Network layer

Routing

IN2140:
Introduction to Operating Systems and Data Communication

Network layer

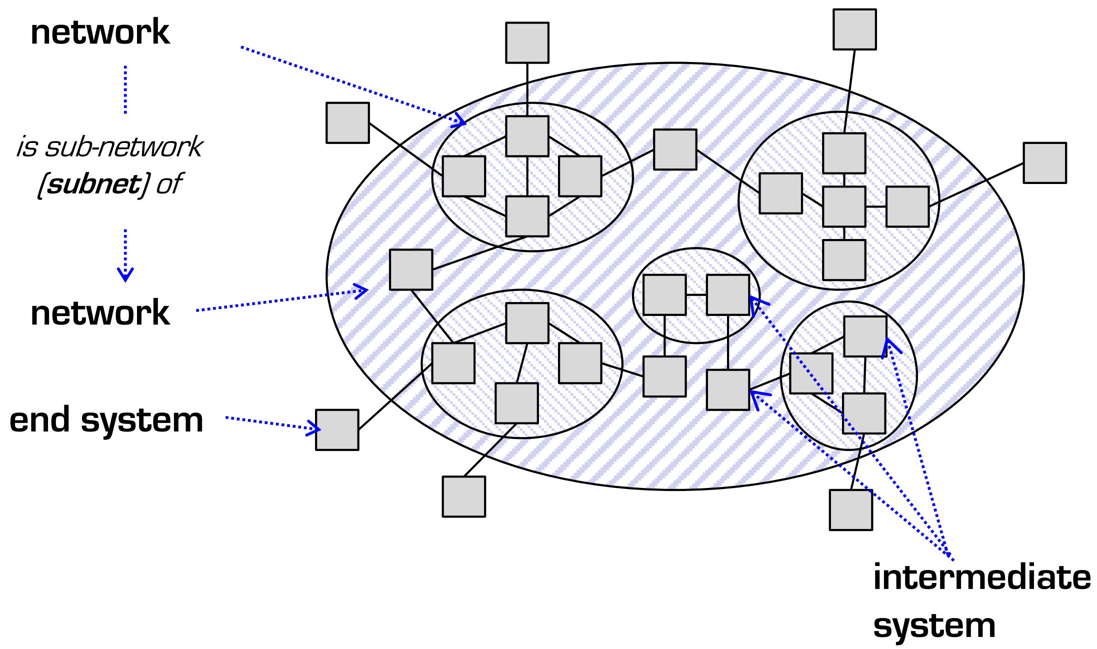
Routing - repetition

Network Layer

- Primary task from a layer model perspective
  - To provide service to the transport layer
    - Connectionless or connection-oriented service
    - Uniform addressing
    - Internetworking: provide transitions between networks
    - **Routing**
    - Congestion control
    - Quality of Service (QoS)

Routing

- The main L3 task is
  - enable data transfer from end system to end system
    - several hops, (heterogeneous) subnetworks
    - compensate for differences between end systems during transmission
- The *Intermediate Systems* are often called *Routers*



routing algorithms work on graphs

routing depends on actual connections between intermediate systems (routers)

- in many case, the organization into subnetworks coincides with routing borders and provides clues
- but it is not essential for routing

Inside the Network Layer

An L3 packet includes

headers and trailer to specify service requirements
in particular:
- information required by intermediate systems for forwarding

for connection-oriented service:
- route label

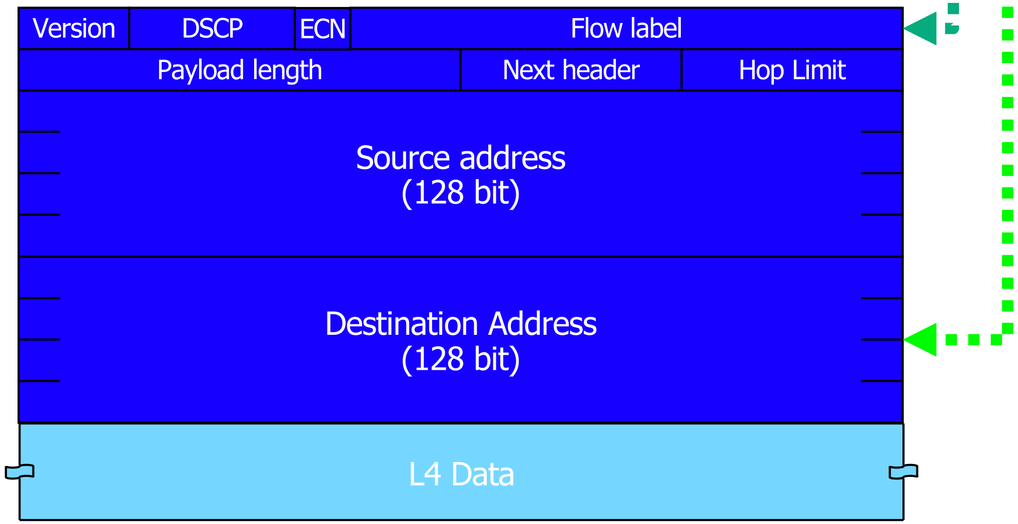
OR

for connectionless service:
- end system address of the destination

virtual circuits require routing during connection setup
- route label is used later

packet switching requires routing for every packet
- destination address is used for every packet

IPv6 Header



Network layer

Routing - foundations

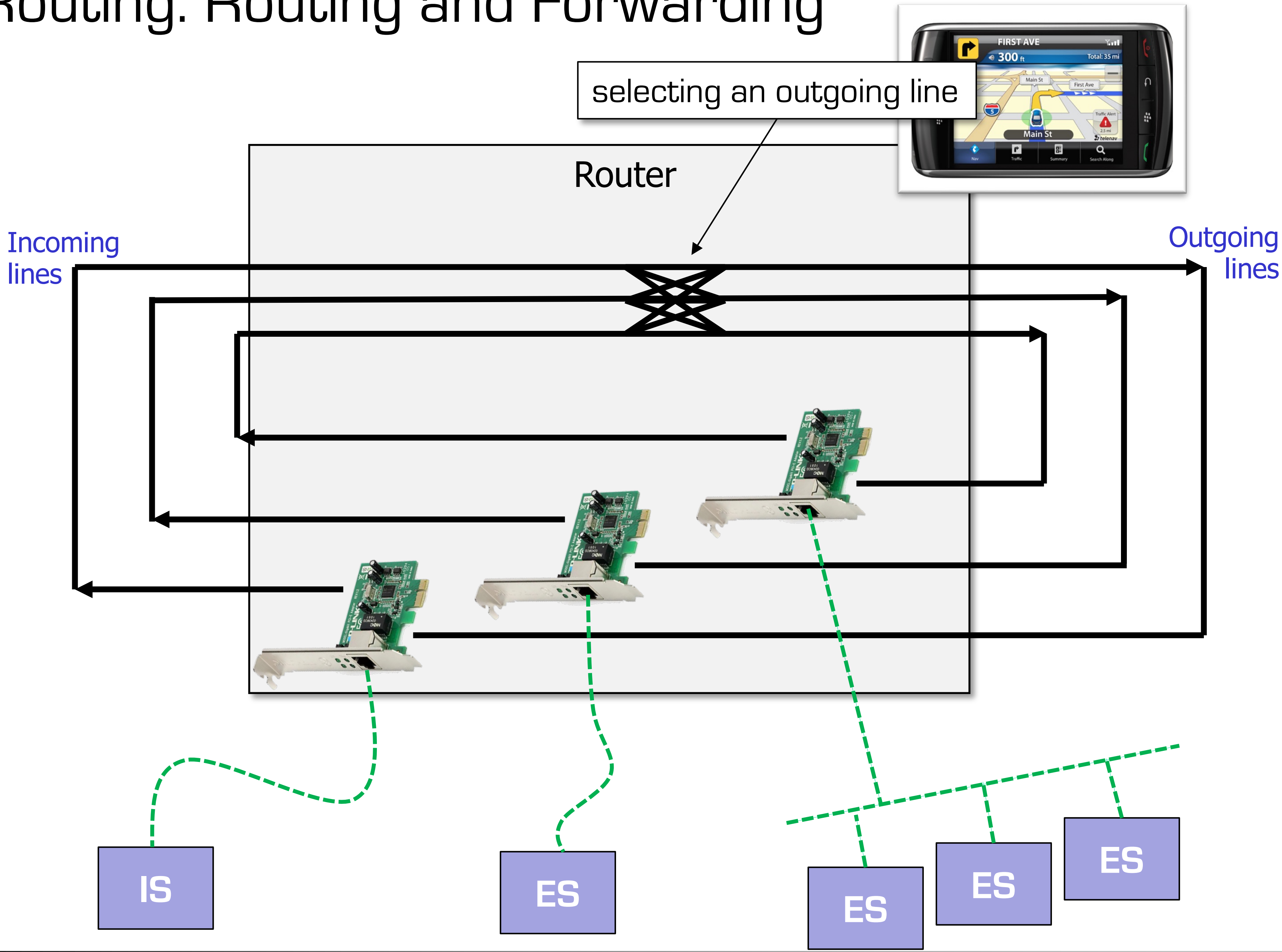
Routing: Foundations

- Task
  - To define the route of packets through the network
    - From the source
    - To the destination system

- Routing algorithm
  - Defines on which outgoing line an incoming packet will be transmitted

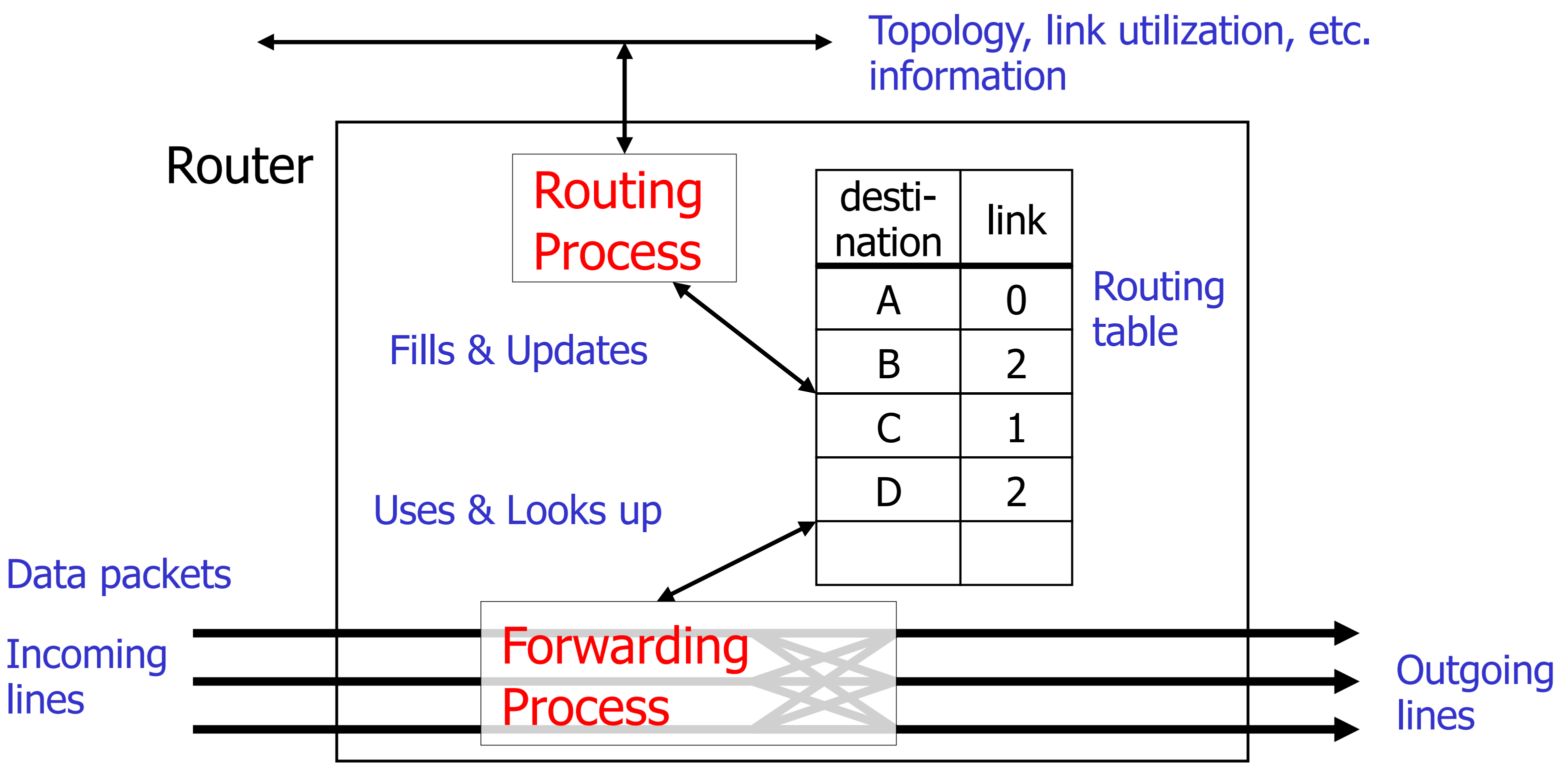
- Route determination
  - Packet
    - Routing algorithm makes individual decision for each packet
  - Virtual circuit
    - Routing algorithm runs only during connect (session routing)

Routing: Routing and Forwarding



Routing: Routing and Forwarding

- Distinction can be made
  - Routing: makes decision which route to use
  - Forwarding: what happens when a packet arrives



desti-nation	link
A	0
B	2
C	1
D	2

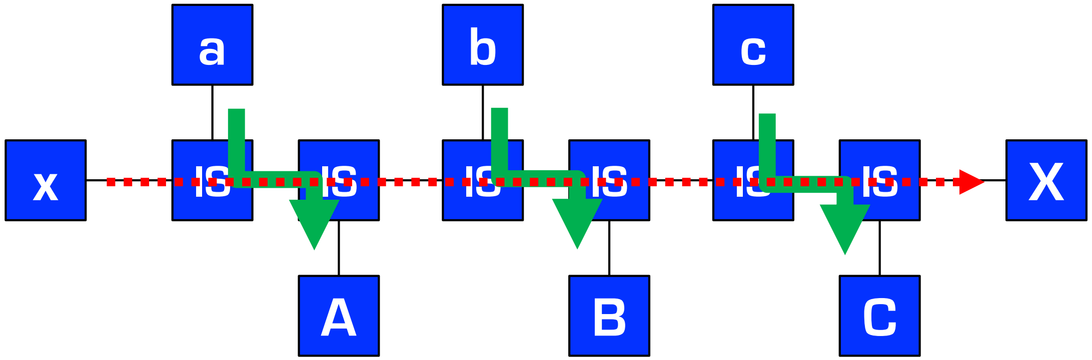
Good Properties for Routing Algorithms

- Correctness
- Simplicity
  - Minimize load of ISes
- Robustness
  - Compensation for IS and link failures
  - Handling of topology and traffic changes
- Stability
  - Consistent results
  - No volatile adaptations to new conditions
- Fairness
  - Among different sources compared to each other
- Optimality

Routing Algorithms: Conflicting Properties

- Often conflicting: fairness and optimization

- Some different optimization criteria
  - average packet delay
  - total throughput
  - individual delay
    - conflict



- Example:
  - communication among
    a → A, b → B, c → C uses full capacity of horizontal line
  - optimized throughput, but
  - no fairness for x → X – Tradeoff between fairness and optimization

- Therefore often
  - hop minimization per packet
    - it tends to reduce delays and decreases required bandwidth
    - also tends to increase throughput

Classes of Routing Algorithms

- Class: Non-adaptive Algorithms
  - Current network state not taken into consideration
    - Assume average values
    - All routes are defined off-line before the network is put into operation
    - No change during operation (static routing)
  - With knowledge of the overall topology
    - Spanning tree
    - Flow-based routing
  - Without knowledge of the overall topology
    - Flooding

[Inkowik@wikipedia, CC BY SA 2.0]

- Class: Adaptive Algorithms
  - Decisions are based on current network state
    - Measurements / estimates of the topology and the traffic volume
  - Further sub-classification into
    - Centralized algorithms
    - Isolated algorithms
    - Distributed algorithms

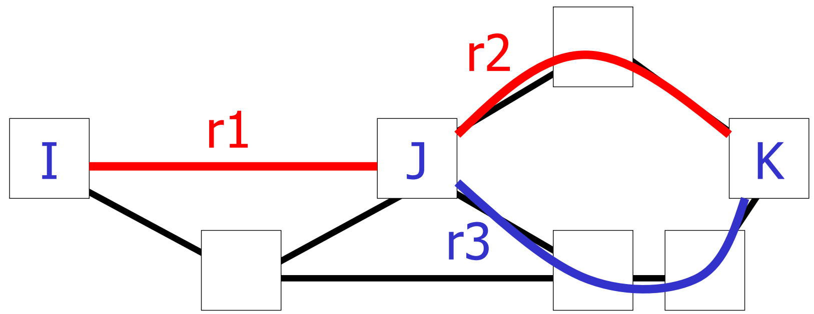
[Ingolfsson@wikipedia]

Optimality Principle and Sink Tree

- Starting idea: using a route has a *cost*
  - number of hops, delay, ...

- General statement about optimal routes
  - if router J is on the optimal path from router I to router K
  - then the optimal path from router J to router K uses the same route

- Idea of the proof
  - best route from I to K is like this:
    - r1: from I to J, then
    - r2: from J to K
  - then r2 is also the best route from J to K
  - if better route r3 from J to K would exist
    - then concatenation of r1 and r3 would improve route from I to K

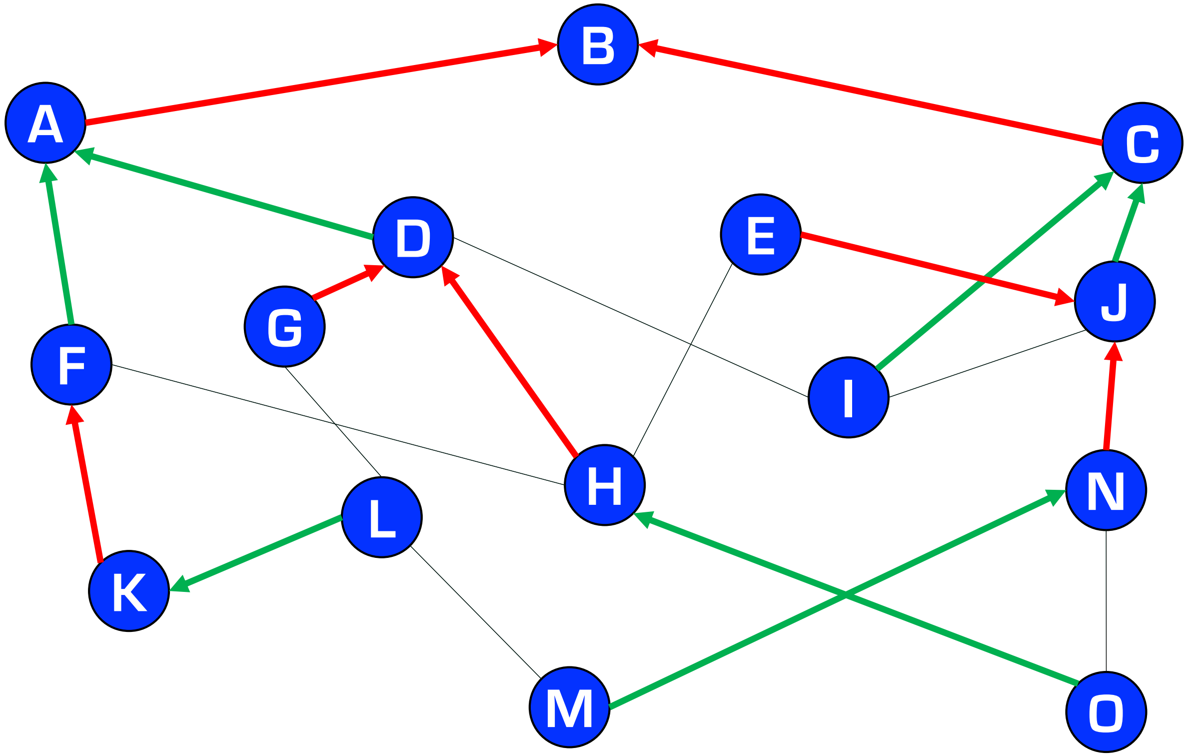


- Set of optimal routes
  - from all sources
  - to a given destination

  form a tree rooted at the destination: ***Sink Tree***

Sink Tree

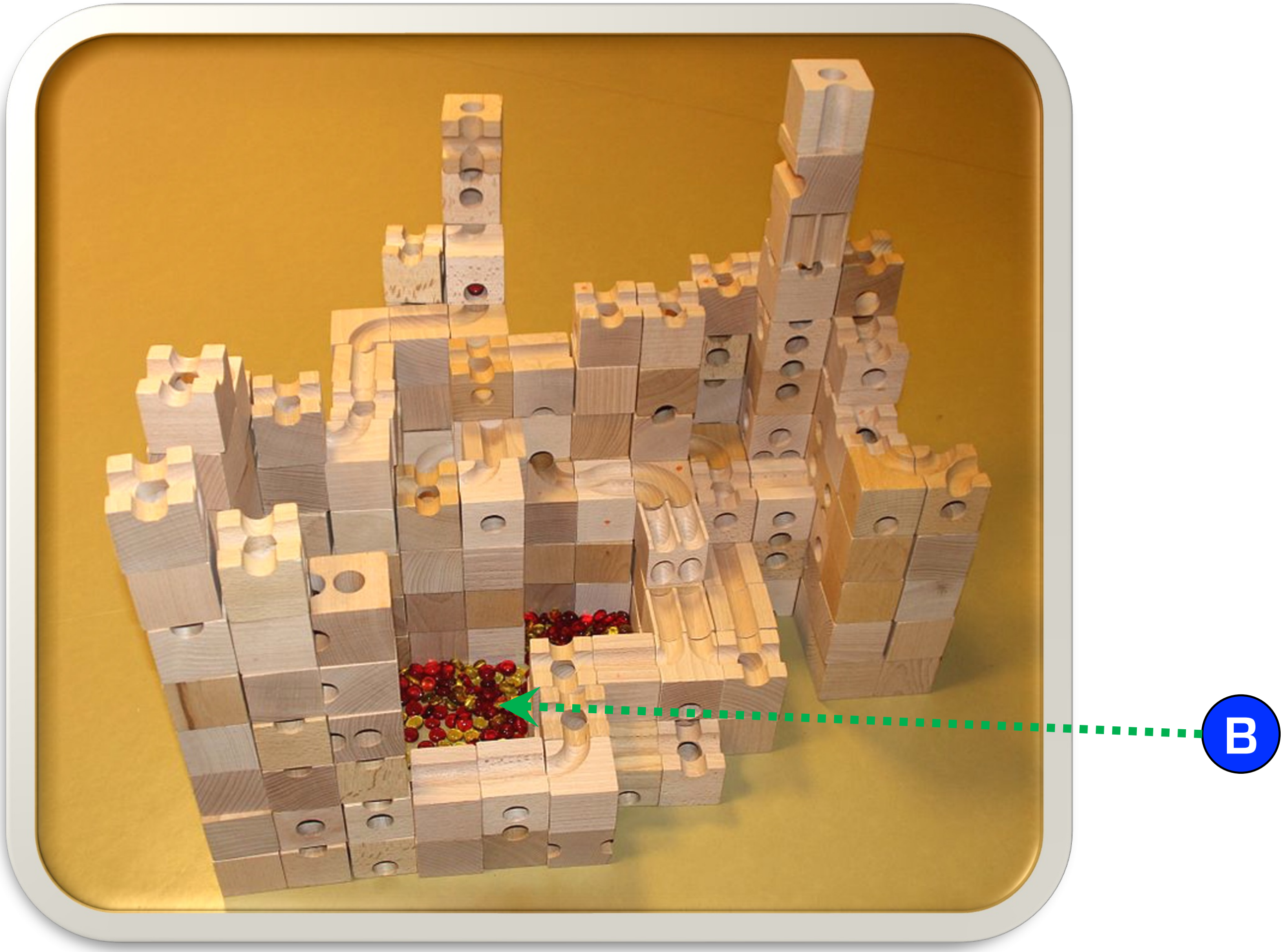
Sink Tree for Destination B



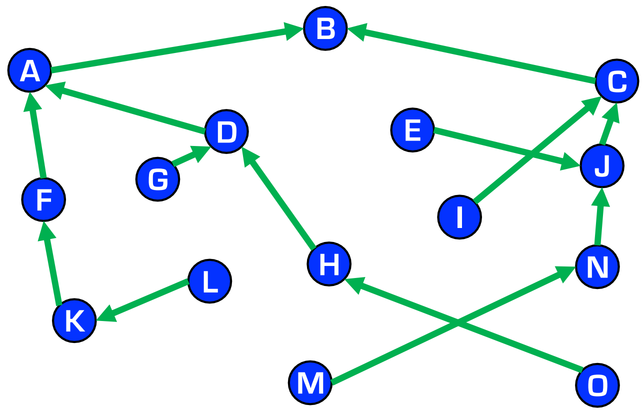
B	0
A	1
C	1
F	2
D	2
I	2
J	2
N	3
E	3
H	3
G	3
K	3
L	4
O	4
M	4

Sink Tree

Sink Tree for Destination B



Sink Tree



Sink Tree for Destination B

- Comments
  - tree: no loops
    - each optimal route is finite with bounded number of hops
  - not necessarily unique
    - other trees with same path lengths may exist

- Goal of all routing algorithms
  - discover and use the Sink Trees for all routers

- Not realistic to use Sink Trees as real-life routing algorithm
  - need complete information about topology
  - Sink Tree is only a benchmark for routing algorithms

Methodology & Metrics

- Compute the *shortest path* between a given pair of routers

- Different metrics for path lengths can be used
  - can lead to different results
  - sometime even combined

- Metrics for the "ideal" route, e.g., a "short" route
  - number of hops
  - geographical distance
  - bandwidth
  - average data volume
  - cost of communication
  - delay in queues
  - ...

Network layer

Distributed Routing:

Link State Routing

Link State Routing

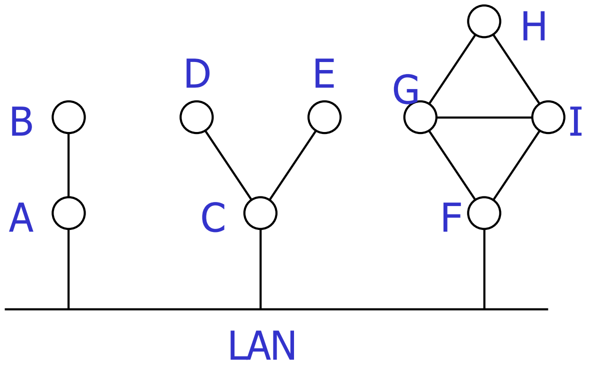
- A very frequently use routing protocol
  - IS-IS (Intermediate System-Intermediate System)
  - OSPF (Open Shortest Path First)

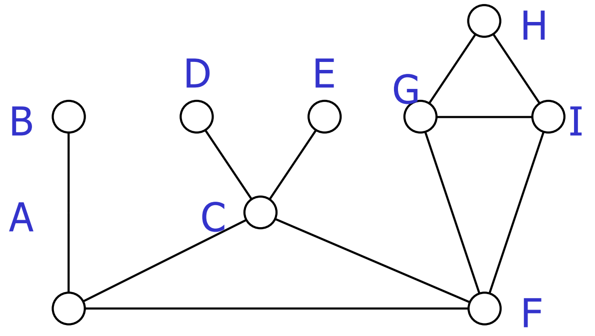
- Basic principle
  - IS measures the "distance" to the directly adjacent IS
  - Distributes information
  - Calculates the ideal route

- Procedure
  1. Determine the address of adjacent IS
  2. Measure the "distance" (delay, ...) to neighbouring IS
  3. Organize the local link state information in a packet
  4. Distribute the information to all IS
  5. Calculate the route based on the information of all IS

Link State Routing

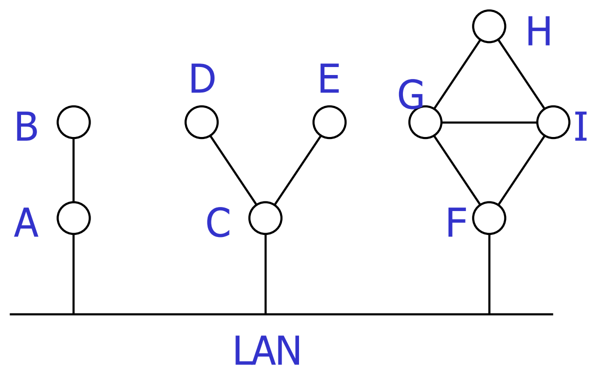
1. Phase: gather information about the adjacent intermediate systems

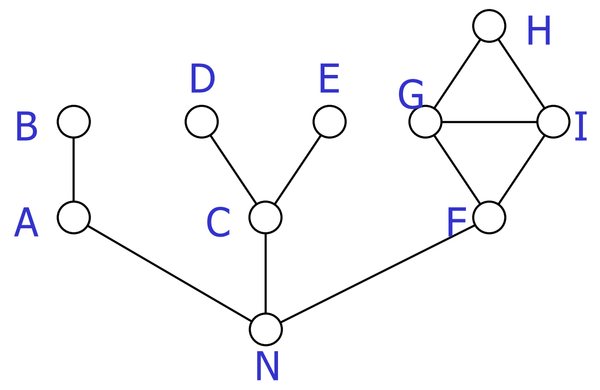




Link State Routing

1. Phase: gather information about the adjacent intermediate systems





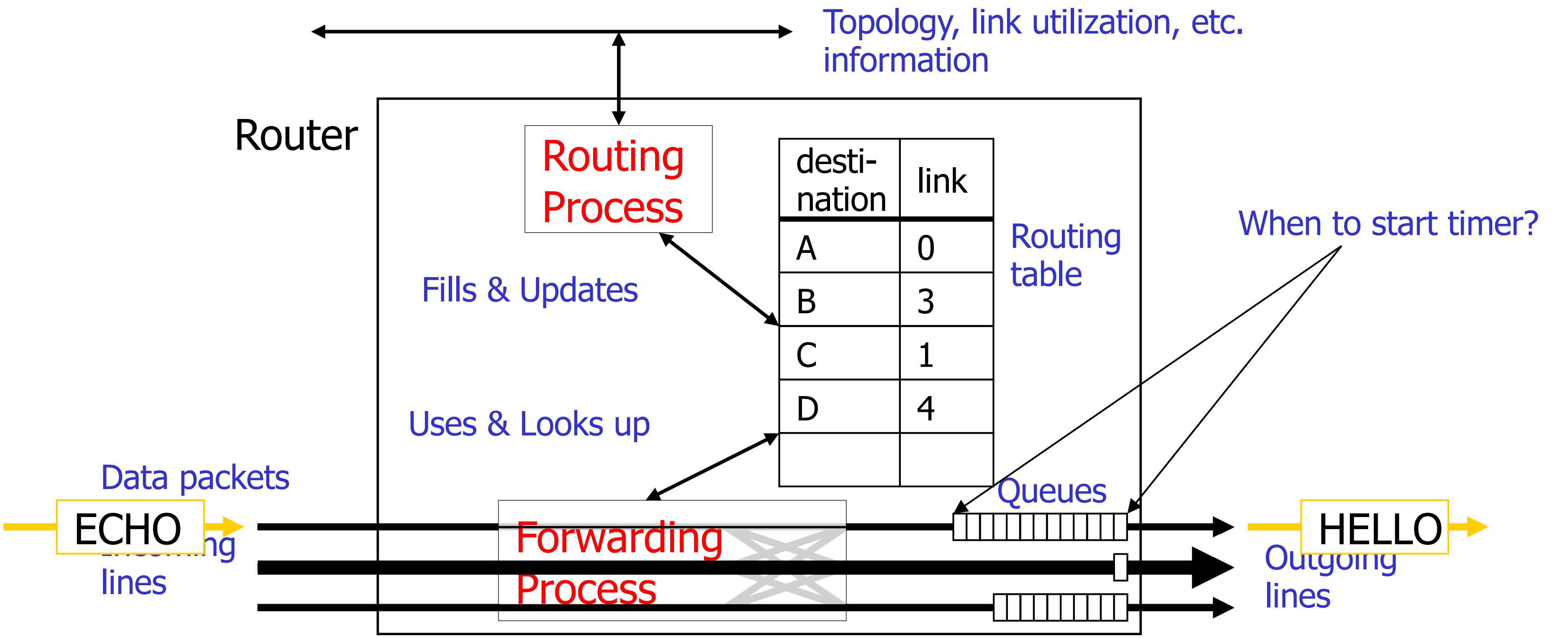
Initialization procedure

- New IS
  - Sends a HELLO message over each L2 channel
- Adjacent IS
  - Responds with its own address, unique within the network

Link State Routing

2. Phase: measure the "distance"

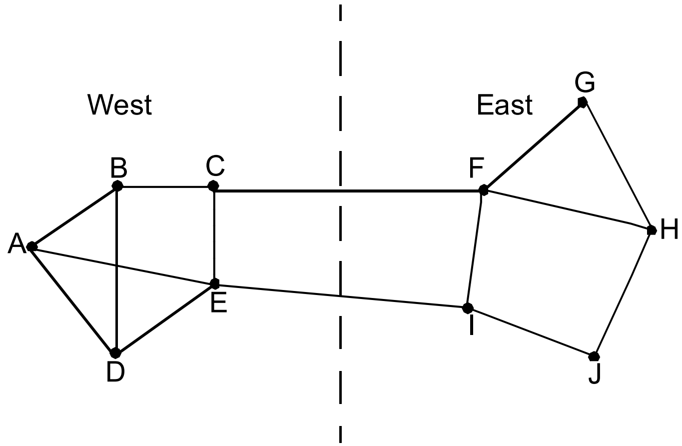
- Definition of distance needed
  - Usually delay
  - Where to measure?



Link State Routing

2. Phase: measure the "distance“

- Queuing delay
  - Measuring without does not take load into account
  - Measuring with does ⇒ usually better

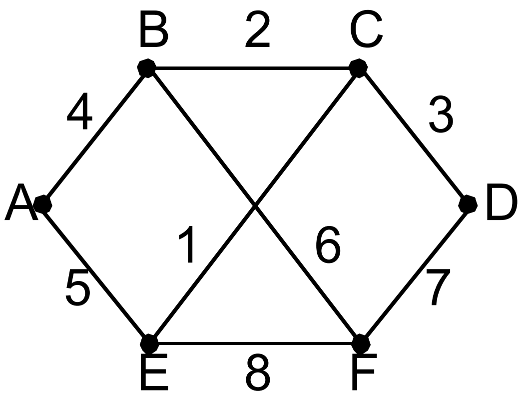


- But
  - Possibility for oscillations (route flapping)
  - Once per routing table update

Link State Routing

3. Phase: organizing the information as link state packet

- Including own address, sequence number, age, "distance"
- Timing problems: validity and time of sending
  - Periodically
  - In case of major changes



Link State Packets:

A	
Seq.	
Age	
B	4
E	5

B	
Seq.	
Age	
A	4
C	2
F	6

C	
Seq.	
Age	
B	2
D	3
E	1

D	
Seq.	
Age	
C	3
F	7

E	
Seq.	
Age	
A	5
C	1
F	8

F	
Seq.	
Age	
B	6
D	7
E	8

Link State Routing

4. Distributing the local information to all IS

- By applying the flooding procedure (very robust)
  - Therefore sequence number in packets
- Problem: inconsistency
  - Varying states simultaneously available in the network
  - Indicate and limit the age of packet,
    i. e. IS removes packets that are too old

5. Computing new routes

- Each IS for itself
- Possibly larger amount of data available

IN2140:
Introduction to Operating Systems and Data Communication

Network layer

Distributed Routing:

Distance Vector Routing

Distance Vector Routing

Principle

- every IS maintains a table (i.e., vector) stating
  - best known distance to destinations
  - and line to be used
- ISes update tables
  - by exchanging routing information with their neighbors

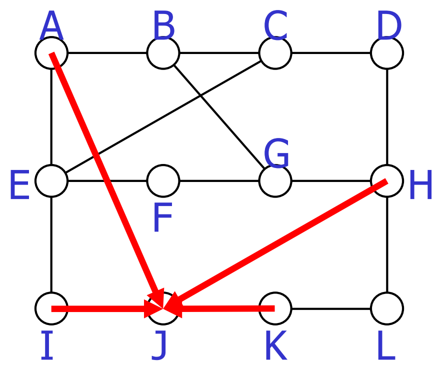
Distance Vector Routing

- Each IS
  - maintains routing table with one entry per router in the subnet
  - is assumed to know the distances to each neighbor
  - sends list with estimated distances to each destination *periodically* to its neighbors

- X receives list E(Z) from neighbor Y
  - Distance X to Y: e
  - Distance Y to Z: E(Z)
  - Distance X to Z via Y: E(Z)+e

- IS computes new routing table from the received lists containing
  - Destination IS
  - Preferred outgoing path
  - Distance

Distance Vector Routing



	A	I	H	K		line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K

delay	JA	JI	JH	JK
	8	10	12	6

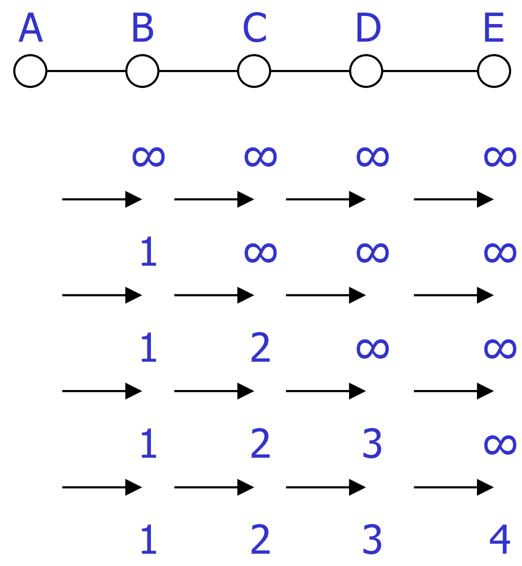
- Previous routing table will not be taken into account
  - Reaction to deteriorations

Distance Vector Routing

- Fast route improvement

- Fast distribution of information about new short paths (with few hops)

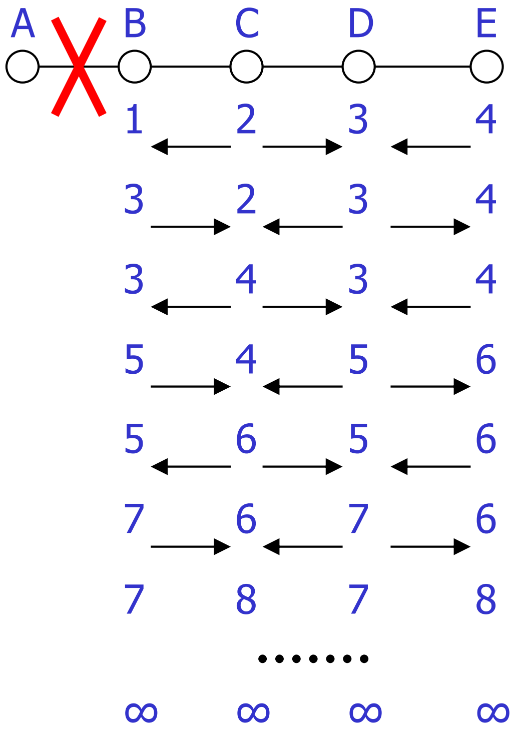
- Example
  - initially A unknown
  - later: A connected with distance 1 to B, this will be announced
  - Distribution proportional to topological spread
  - Synchronous (stepwise) update is a simplification



Distance Vector Routing

- Slow distribution of information about new long paths (with many hops)
- "Count to Infinity" problem of DVR

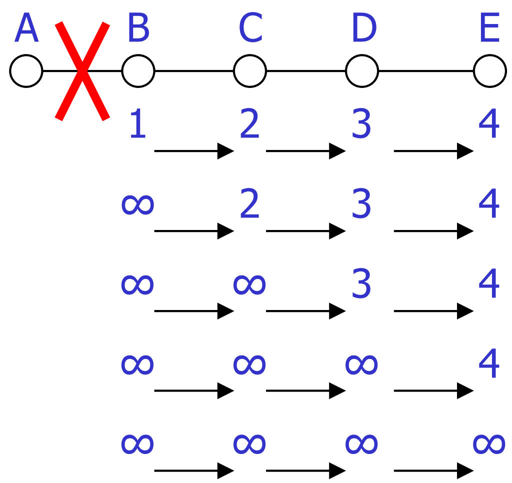
- Example: deterioration
  - Here: connection destroyed
  - A was previously known, but is now detached
  - The values are derived from (incorrect) connections of distant IS



- Comment
  - Limit "infinite" to a finite value, depending on the metrics, e.g.
    - 'infinite' = maximum path length+1

Distance Vector Routing

- Variant: 'Split Horizon Algorithm'
- Objective: improve the "count to infinity" problem
- Principle
  - In general, to publicize the "distance" to each neighbour
  - If neighbor Y exists on the reported route,
    X reports the response "false" to Y
    - distance X (via Y) according to arbitrary i: ∞



- Example: deterioration (connection destroyed)
  - B to C: A = ∞ (real),
    C to B: A = ∞ (because B is on path to A), ...

- But: still poor, depending on topology, example
  - Connection CD is removed
  - A receives "false information" via B
  - B receives "false information" via A
    - Slow distribution (just as before)