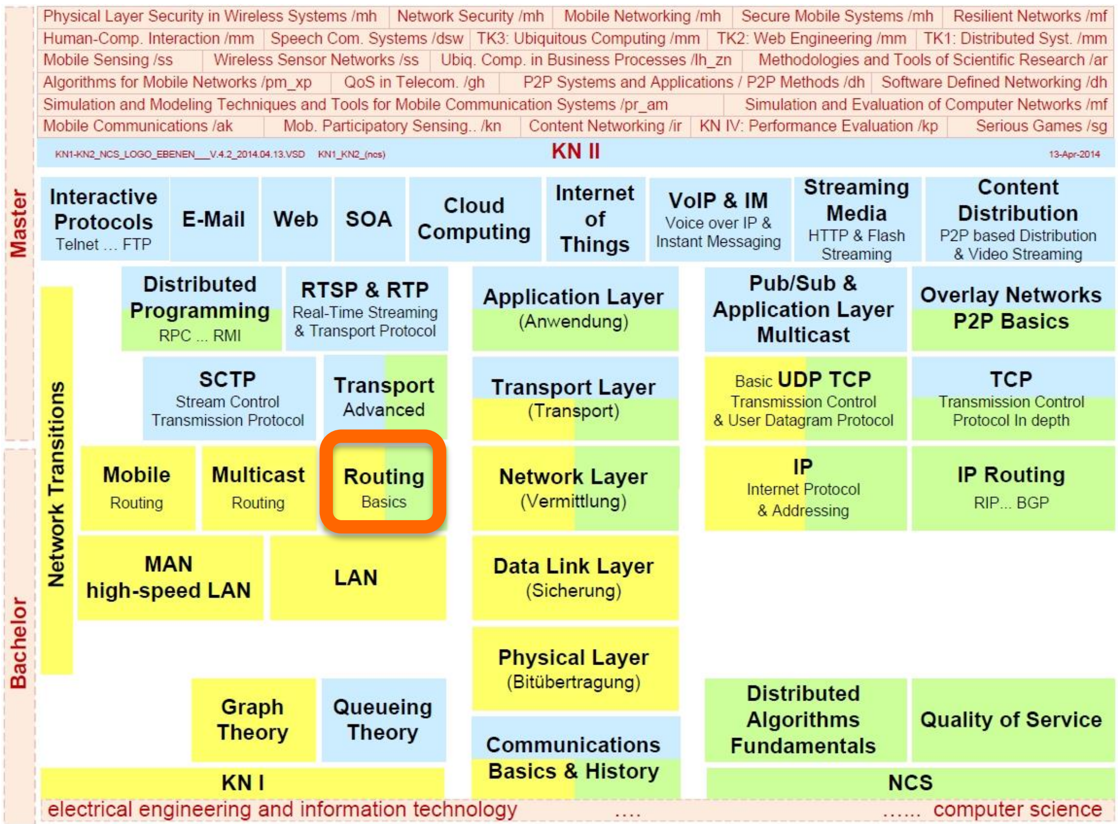
Communication Networks I

Routing

TECHNISCHE UNIVERSITÄT DARMSTADT



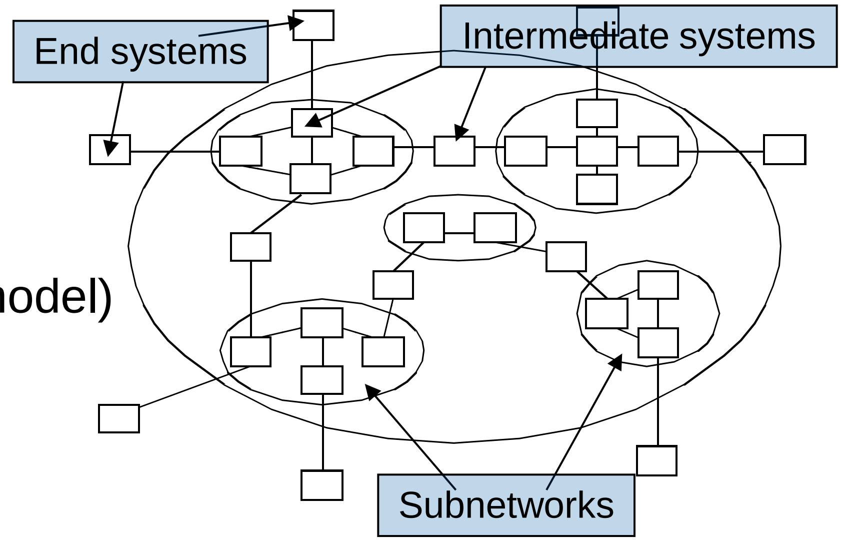
Prof. Dr.-Ing. **Ralf Steinmetz**
KOM - Multimedia Communications Lab

Overview

1 Foundations of Routing

Task of routing

- Comp. A wants to send message to B
- A and B are both part of a larger network
- To find a route (path) through the network from A to B
- Belongs to Network Layer (layer 3 in OSI model)



Routing algorithm determines the path

- Network consists of
  - End systems and
  - Routers
- Router runs routing algorithm and forwards packets to the right nodes
  - Defines on which outgoing line an incoming packet will be transmitted
- Given the network, routing algorithm finds a “good” path from A to B
  - “Good” typically means “lowest cost”

Different networks have different routing algorithms

- Internet uses several routing algorithms “simultaneously”

1.1 Datagrams vs. Virtual Circuits & Virtual Circuits vs. Connection Oriented

….

- Datagram networks

  vs

- Virtual circuit networks

….

- In connection-oriented, only the end-systems know they are connected

  vs

- In virtual circuit, every intermediate system knows about the connection

Datagrams vs. Virtual Circuits (Refinement)

Route determination (i.e., 2 main types of networks)

- Datagram networks
  - Routing decisions made for each packet individually
  - Routers can be made simpler

- Virtual circuit networks
  - When A wants to send a packet to B:
    - Set up a virtual circuit (connection) from A to B
    - Send data
    - Tear down the virtual circuit
  - Routing decisions made only during connection setup
    - All subsequent packets use the same route
  - Each intermediate router must keep track of virtual circuits

Note: Difference between datagrams and virtual circuits similar to "connectionless" and "connection-oriented"

Virtual Circuits vs. Connection-Oriented (Refinement)

There is a subtle difference between "virtual circuit" (layer 3) and "connection-oriented" (layer 4)

- In connection-oriented, only the end-systems know they are connected
- In virtual circuit, every intermediate system knows about the connection

Internet has a datagram network as layer 3

- Datagram passes through the network as an isolated unit
  - Has complete source and destination addresses
  - Individual route selection for each datagram
  - Generally no resource reservation
  - Correct sequence not guaranteed

On layer 4, there is both

- Connection-oriented and connectionless services possible

Network layer typically offers

- Either datagram or virtual circuit service,
- But not both

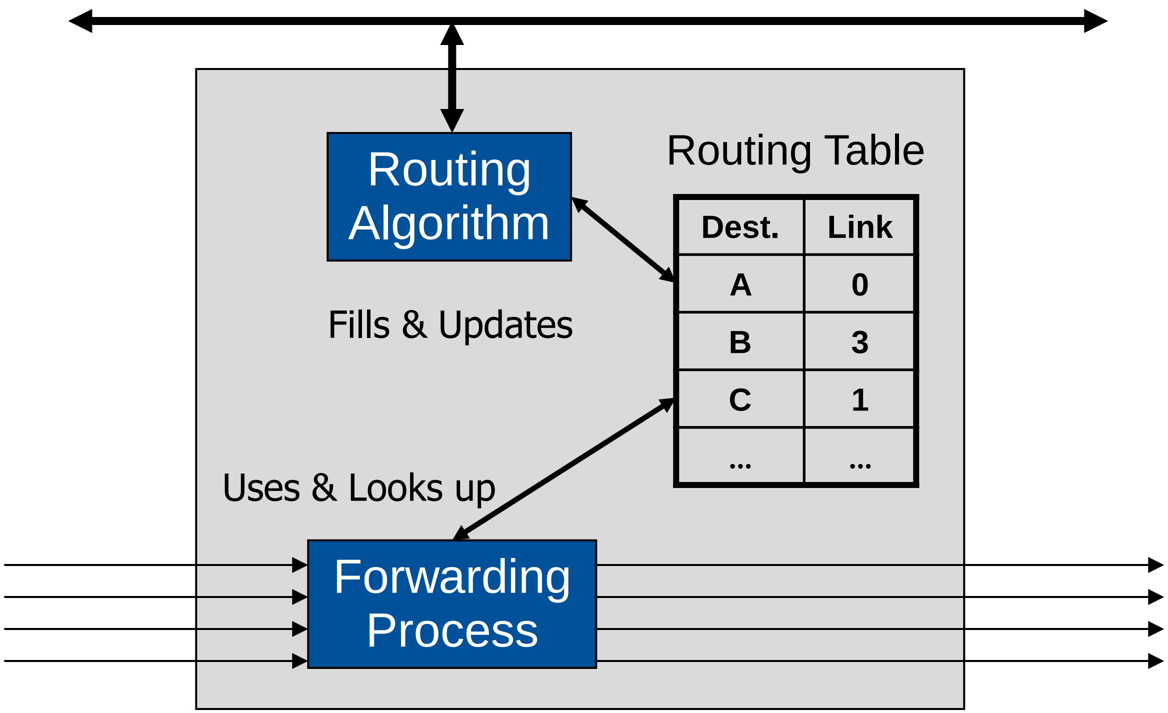
TECHNISCHE UNIVERSITÄT DARMSTADT

Network consists of end-systems and intermediate systems (IS)

- Intermediate systems also called routers

Routers have two main functions

- Routing: Determine which route to use
- Forwarding: What happens when a packet arrives?



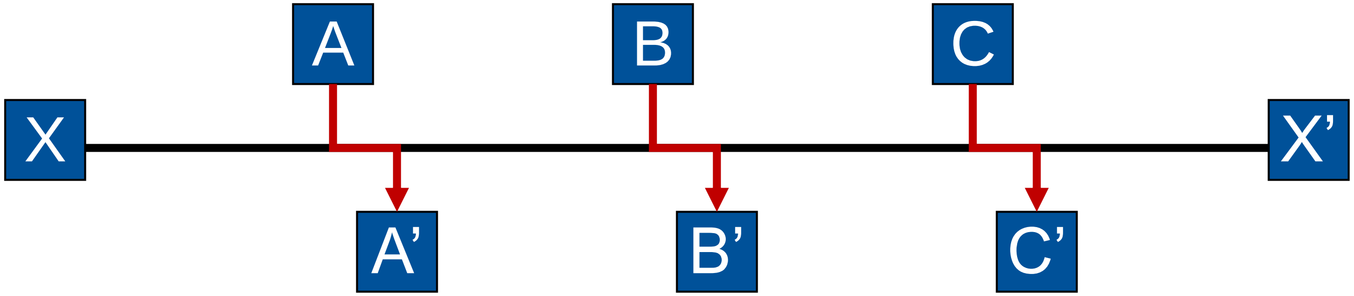
Dest.	Link
A	0
B	3
C	1
...	...

1.3 Desirable and (sometimes) conflicting Properties

Desirable properties

- Correctness, Simplicity, Robustness, Stability, Fairness, Optimality

Often conflict between fairness and optimization, e.g.:



Optimal use of horizontal line is to allow A->A’, B->B’, and C->C’ to use full capacity

- But this is not fair to X and X’

Different Optimization Criteria

Some different optimization criteria for routing algorithms
- Average packet delay
- Total throughput
- Individual delay

→ May lead to conflicts with other criteria

In practice, routing algorithms attempt to minimize number of routing hops per packet
- Tends to reduce delays and decreases bandwidth requirements
- Also tends to increase throughput
- No guarantees about optimality,
  - But “good enough” and
  - Fulfills the required properties “well-enough”

1.4 Classes of Routing Algorithms

Class Non-adaptive Algorithms

- Current network state not taken into consideration
- Class members with knowledge of the overall topology
  - Like spanning tree, flow-based routing
- Class without knowledge of the overall topology
  - Like flooding

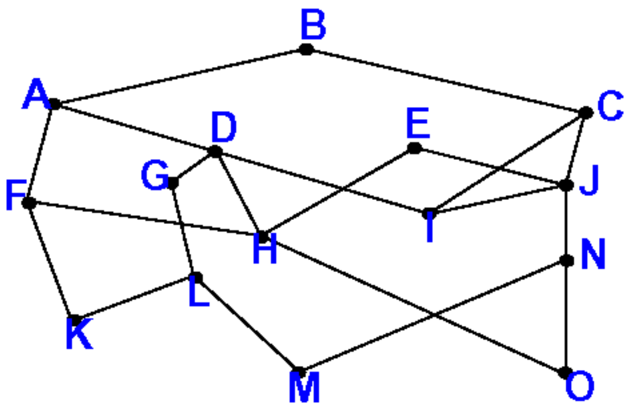
Enhancements (adaptive and non-adaptive algorithms)

- multiple routing and hierarchical routing definition

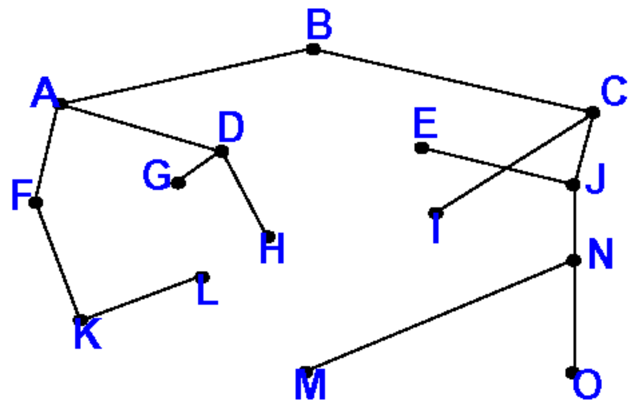
Class Adaptive Algorithms

- Decisions are based on current network state
- Subclasses type according to
  - Centralized algorithms
  - Isolated algorithms
  - Distributed algorithms
    - like distance vector algorithms (link state routing,..)
- Or subclasses type according to
  - Global knowledge available at each node
  - Distributed knowledge available at each node

Sink Tree: Example



Subnet



Sink Tree for Destination Node B

Comments

- Tree: no loops
  - Each packet reaches its destination within finite and bounded number of hops
- Not necessarily unique
  - Other trees with same path lengths may exist

Goal of all routing algorithms

- To discover and to use the sink trees for all routers

Further comments

- Information about network topology necessary for sink tree computation
  - yet, sink tree provides benchmark for comparison of routing algorithms

Networks represented as graphs

- Node - represents a router
- Arc - represents a communication line (link)

Compute the SHORTEST PATH between a given pair of routers

Different metrics for path lengths can be used

- Can lead to different results
- Sometimes even combined
  - (but this leads to computational problems)

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
- ...

UNICAST Routing & NON-ADAPTIVE

I.e.

- Current network state not taken into consideration
  - to 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

2 Non-Adaptive Shortest Path Routing

Static Procedure

- Network operator generates tables
- Tables
  - are loaded when IS operation is initiated and
  - will not be changed any more

Characteristics

+ simple
+ good results with relatively consistent topology and traffic
- poor performance, if traffic volume or topologies change over time

See graph theory – e.g., Dijkstra

Usage

- Topology
- Average utilization and available capacity per edge/sub-path
  - sometimes useful to choose a route that is longer but available

Procedure

- Given: assumption for a path's average load over a pre-selected path

1. Computation of the average delay per edge

- by means of queuing theory
- average delay at an edge

$$T_i = \frac{1}{\text{edge capacity - average edge utilization}} = \frac{1}{\mu C_i - \lambda_i}$$

- includes
  - service time (occurs also during no load, li=0)
  - actual waiting time

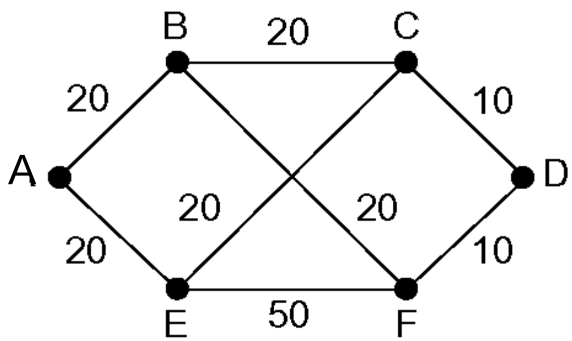
2. Computation of the total average delay of a subnetwork

- by weighted sum of the delays at single edges

3. Different total average delays result from selecting different paths

- subnetwork with MINIMAL OVERALL DELAY used for routing

Non-Adaptive Flow-Based Routing



Source \ Destination	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF
B	9 BA		8 BC	**3 BFD**	2 BFE	4 BF
C	4 CBA	8 CB		3 CD	3 CE	2 CEF
D	1 DFBA	3 DFB	3 DC		3 DCE	4 DF
E	7 EA	2 EFB	3 EC	3 ECD		5 EF
F	4 FEA	4 FB	2 FEC	4 FD	5 FE	

Example: Assumption / Requirements

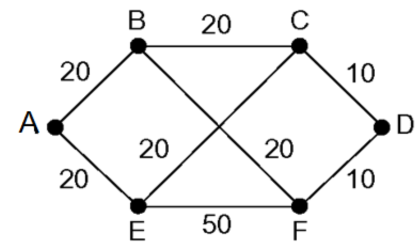
- Network with fully duplex channels
- Given TOPOLOGIES and CAPACITIES
- Given paths to be selected including number of packets/sec
  - example from B to D: path BFD with 3 packets/sec
  - MATRIX pre-defined by a different algorithm
  - overall solution varies depending on the matrix

Example: initial information

Edge xy	λ_{xy} (pkts/sec)	C_{xy}(kbps)	μC_{xy}(pks/sec)	T_{xy}(msec)	$Weight_{xy}$
AB		20			
BC		20			
CD		10			
AE		20			
EF		50			
FD		10			
BF		20			
EC		20			

Non-Adaptive Flow-Based Routing

	A	B	C	D	E	F
A		9 AB	4 ABC	1 ABFD	7 AE	4 AEF



λ_{xy}: Average load

- the sum of all median packets/sec at the respective edge
- example: λ_{AB} = AB (AB=9) + AC (ABC=4) + AD (ABFD=1) = 14

C_{xy}: Capacity of each edge in kbps (known from the graph)

μC_{xy}: Capacity of each edge at given median packet size

- example: AB: 20 kbit/sec and packets in median 800 bit/packet

$$\mu C_{xy} = \frac{20\ kbits/sec}{800\ bit/packet} = 25\ packets/sec$$

T_{xy}: Average delay on each edge

$$T_{xy} = \frac{1}{\mu C_{xy} - \lambda_{xy}} = \frac{1}{25\ packets\ /\ \sec - 14\ packets/sec} = 90{,}9\overline{09}\ msec/packet$$

Non-Adaptive Flow-Based Routing

Example: final results

Edge xy	λ_{xy} (pkts/sec)	C_{xy}(kbps)	μC_{xy}(pks/sec)	T_{xy}(msec)	Weight_{xy}
AB	14	20	25	91	0.171
BC	12	20	25	77	0.146
CD	6	10	12.5	154	0.073
AE	11	20	25	71	0.134
EF	13	50	62.5	20	0.159
FD	8	10	12.5	222	0.098
BF	10	20	25	67	0.122
EC	8	20	25	59	0.098

Non-Adaptive Flow-Based Routing

1. Computation of the average delay per edge

- Weight: the relative traffic of data using this edge (in relation to the overall traffic)

$$Weight(AB) = \frac{\lambda_{AB}}{\sum_{all\ lines\ xy} \lambda_{xy}} = \frac{14}{82} = 0{,}1707$$

2. Computation of the total average delay of a subnetwork

$$\sum_{all\ lines\ xy} Weight(xy) \cdot T_{xy} = 86 msec$$

3. Different total average delays result from selecting different paths subnetwork with minimal overall delay used for routing

4 Non-Adaptive Flooding

Principle: IS transmits the received packet to all adjacent IS

- Except over the path it came in
- But generates "an infinite amount" of packets

Methods to limit packets

1. HOP COUNTER in the packet header

- Each IS decrements this hop counter
- When the hop counter = 0
  - the packet is discarded
- Initialization for maximum path length (if known);
  - worst case: subnet diameter

2. Each STATION

- REMEMBERS THE PACKETS THAT HAVE ALREADY BEEN TRANSFERRED
- And deletes them upon recurrence
- I.e.
  - source router inserts sequence number into packets received from hosts
  - each router needs an "already seen sequence number" list per source router
  - packets with sequence number on list is dropped
  - sequence number list must be prevented from growing without bounds
  - store only upper-counter / highest sequence number(s)

Variation: Selective Flooding

Approach

- Do not send out on every line
- IS transmits received packet to adjacent stations,
  LOCATED IN THE DIRECTION OF THE DESTINATION

Comment

- With 'regular' topologies this makes sense and is an optimization
- But some topologies do not fit well to this approach
- Geographically-oriented routing got recent interest for mobile scenarios

Flooding: Evaluation and use

- Overhead: not practical in most applications
- Extremely robust: military use
- Reaches all IS: e.g., the exchange of control data between nodes
- Initialization phase: does not need information about the topology
- Always finds shortest path: use as benchmark

Class ADAPTIVE ALGORITHMS

- Decisions are based on
  - current network state
  - measurements / estimates of the topology
  - the traffic volume

Further sub-classification into

- Centralized algorithms
- Isolated algorithms
- Distributed algorithms

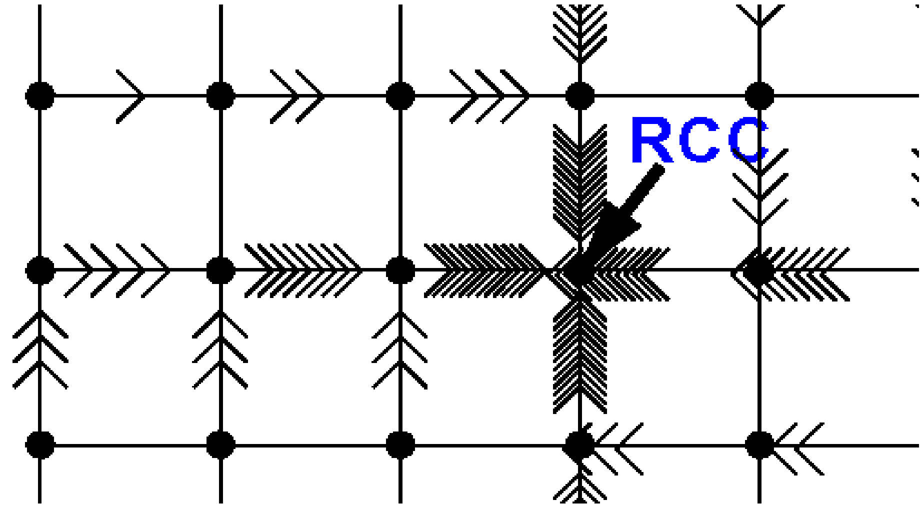
Adaptive Centralized Routing

Principle

- in the network:
  - RCC (Routing Control Center)

- each IS sends periodically information on the current status to the RCC
  - list of all available neighbors
  - actual queue lengths
  - line utilization, etc.

- Routing Control Center RCC
  - collects information
  - calculates the optimal path for each IS pair
  - forms routing tables and distributes these to IS



Example: TYMNET

- packet exchanging network
- 1000 nodes/IS
- virtual circuits

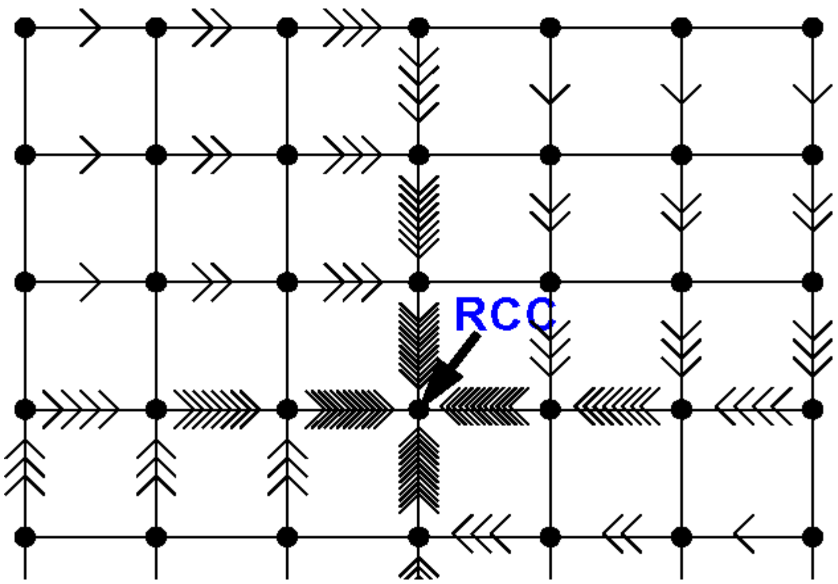
Adaptive Centralized Routing

Characteristics

- Routing Control Center RCC has complete information → perfect decisions
- IS is free of routing calculations

But

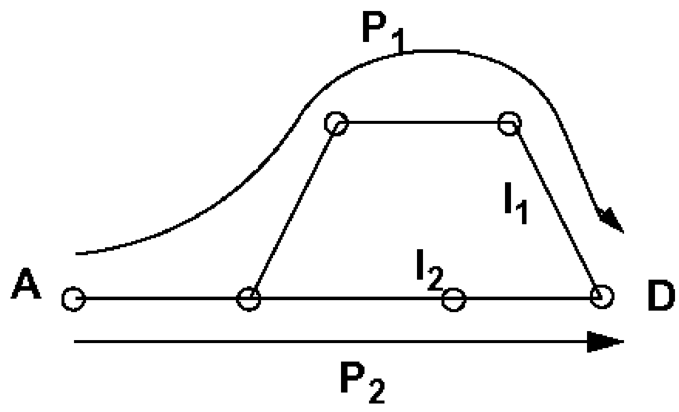
- Re-calculations quite often necessary (approx. once/min or more often)
- Low robustness
- No correct decisions if network is partitioned
- IS receive tables at different times
- Traffic concentration in Routing Control Center RCC proximity



Adaptive Isolated Routing – Backward Learning

Isolated routing

- Every IS makes decision based on locally gathered information only
  - No exchange of routing information among nodes
  - Only limited adaptation possibility to changed traffic / topology



IS "learns" from received packets (..., S, C, ...)

- S : source - IS
- C : hop counter

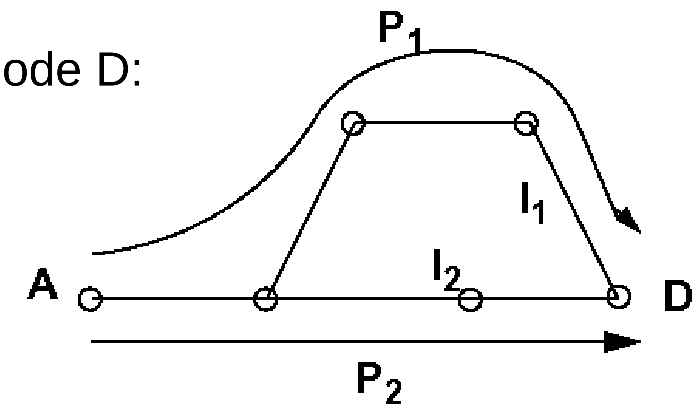
Routing table in IS

- Per line: L-table
  - (destination - IS, outgoing line, C_{min})
- Update of the routing table
  - IS receives packet (..., S, C, ...) on L

Adaptive Isolated Routing – Backward Learning

Example

- Packet (..., source - IS, section counter, ...) at node D:
- P1 (..., A, 4, ...) → Add (A, I_1, 4)
- P2 (..., A, 3, ...) → Update (A, I_2, 3)



Problem

- Packets use a different route, e. g., because of failures, high load
- Algorithm retains only the old value (because it was "better"),
  - i.e., algorithm does not react to deteriorations

Solution

- Periodic deletion of routing tables
  - (new learning period)
- Table deletion
  - too often: mainly during the learning phase
  - not often enough: reaction to deteriorations too slow

Distance-Vector Routing

Group of DISTANCE VECTOR ROUTING ALGORITHMS

- Also known as distributed Bellman-Ford algorithm, Ford-Fulkerson algorithm

Use

- Was the original ARPANET routing algorithm
- Has been used in the Internet as RIP ROUTING INFORMATION PROTOCOL

Basic principle

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

Distance Vector Routing: Principle

TECHNISCHE UNIVERSITÄT DARMSTADT

Each node

- Maintains routing table (distance vector) with one entry per router (of the subnet)
  - Best known distance to (all) destinations
  - Next hop towards a given destination
- Is assumed to know the distances to each neighbor

Node sends list with

- Estimated distances & “first hop to be used” to/for all known destinations
- Periodically
- To its neighbors

Node X receives list E(Z) from neighbor Y

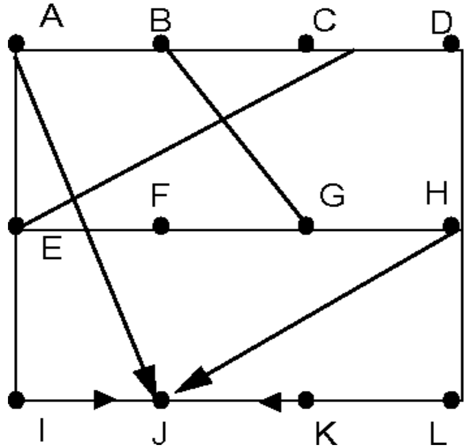
▪ Knows	distance	X to neighbor Y:	e
▪ Gets	distance	(neighbor) Y to Z:	E(Z)
▪ Computes	distance	X to Z via (neighbor) Y:	E(Z) + e
▪ Decides	which path to use		

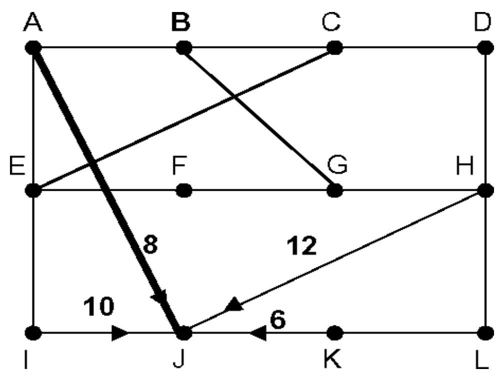
Node computes new routing table from the received lists containing

- Destination node
- Preferred outgoing path
- Distance

7.1 Overall Example - Distance-Vector Routing

Subnet





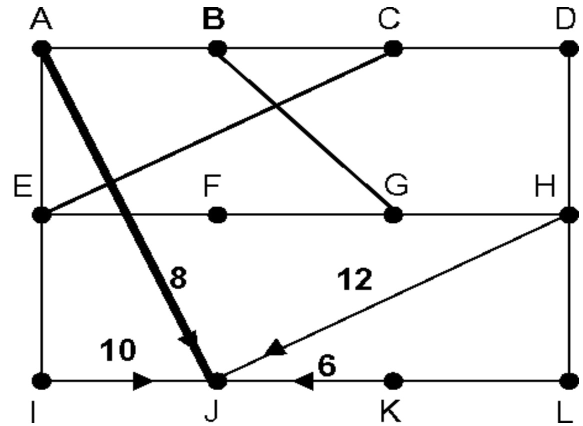
Delays from nodes A/I/H/K/. (column).
to nodes A;B;C;D… (row)

To	routing table of A	routing table of I	routing table of H	routing table of K	new estimated delay from J	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	8	10	12	6	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA Delay=8	JI Delay=10	JH Delay=12	JK Delay=6	new routing table for J	

Previous routing table will not be taken into consideration

→ Reaction to deteriorations

Overall Example - Distance-Vector Routing



Example: defining a section at (from) node J to B via …

A,I,H,J sends data to node B

	A	I	H	K	newly estimated delay starting at J	edge
B	12	36	31	28	20	A
	JA Delay =8	JI delay =10	JH delay =12	JK delay =6		

to B via A: costs (JA) + costs path (AB) = 8 + 12 = 20

to B via I: costs (JI) + costs path (IB) = 10 + 36 = 46

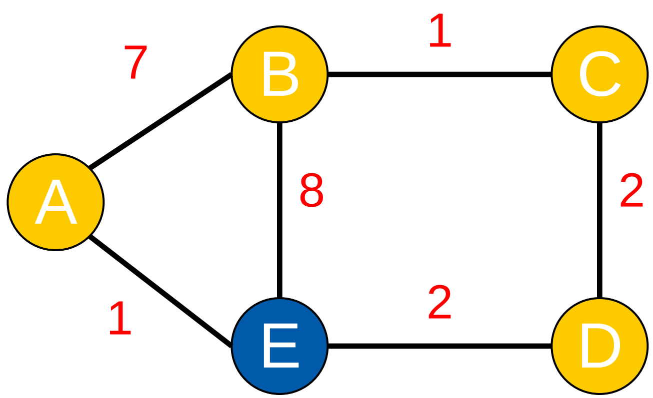
to B via H: costs (JH) + costs path (HB) = 12 +31 = 43

to B via K: costs (JK) + costs path (KB) = 6 + 28 = 34

seek for minimum: Min (JAB, JIB, JHB, JKB) = JAB = 20

7.2 Detailed Example - Distance-Vector Routing

Consider the following network:



$D^E()$	Next hop A	B	D
Destination A	1		
B			5
C			4
D			2

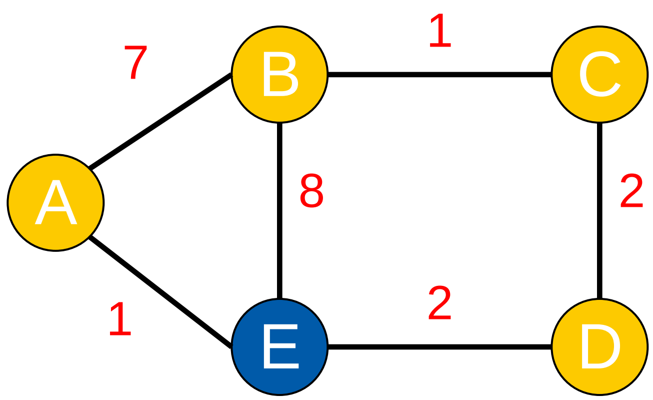
Node E

- Shall have the following routing table

from E: $D^E()$	via	dist.
to A	A	1
to B	D	5
to C	D	4
to D	D	2

Vector of node E

Distance Vector Routing:
Distance Vector based on Global Knowledge

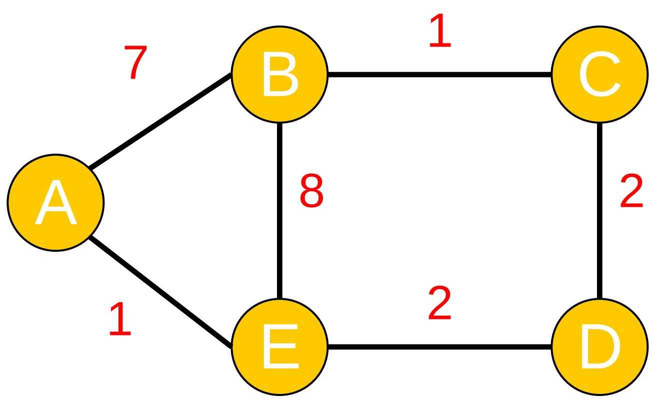


		Next hop		
	$D^E()$	A	B	D
Destination	A	1	14	5
	B	7	8	5
	C	6	9	4
	D	4	11	2

Notation:

- $D^X(Y, Z)$ is distance (cost) from node X to Y via Z,
  - Assuming that X first routes to Z and afterwards it takes the shortest path from Z to Y
  - $D^X(Y, Z)$ is entry on row Y, column Z in distance table of X
- C(X, Y) is cost of direct link from X to Y

Distance Vector Routing: Distance Vector based on Global Knowledge



		Next hop	
$D^E()$	**A**	**B**	**D**
Destination A	1	14	5
Destination B	7	8	5
Destination C	6	9	4
Destination D	4	11	2

Consider the first row, distances from E to A

- Cost from E to A via direct link is 1, $D^E(A, A) = 1$

- Cost from E to A, using D as first hop is 5, $D^E(A, D) = 5$
  - Cost from E to D is 2, minimum cost from D to A is 3 via E!
  - So, we go from E to D <u>and back to E?!? </u>This can become a problem…

- Cost from E to A, using B as first hop is 14, $D^E(A, B) = 14$

Red entries show which hop to use for which destination

- Note: We took a global view, now we do the same in decentralized way

Distance Vector Algorithm

1. Initialization

- $D^X(v, v) = C(X, v)$ if X and v are neighbors, ∞ otherwise
- Send shortest distances to all destinations to all neighbors

2. Update received or link cost to neighbor changed

- If link cost to neighbor V changed by d (positive or negative!), then $D^X(y, V) = D^X(y, V) + d$ for all destinations y
- If update $D^V(Y, w)$ received, then $D^X(Y, V) = C(X, V) + \min(D^V(Y, w))$
  - Minimum over all neighbors w of V, including X

3. Send routing updates periodically to all neighbors by broadcasting their entire routing tables

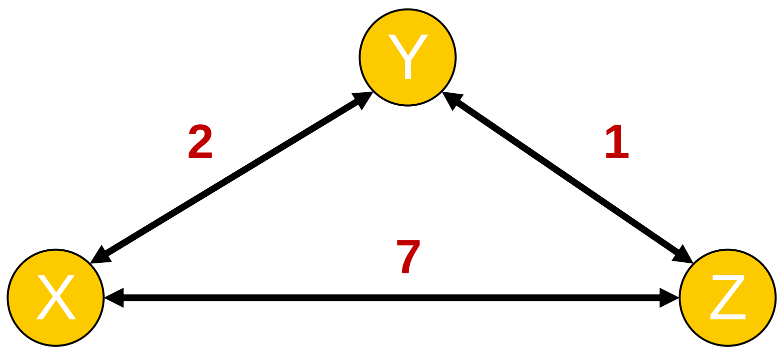
- New minimum could be result of a better route appearing or the old minimum getting worse
- Sometimes: If there is new minimum $D^X(Y, w)$ for any destination Y, then send new $D^X(Y, w)$ to all neighbors

7.3 Example: Initialization of Distance Tables and Sending Vector for 1st time

Example:

D^Y	X	Z
X	2	∞
Z	∞	1

Initialization of Distance Tables



Cost via

Dest. D^X	Y	Z
Y	2	∞
Z	∞	7

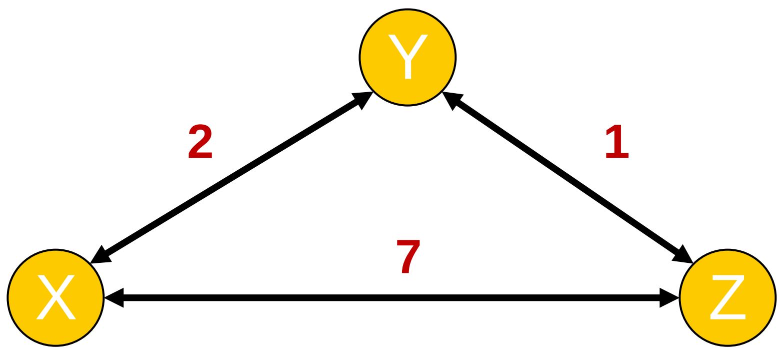
D^Z	X	Y
X	7	∞
Y	∞	1

Example: Sending Vector for 1st time

D^Y	X	Z
X	2	∞
Z	∞	1

Y	Sent to X	
(to X	X	2)
(to Z	Z	1)

Y	Sent to Z	
(to X	X	2)
(to Z	Z	1)



X	Sent to Y	
(to Y	Y	2)
(to Z	Z	7)

Z	Sent to Y	
(to X	X	7)
(to Y	Y	1)

Cost via

D^X	Y	Z
Y	2	∞
Z	∞	7

Dest.

X	Sent to Z	
(to Y	Y	2)
(to Z	Z	7)

Z	Sent to X	
(to X	X	7)
(to Y	Y	1)

D^Z	X	Y
X	7	∞
Y	∞	1

Example: Sending Vector (Refinement)

Each node sends "updates" to its neighbors

- Update message contains (node, distance) pairs which indicate that the sending node has "distance" hops to "node"

When another node receives update, it checks:

- If "announced distance + distance to the announcer" is less than current distance to "node", then use that route to "node"

Following updates are sent:

- X sends: (Y, 2), (Z, 7) to Y and Z
- Y sends: (X, 2), (Z, 1) to X and Z
- Z sends: (X, 7), (Y, 1) to X and Y

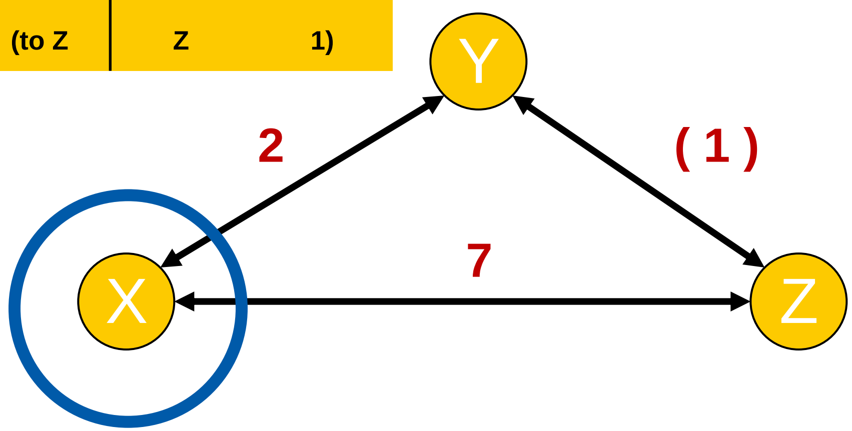
Note: If nodes are ordered, we can simply send a vector with shortest distances to all other nodes

7.4 Example: Updating the Table

E.g. node X here:

Example: Updating Table for the 1st time

Y	Sent to X	
(to X	X	2)
(to Z	Z	1)

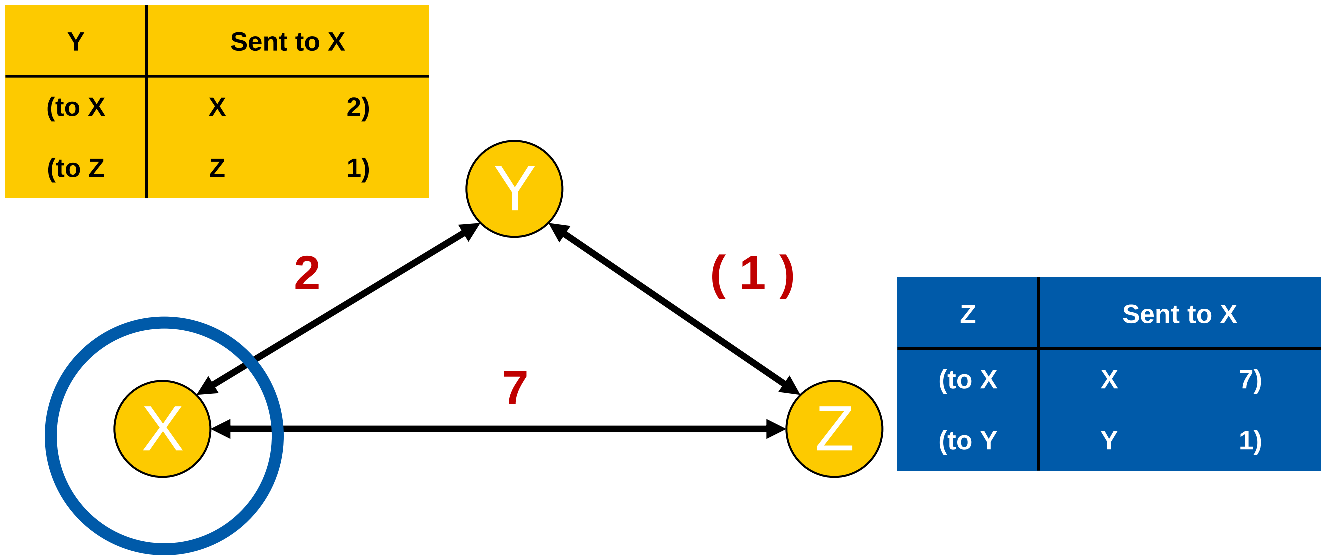


Z	Sent to X	
(to X	X	7)
(to Y	Y	1)

D^X	via Y	via Z
X to Y	2	∞ ?
X to Z	∞ ?	7

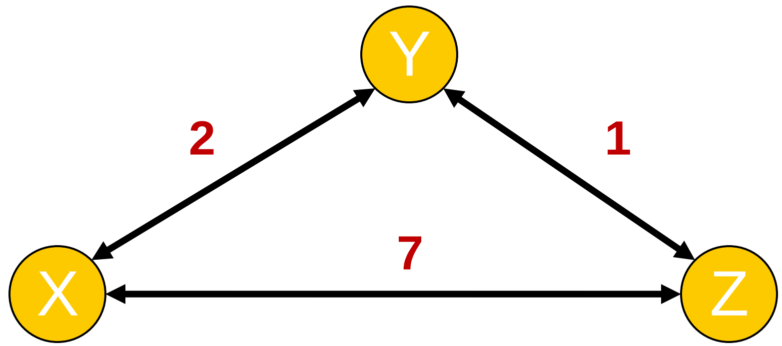
Example: Updating Table for the 1st time

E.g. node X here:



D^X	via Y	via Z
X to Y	2	Min (∞, 7+1) = 8
X to Z	Min (∞, 2+1) = 3	7

Example: Updating Table for the 1^{st} time



After all nodes sent vectors to neighbors:

Cost via

Dest.

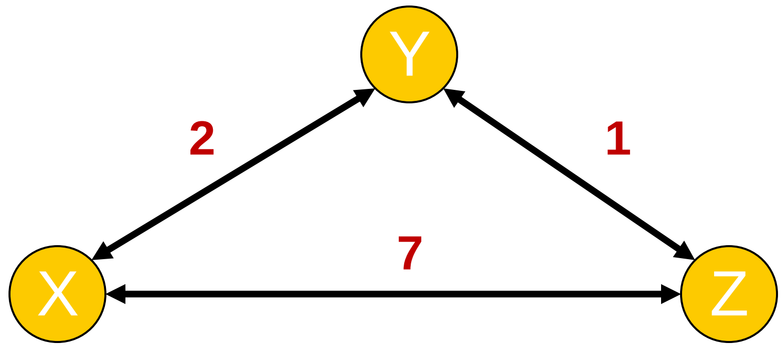
D^X	Y	Z
Y	2	8
Z	3	7

D^Y	X	Z
X	2	8
Z	9	1

D^Z	X	Y
X	7	3
Y	9	1

Current minimum cost New minimum cost

Example: Sending Vector for the 2nd Time



X, Y and Z sent vectors

- Note: The vector sent by Y does not provide new information

No changes anymore after these updates

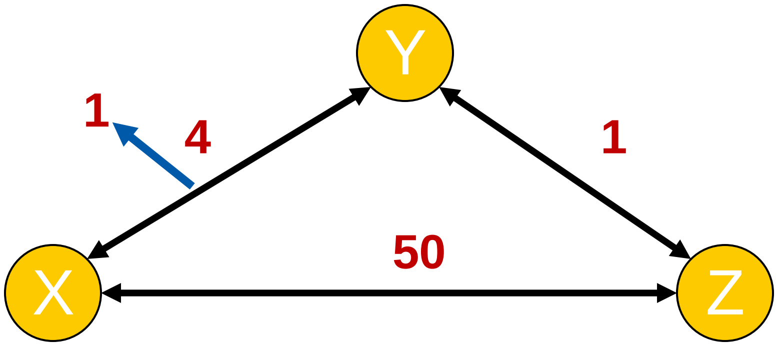
Cost via

	D^X	Y	Z	D^Y	X	Z	D^Z	X	Y
Dest	**Y**	2	8	**X**	2	8	**X**	7	3
	Z	3	7	**Z**	9	1	**Y**	9	1

Current minimum cost New minimum cost

TECHNISCHE UNIVERSITÄT DARMSTADT

Consider following network:



What happens when cost of link XY goes to 1?

- We consider only distances from Y and Z to X

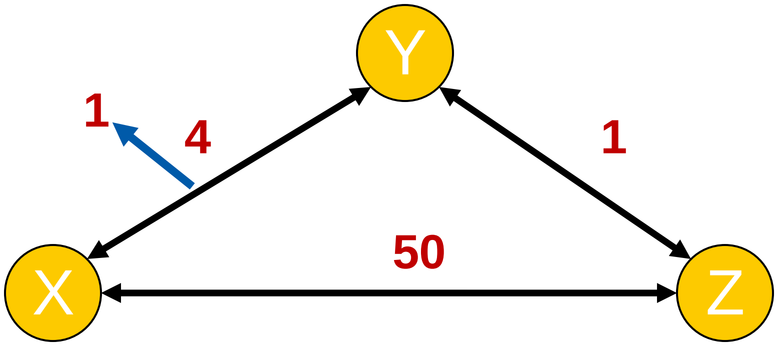
Before change, assume that the rows for X in distance tables on Y and Z look like this:

D^Y	X	Z
X	4	6

D^Z	X	Y
X	50	5

Distance Vector: Link Cost DEcreases

Consider following network:



What happens when cost of link XY goes to 1?

- We consider only distances from Y and Z to X

Before change, assume that the rows for X in distance tables on Y and Z look like this:

D^Y	X	Z
X	4	6

D^Z	X	Y
X	50	5

Distance Vector: Link Cost DEcreases

With periodic sending of the vector, Y also sends update to Z

- E.g. Y detects reduction of size of link first
- Changes are computed and sent

D^Y	X	Z
X	1	6

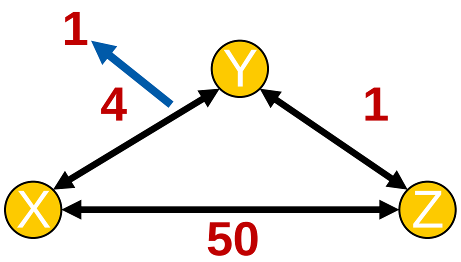
D^Z	X	Y
X	50	5

D^Y	X	Z
X	1	6

D^Z	X	Y
X	50	2

D^Y	X	Z
X	1	3

D^Z	X	Y
X	50	2



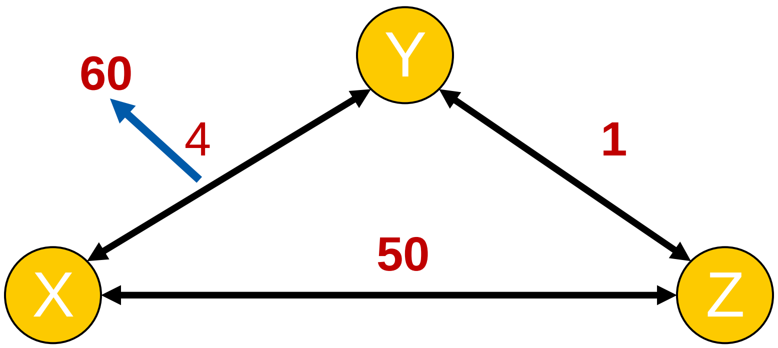
Information about lower cost propagates very fast, only two iterations needed

"Good news travels fast"

Distance Vector: Link Cost INcreases

TECHNISCHE UNIVERSITÄT DARMSTADT

Consider following network:



What happens when cost of link XY goes up to 60?

- We consider only distances from Y and Z to X
- Before change, the rows for X in distance tables on Y and Z look like this:

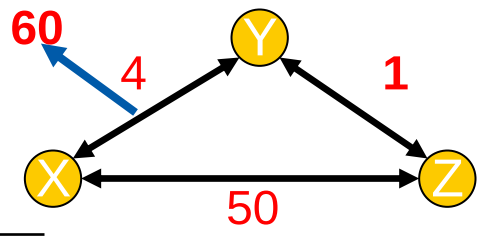
D^Y	X	Z
X	4	6

D^Z	X	Y
X	50	5

Distance Vector: Link Cost INcreases

With periodic sending of the vector, Y also sends update to Z

- E.g. Y detects increase of size of link first and
- Changes are computed and sent



D^Y	X	Z
X	60	6

D^Z	X	Y
X	50	5

D^Y	X	Z
X	60	6

D^Z	X	Y
X	50	7

D^Y	X	Z
X	60	8

D^Z	X	Y
X	50	7

D^Y	X	Z
X	60	8

D^Z	X	Y
X	50	9

Keep on sending (updates) for many (44) iterations

Finally cost from Z to X via Y will become larger than the direct cost of 50 and updates do not improve the result

"Bad news travels slowly"

- Updates about increased cost travel very slowly

This problem is known as "count-to-infinity" problem

Potential solution: Poisoned Reverse Algorithm

- Another solution: Split Horizon

Distance Vector Routing – Property “Count to Infinity”

Information distribution over new

- short paths (with few hops) & Route improvement

  → FAST
- long paths (with many hops) & Route deterioration

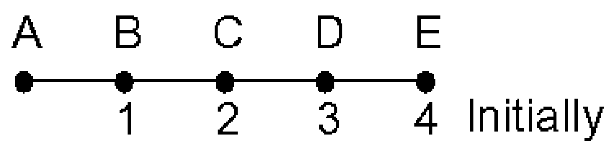
  → SLOW

Route improvement

- previously:
  - A unknown
- later:
  - A connected with distance 1 to B, this will be announced
- Note:
  - Synchronous update used here for simplification
  - distribution proportional to topological spread

→ fast

Example:



A	B	C	D	E	
	∞	∞	∞	∞	**Initially**
	1	∞	∞	∞	**After 1 exchange**
	1	2	∞	∞	**After 2 exchanges**
	1	2	3	∞	**After 3 exchanges**
	1	2	3	4	**After 4 exchanges**

Distance Vector Routing – Property “Count to Infinity”

Route deterioration

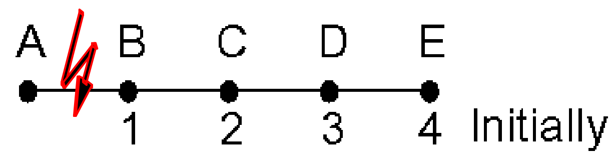
- here: connection destroyed
- A previously known, but now detached
- the values are derived from (incorrect) connections of distant IS

Comment

- limit “infinite” to a finite value, depending on the metrics
  - example: "infinite = maximum path length + 1“

→ slow

Example:



A_1	B_1	C_1	D_1	E_1	
	1	**2**	**3**	**4**	**Initially**

B: no connection directly to A,
but C reports distance CA=2
i. e. BA = BC+ CA = 1 + 2 = 3
actually wrong!

	3	**2**	**3**	**4**	**After 1 change**
	3	**4**	**3**	**4**	**After 2 changes**
	5	**4**	**5**	**4**	**After 3 changes**
	5	**6**	**5**	**6**	**After 4 changes**
	7	**6**	**7**	**6**	**After 5 changes**
	7	**8**	**7**	**8**	**After 6 changes**

...

	∞	∞	∞	∞	

Problem with count-to-infinity

Y tries to route to X via Z and

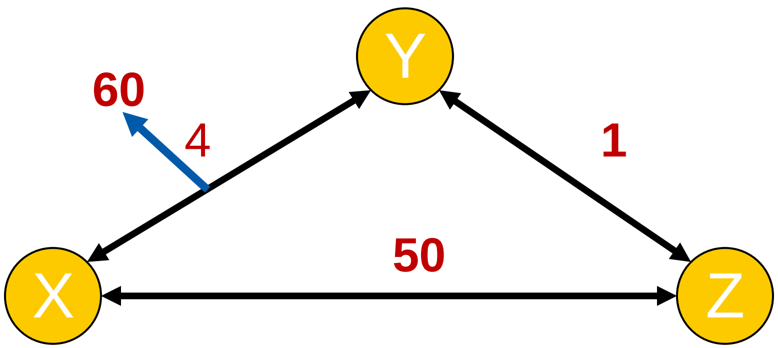
Z believes best path to X goes via Y

- Routing loop

Idea behind poisoned reverse:

- If node Z routes through Y to get to X,
  then Z advertises to Y that its distance to X is infinity
- Everything else works normally

Example:



Distance Vector: Poisoned Reverse - Link Cost Changed

When link cost changes, Y sends update to Z

D^Y	X	Z
X	4	∞

D^Z	X	Y
X	50	5

D^Y	X	Z
X	60	∞

D^Z	X	Y
X	50	61

D^Y	X	Z
X	60	51

D^Y	X	Z
X	60	51

D^Z	X	Y
X	50	∞

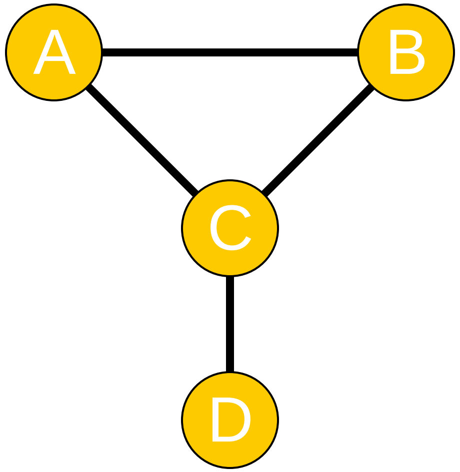
Distance Vector: Poisoned Reverse - Discussion

**Poisoned Reverse solved the problem,
only a few iterations were needed to converge to a stable state**

**But: In a larger network,
Poisoned Reverse does not solve count-to-infinity problem :-(**

Consider the following network:

- If link CD is removed, then:
  - A receives bad information from B
  - B receives bad information from A
  - Count-to-infinity problem persists



7.7 Distance Vector Routing: Split Horizon

Split Horizon is a similar algorithm

Idea:

- Do not advertise routes through a neighbor that sent the update

Comparison:

- Poisoned Reverse sends the updates,
  but sets the distance to infinite

Terminology confusion:

- Split Horizon
- Poisoned Reverse
- Split Horizon with Poisoned Reverse

Last two are the same thing

Distance Vector Routing Variant "Split Horizon Algorithm"

Improvement:

- to improve the "count to infinity" property
- Objective - based on the Distance Vector principle

Principle

- in general, to publicize the "distance" to each neighbor
- special case:
  - if neighbor Y exists on the reported route,
  - then X reports the response "false" to Y

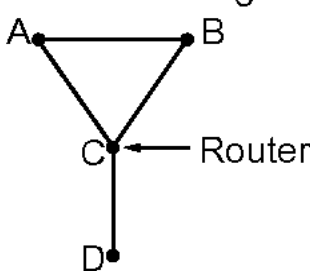
distance X (via Y) according to arbitrary i: ∞

Example: deterioration,
e.g. connection destroyed

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

Note:
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)



7.8 Distance Vector: Summary

Distance vector is a simple algorithm

Works in a fully decentralized manner

Suffers from count-to-infinity problem

- Solvable by Poisoned Reverse/Split Horizon only in small networks

comparison to link state

- Neither algorithm is a winner over the other
- Both distance vector and link state are being used in the Internet

Some other routing algorithms:

- Hot potato routing: Forward packet as quickly as possible
- Traffic matrix: Assign traffic flows to network links
  - Requires identification of traffic flows

8 Adaptive Distributed – Link State Routing

also "distributed routing"

Basic principle

IS

- measures the "distance" to the directly adjacent IS,
- distributes information,
- calculates the ideal route

Use

- introduced into the ARPANET in 1979, nowadays most prevalent
- IS-IS (Intermediate System-Intermediate System)
- OSPF (Open Shortest Path First)
  - since 1990 Internet RFC 1247

Procedure

1.

- To determine the address of adjacent IS

2.

- To measure the "distance" (delay, ...) to neighbor IS

3.

- To organize the local link state information in a packet

4.

- To distribute the information to all IS

5.

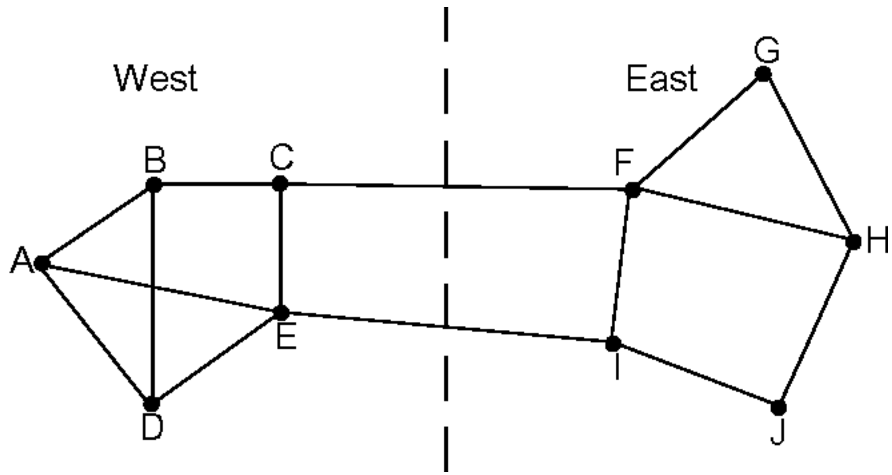
- To calculate the route based on the information of all IS

Link State Routing - 1. Phase

1. Phase:
to 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

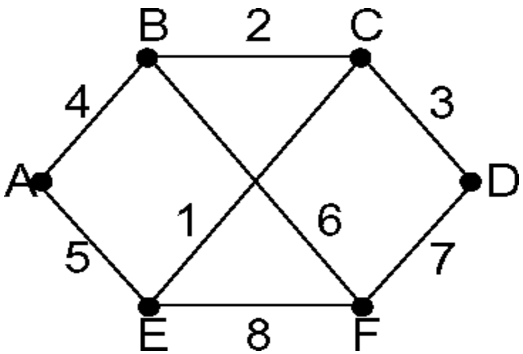
Link State Routing - 2. Phase



2. Phase:
to define the "distance"

- distance is generally defined as delay
- detection via transmission of ECHO messages, which are reflected at receiver
- multiple transmission:
  - improved average value
  - with or without payload:
    - with payload is usually better,
  - but "with load" may lead to an "oscillation" of the load:
  - after each new routing table the other link CF or EI is charged

Link State Routing - 3. Phase



Link State Packets:

A		B		C		D		E		F	
Seq.		Seq.		Seq.		Seq.		Seq.		Seq.	
Age		Age		Age		Age		Age		Age	
B	4	A	4	B	2	C	3	A	5	B	6
E	5	C	2	D	3	F	7	C	1	D	7
		F	6	E	1			F	8	E	8

3. Phase:
to organize 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 Routing - 4. Phase

4. Phase:
to distribute 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

Link State Routing – 5. Phase

5. Phase:
to compute new routes

- each IS for itself
- possibly larger amount of data available
- Making use of e.g. Dijkstra’s algorithm

Multipath Routing

Hierarchical Routing

Routing

- Mobility
- Security

9.1 Multipath Routing

Principle

- using alternative routes between the IS pairs
- frequency of usage depends on the quality of the alternative
- higher throughput due to the data traffic being distributed to various paths
- increased reliability

Implementation

- each IS contains a rating table including
- one row for each possible destination IS

Z	A_1	G_1	A_2	G_2	…	A_n	G_n

Z … destination

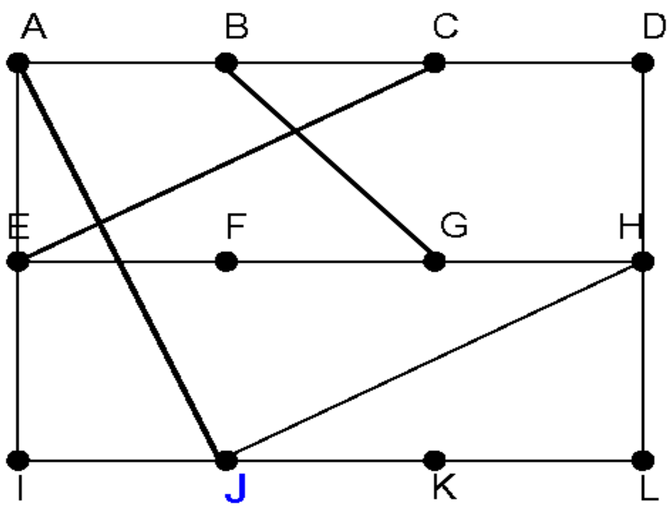
A_i … i-best outgoing line

G_i … weight for A_i

G_i determines the probability with which A_i will be used:

$$\left(\sum_{i=1}^{n} G_i = 1 \right)$$

Multipath Routing



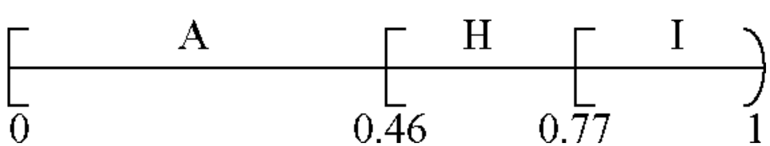
dest.	1st choice		2nd choice		3rd choice	
A	**A**	**0.63**	**I**	**0.21**	**H**	**0.16**
B	**A**	**0.46**	**H**	**0.31**	**I**	**0.23**
C	**A**	**0.34**	**I**	**0.33**	**H**	**0.33**
D	**H**	**0.50**	**A**	**0.25**	**I**	**0.25**
E	**A**	**0.40**	**I**	**0.40**	**H**	**0.20**
F	**A**	**0.34**	**H**	**0.33**	**I**	**0.33**
G	**H**	**0.46**	**A**	**0.31**	**K**	**0.23**
H	**H**	**0.63**	**K**	**0.21**	**A**	**0.16**
I	**I**	**0.65**	**A**	**0.22**	**H**	**0.13**
K	**K**	**0.67**	**H**	**0.22**	**A**	**0.11**

Example:

Table from J →

Selecting the alternatives: i.e., generating a random number z (0 ≤ z < 1)

Example: destination B



Above, our network was flat

- All routers were identical and knew about all other routers

Two problems with this approach:

It does not scale

- Internet has millions of hosts and devices
  not feasible to have one entry per host in routing table
- Routing tables become
  - too large and
  - algorithms never converge

No administrative autonomy

- An organization should be able to run its own network the way they want and still be able to connect to “outside” networks

Solution: Hierarchical Routing and Autonomous Systems (AS)

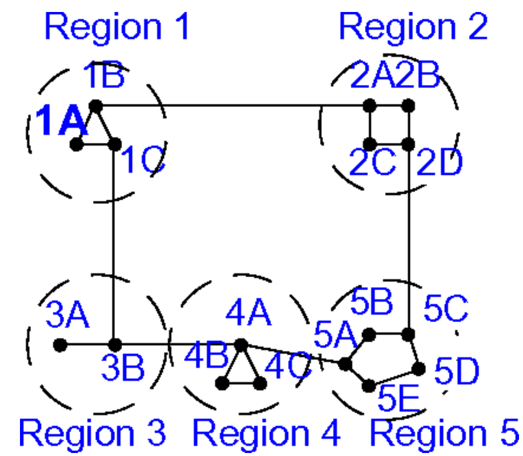
Hierarchical Routing

Motivation

- a large number of IS means
  - time-consuming dynamic routing calculation
  - storage of very large routing tables

→ **hierarchical structure**

- reduces individually treated IS



Example (of 2 tables)

Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2A	1B	2
2B	1B	3
2C	1B	3
2D	1B	4
3A	1C	3
3B	1C	2
4A	1C	3
4B	1C	4
4C	1C	4
5A	1C	4
5B	1C	5
5C	1B	5
5D	1C	6
5E	1C	5

Comparison

- But the best path is not always
- Calculated design: number of layers

Hierarchical table for 1A

Dest.	Line	Hops
1A	-	-
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4

Overlay networks have become extremely popular

- All peer-to-peer file sharing networks are overlay networks

Overlay network is simply a virtual network topology overlaid over a real network

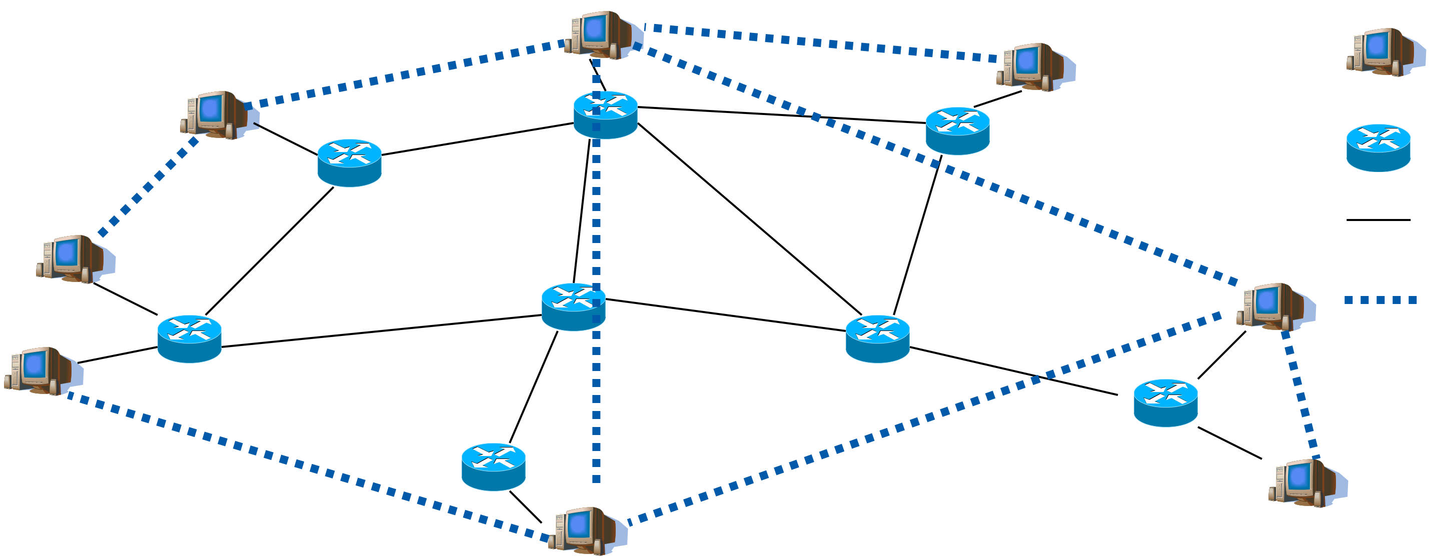
Neighbors in overlay are not (necessarily) neighbors in underlying network

For example:

- An application is overlaid on top of the P2P network, …
- P2P network is overlaid on top of IP
- Internet (IP) is overlaid over the physical links
  - Neighboring nodes in IP might not be directly connected
- At the bottom are the real network links (fiber and cable)

Overlay Network

Typically overlay network means a virtual network on top of the underlying IP network



Routing in Overlay Networks

Which kind of routing algorithms can be used in overlay networks?

Answer:
Exactly the same as in IP networks :-)

When node A sends a message to its overlay neighbor B:

- The message gets routed over the Internet from A to B
  - Takes possibly several hops through Internet routers
- From overlay point of view, message takes only one hop

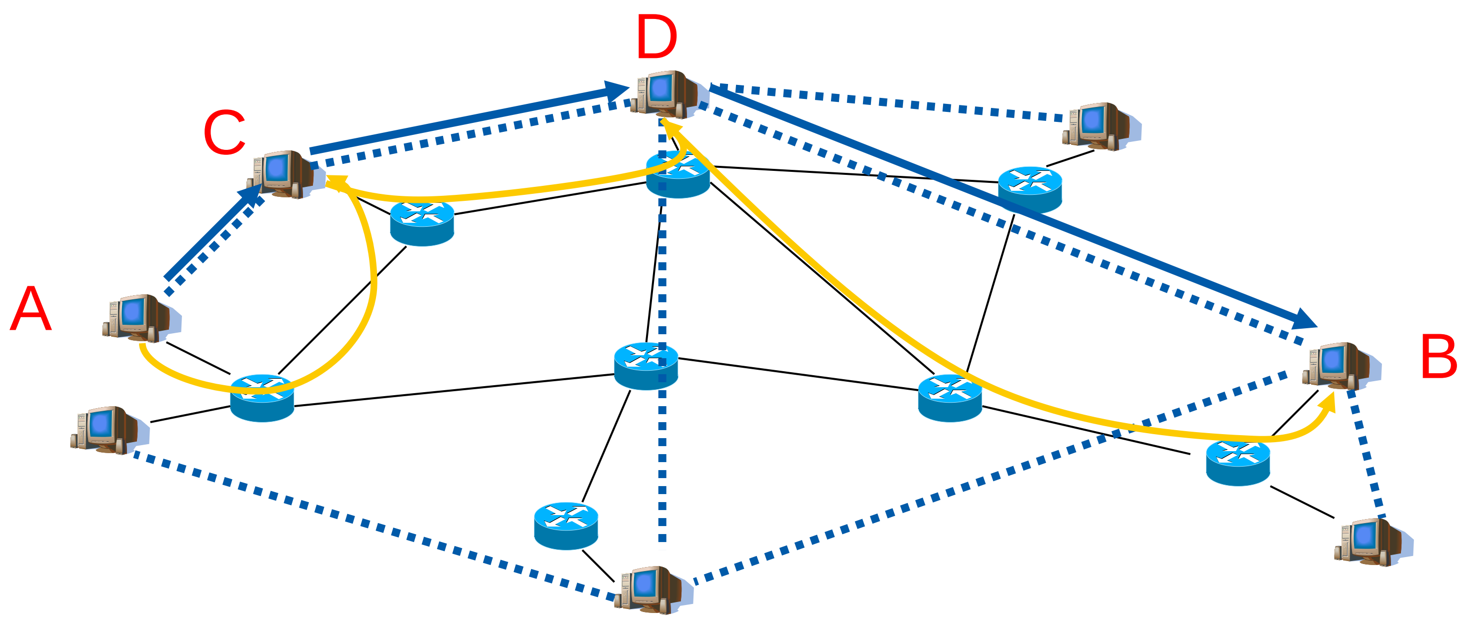
Two important metrics for overlay routing:

- How many hops in overlay?
  - Same as for traditional routing algorithms
  - Goal:
    to minimize hops in overlay
- How many hops in underlying network?
  - Routing in overlay from A to B to C is (likely) longer than standard Internet path from A to C (**stretch**)
  - Second goal:
    to minimize stretch (to the extent that this is possible)

TECHNISCHE UNIVERSITÄT DARMSTADT

A wants to send message to B

- Shortest path in overlay is A, C, D, B
  - A to C is 3 (IP) hops, C to D is 3 hops, D to B 4 hops, total 10 hops
  - Shortest direct IP path from A to B is 5 hops
  - Stretch factor is in this case 2



Routing is essential for communications

Routers have two main functions

- **Routing**: Determine which route to use - Algorithms
- **Forwarding**: What happens when a packet arrives?

Algorithms

- Non-adaptive
- Adaptive like distance vector

TECHNISCHE UNIVERSITÄT DARMSTADT

Two different classification schemes for routing algorithms

- Global vs. decentralized
- Dynamic vs. static
- 4 different classes of routing algorithms

Non-Adaptive (static) algorithms

- Routes change very slowly, typically only manually changed
- E.g. (Dijkstra) Spanning tree, Flooding

Adaptive (dynamic) algorithms

- Dynamic algorithms react to changes in network topology and traffic conditions
- Can be run periodically or whenever there is a change
- Dynamic algorithms are more responsive, but also harder to design and implement
- E.g. Link State routing, Distance Vector Routing, …

Global routing algorithm

- Knows all the nodes and links in the network (global view)
- Algorithm can be run in a single place (centralized), or in several place (but still with complete information)
- Known as link state algorithms

Decentralized routing algorithm

- Best path is calculated in an iterative, distributed manner
- No node knows everything about the network
- Nodes exchange information with neighbors and slowly learn routes to other nodes
- Known as distance vector algorithms

Internet uses two kinds of routing algorithms

- Dynamic, global link state algorithm
- Dynamic, decentralized distance vector algorithm