- path vector protocol similar to distance vector
  - communication only with direct neighbors
  - exchange of complete paths to destinations (instead of next hop and hop count to destination)
- BGP advertisements
  - destination networks in CIDR notation
  - path of AS numbers (ASNs) to the destination
  - example path vector from X to Y: { X, AS1, AS2, ... , Y }
- BGP messages are transmitted using TCP

BGP operation

Basic operation

- reception and evaluation of advertisements from neighboring routers
    - discard paths via own AS to avoid loops
- computation and selection of suitable routes
    - consideration of policies and peering agreements
      Whom do I want to forward my traffic?
    - choice of the best path
    - update of own routing table
- sending BGP advertisements to direct neighbors
    - advertise only selected routes to selected neighbors
      Whose traffic do I want to forward?
    - controlled by the administrator
    - control of incoming traffic
      (certain ASs do not learn about certain paths)

BGP messages

BGP uses four basic message formats

- OPEN
  - initiates connection to a neighbor
  - authentication of the sender
- UPDATE
  - advertise new path or
  - deletion of old information
- KEEPALIVE
  - acknowledgement after OPEN
  - indicates availability of a router (if no UPDATES are sent for a certain period of time)
- NOTIFICATION
  - indication of erroneous messages
  - indicates the termination of a connection

Intra-AS vs. inter-AS routing

Why are there different intra-AS and inter-AS routing protocols?

- Policy
  - inter-AS: policies regarding who forwards whose traffic as supported by BGP are highly important
  - intra-AS: everything is under the same administrative control and policies are less important
- Scale
  - inter-AS: scalability is critical as the number of networks grows
  - intra-AS: if an AS becomes to large it can be split into several areas using OSPF
- Performance
  - inter-AS: performance often is second to constraints due to policies
  - intra-AS: performance and efficient resource utilization are most important