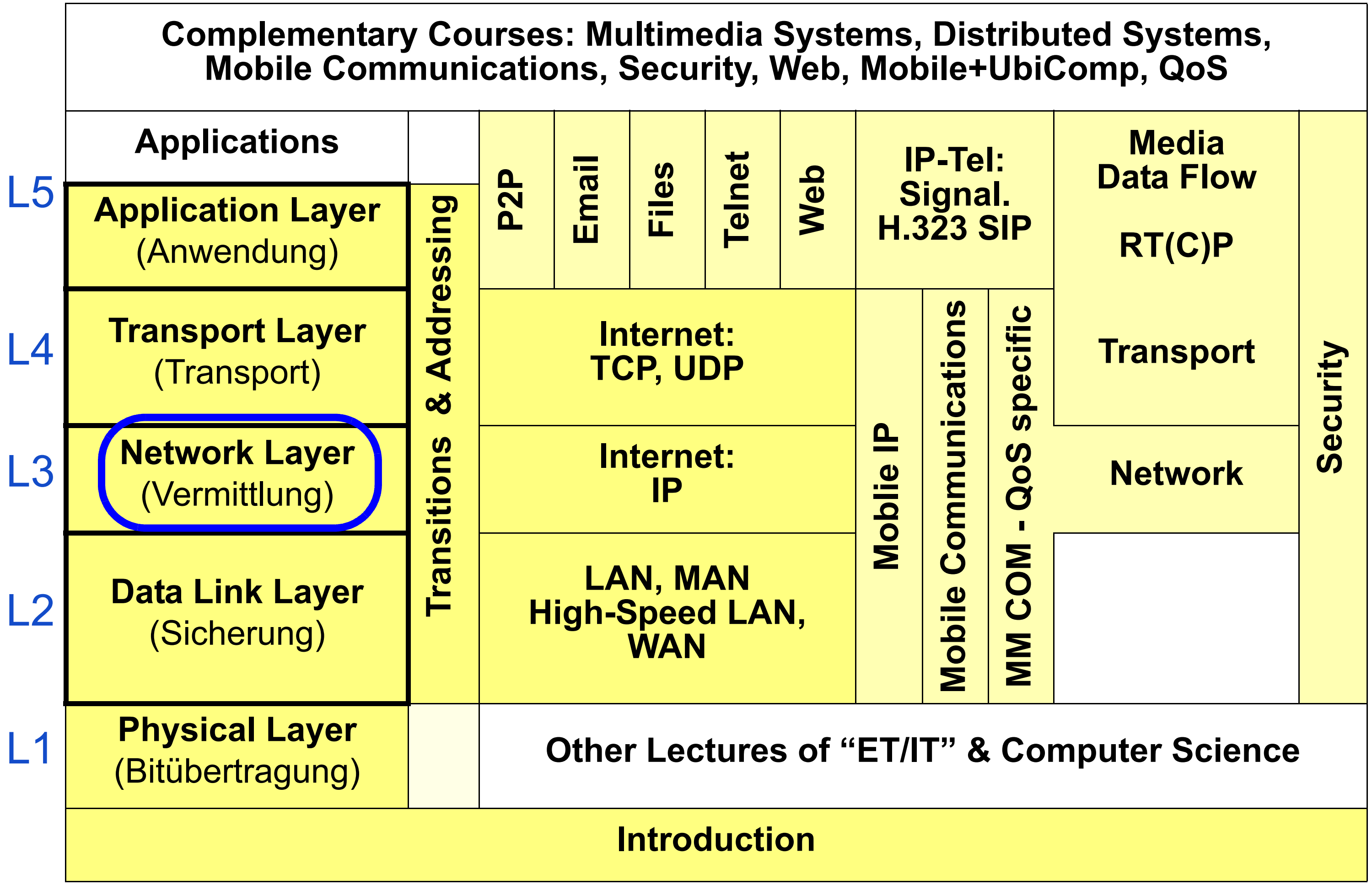
Communication Systems
Network Layer

Prof. Dr.-Ing. **Lars Wolf**

TU Braunschweig
Institut für Betriebssysteme und Rechnerverbund

Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
Email: wolf@ibr.cs.tu-bs.de

Scope



Complementary Courses: Multimedia Systems, Distributed Systems, Mobile Communications, Security, Web, Mobile+UbiComp, QoS

Applications

L5 **Application Layer** (Anwendung)

L4 **Transport Layer** (Transport)

L3 **Network Layer** (Vermittlung)

L2 **Data Link Layer** (Sicherung)

L1 **Physical Layer** (Bitübertragung)

Transitions & Addressing

P2P

Email

Files

Telnet

Web

IP-Tel: Signal. H.323 SIP

Media Data Flow

RT(C)P

Transport

Network

Internet: TCP, UDP

Internet: IP

LAN, MAN High-Speed LAN, WAN

Mobile IP

Mobile Communications

MM COM - QoS specific

Security

Other Lectures of “ET/IT” & Computer Science

Introduction

Overview

1. Functions of the Network Layer

2. Switching Approaches

3. Services

4. Routing
   Non-adaptive Procedures
   Adaptive Procedures
   Extensions

5. Broadcast Routing

6. Multicast Routing

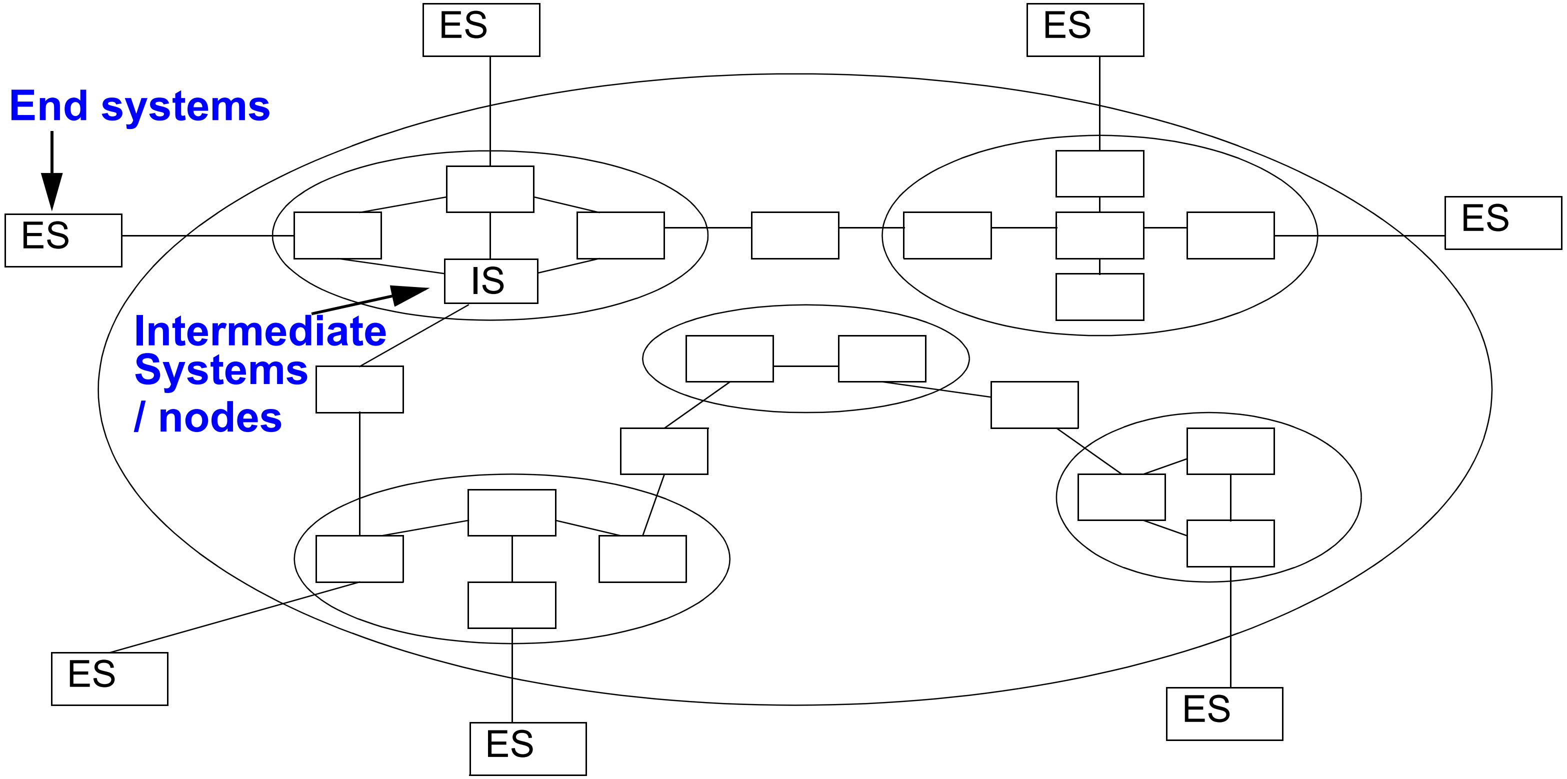
7. Congestion Control

8. Addressing

1. Functions of the Network Layer

Data transfer from end system to end system

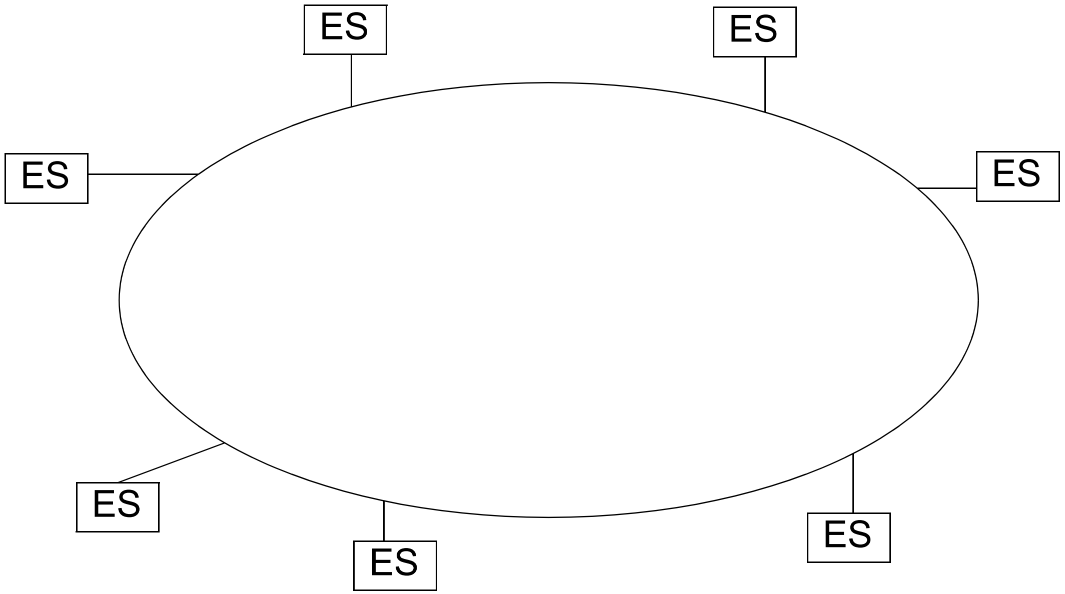
- **several hops**
- **(heterogeneous) subnetworks**
- **compensate for differences between end systems during transmission**



Relevance of the interface: switching vs. transport service

- **L1 up to + incl. L3:** **organization: carrier**
- **from L4 onward:** **user/customer/company/institute**

Functions of the Network Layer (2)



The provided services are

- **standardized for end systems**
- **independent from network technology**
- **independent from number, type and topology of the subnetworks**

Subnetworks (IS 7498):

> **A multiple of one or several intermediary sytems that provide switching functionalities and through which open end systems can establish network connections**

Functions of the Network Layer (3)

Primary tasks

- virtual circuits or datagram transmissions
- routing
- congestion control
- Internetworking: provide transitions between networks
- addressing
- Quality of Service (QoS)

Secondary tasks, based on type service and request:

- multiplexing of network connections
- fragmentation and reassembling
- error detection and correction
- flow control as a means to handle congestion
- maintaining the transmission sequence

Functions of the Network Layer (4)

Required knowledge

- **subnetwork topology**
- **address / localization of the end system**
- **network status (utilization,...)**
- **packet / data stream communication requirements (Quality of Service)**

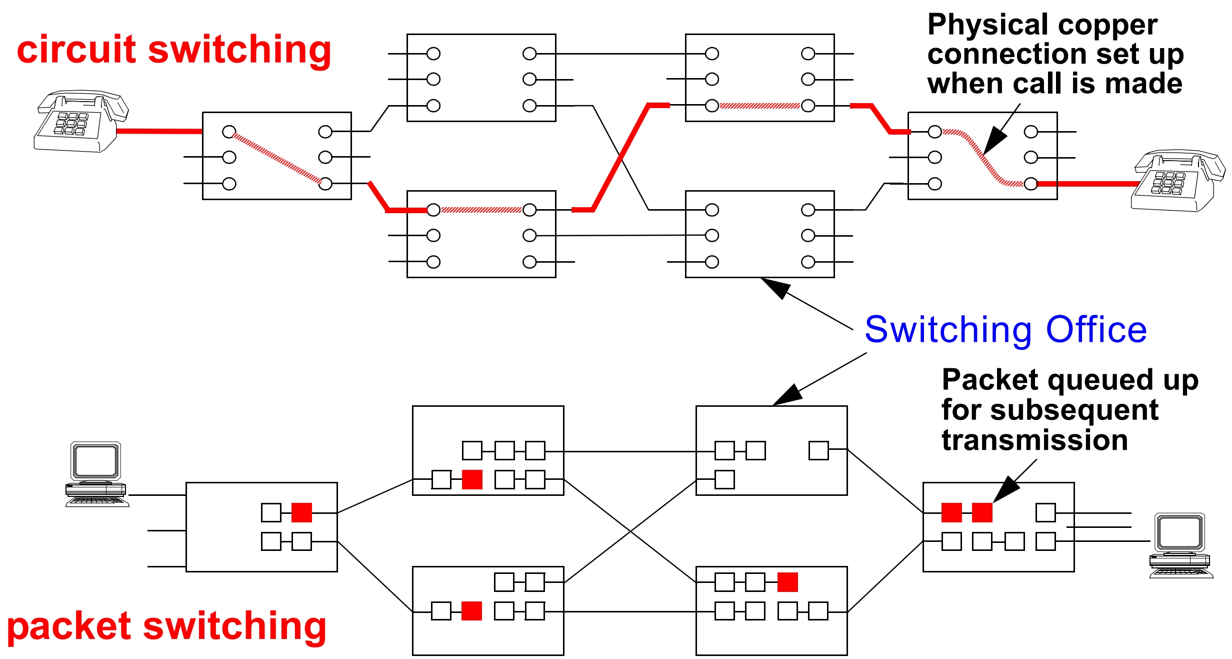
Examples

- **X.25 (ISDN, ...)**
- **Internet protocol IP (TCP/IP,..)**

Nomenclature:

Layer	**Data Entity**
Transport	...
Network	**Packet**
Data Link	Frame
Physical	Bit/Byte (bit stream)

2. Switching Approaches


circuit switching
Physical copper connection set up when call is made
Switching Office
Packet queued up for subsequent transmission
packet switching

Circuit switching

- switching a physical connection

Message switching

- message is stored and passed on by one hop

Packet switching

- store-and-forward, but transmission packets limited in size

Switching by virtual circuit

- packets (or cells) over a pre-defined path

Circuit Switching

Principle:

- **dedicated path from source to destination for entire duration of a call**
  - connections between switching centers (frequency spectrum, dedicated ports)

Implementation examples:

- **historically: on switching boards**
- **mechanical positioning of the dialers**
- **setting coupling points in circuits**
- **early alternative of B-ISDN: STM (Synchronous Transfer Mode)**

Properties:

- **connection has to be setup before transmission**
  - establishing a connection takes time
- **fixed allocation of bandwidth ⇒ no congestion during transfer**
- **constant delay**
  - No processing of data at intermediate nodes ⇒ short delay
- **information delivery is sequenced (by nature)**
- **resource allocation too rigid (possibly wastage)**
  - No support for transmission of bursty data ⇒ potential resource underutilization
- **once connection is established it cannot be blocked anymore**

Message Switching

Principle:

- **all data to be sent is treated as a "message"**
- **"store and forward" network:
  in each node the message is**
    1. accepted,
    2. checked for errors,
    3. stored and
    4. forwarded (as a whole to the next node)

Example:

- **first telegram service**

Properties:

- **high memory requirements at each node (switching centers)**
    - because message may be of any size
    - usually stored on secondary repository (harddisk)
- **node may be used completely (whole capacity) over a long period of time by one message**
    - i.e., better if packets are of limited size (packet switching)

Packet Switching

Principle:

- **packets of limited size**
- **dynamic determination of route for every packet**
- **no dedicated path from source to destination**

Properties:

- **no connect phase**
- **dynamic allocation of bandwidth**
    - suitable for bursty traffic
    - flexible, provides for resource sharing and good utilization
- **congestion possible**
- **bandwidth reservation difficult, QoS provisioning limited**
- **variable end-to-end delay**
    - due to queuing at intermediate nodes (and varying routes)
- **information delivery may not be sequenced or reliable**

Example:

- **Internet**

Virtual Circuit Switching

Principle:

- **setup path from source to destination for entire duration of call**
- **using state information in nodes but no physical connection**
- **connection setup: defines data path**
- **messages: as in packet switching**
    - follow all **ONE** path
    - but (may) have only the address of the network entry point
        - not the destination address, e.g., ATM: VPI/VCI

Examples:

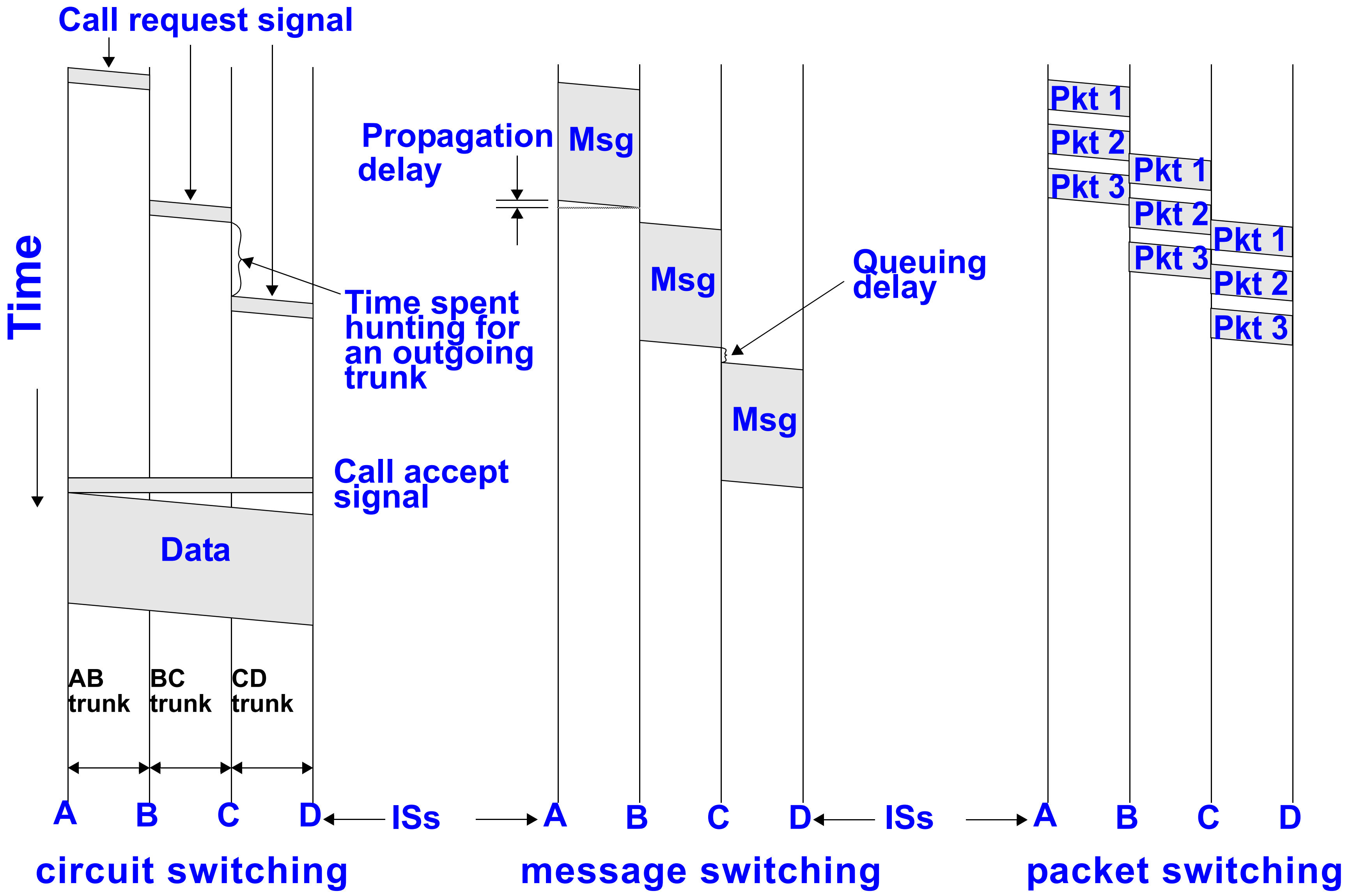
- **ATM PVC (permanent virtual circuit)**
    - established "manually" (similar to dedicated lines)
- **ATM SVC (switched virtual circuit)**
    - signaling: connect and disconnect corresponding to the telephone network
- **Internet Integrated Services**
    - state established via signaling protocol (RSVP)
    - full addresses are used

Properties

- **all messages of a connection are routed over the same pre-defined data path, i.e., sequence is maintained**
- **it is easier to ensure Quality of Service (see also ATM)**

Comparison: Temporal Performance

Timing of events:



Comparison: Circuit and Packet Switching

Circuit switching:

- **connection establishment can take a long time**
- **bandwidth is reserved**
  - no danger of congestion
  - possibly poor bandwidth utilization (bursty traffic)
- **continuous transmission time, because all data is transmitted over the same path**
- **price calculation:**
  - duration of connection

Packet switching:

- **connect phase not (absolutely) necessary**
- **dynamic allocation of bandwidth**
  - danger of congestion
  - optimized bandwidth utilization
- **varying transmission time:**
  - because packets of a connection may use different paths
  - not suitable for isochronous data streams
- **price calculation:**
  - transfer volume

Switching Approaches: Applicability

Circuit switching:

- **telephone system**
- **until now minor usage for computer networks, but various multimedia applications require isochronous data streams**

Packet switching:

- **used frequently for computer networks**
- **difficult for voice transmissions but with dominance of Internet (and VoIP) getting importance also here**

Message switching:

- **seldomly used for computer systems**
  - complex storage management (secondary storage)
  - blockage because of large messages

Virtual circuit switching:

- **important for QoS provisioning (perhaps in modified manner)**
- **integrated services**
- **voice transmission**

3. Services

Concepts

- Connection-oriented vs. connectionless communication

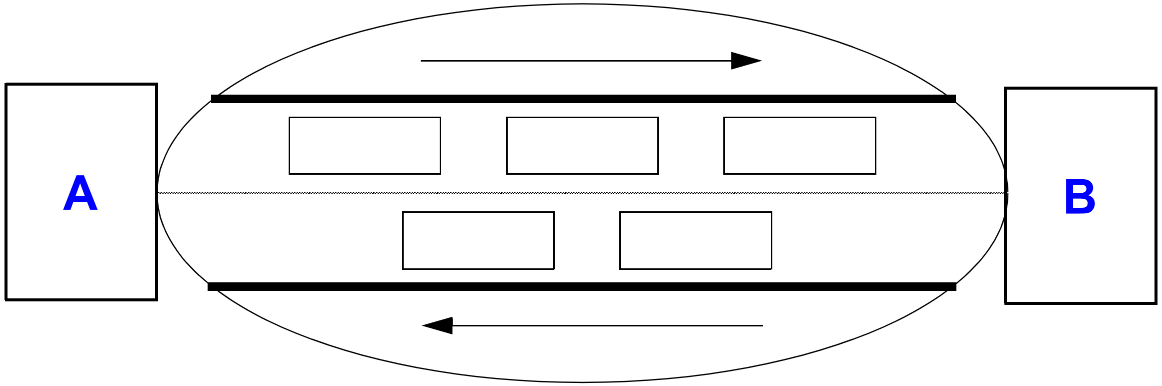
Connection-oriented

- goal: error free communication channel
- usually error control: L3 (or network)
    - flow control, ...
- duplex communication
- has advantages for realtime communications
- typical approach from telephone and telecommunication companies:
    - X.25, ATM

Connectionless

- unreliable communication
- hardly any error control: left to L4 or higher layers
    - sequence not ensured, ...
- simplex communication
- more favourable for simple data communication:
    - SEND-PACKET, RECEIVE-PACKET
- Internet community: IP

Connection-Oriented Communication



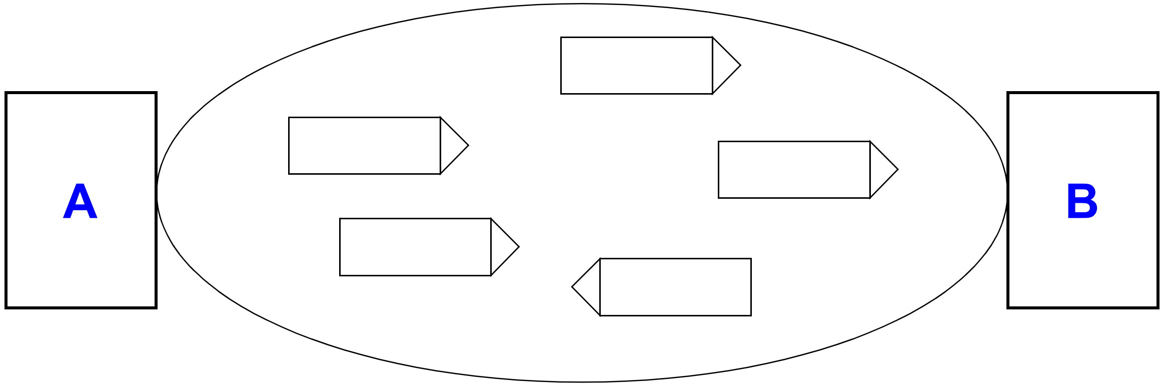
Properties:

- **3-phase interaction**
  - connect
  - data transfer
  - disconnect
- **(allows for) QUALITY OF SERVICE NEGOTIATION (e.g., throughput, error probability, delay)**
- **(typically) RELIABLE COMMUNICATION in both directions**
  - no loss, no duplicates, no modification
  - ensures maintainance of the correct sequence of transmitted data
- **FLOW CONTROL**
- **relatively complex protocols**

Example:

- **telephone service**

Connectionless Communication



Properties:

- **network transmits packets as ISOLATED UNITS (datagram)**
- **UNRELIABLE COMMUNICATION:**
  - loss, duplication, modification, sequence errors possible
- **no flow control**
- **comparatively SIMPLE PROTOCOLS**

Example:

- **mail delivery service**

Comparison of Concepts

Arguments pro connection-oriented service:

- **simple, powerful paradigm**
- **allows for simplification of the upper layers (L4 - L7)**
- **simplifies task of end systems**
- **for some applications efficiency in time is more important than error-free transmission**
  - (e.g. realtime applications, digital voice transmission)
- **suitable for a wide range of applications**

Arguments pro connectionless service:

- **high flexibility and low complexity**
- **avoids high costs for connects and disconnects for transaction-oriented applications**
- **easier to optimize the network load**
- **compatibility and costs: IP common**
- **"END-TO-END ARGUMENTS" (Saltzer et al.):**
  - reliable communication requires error control within the application
  - and: error control in one layer can replace the error control in the layer underneath it

3.1 Layer 3 Services and their Implementations

ISO IS 8348 Network Service Definition

2 Service classes:

- **Connection-Oriented Network Service (CONS)**
- **Connection-Less Network Service (CLNS)**

Implementations:

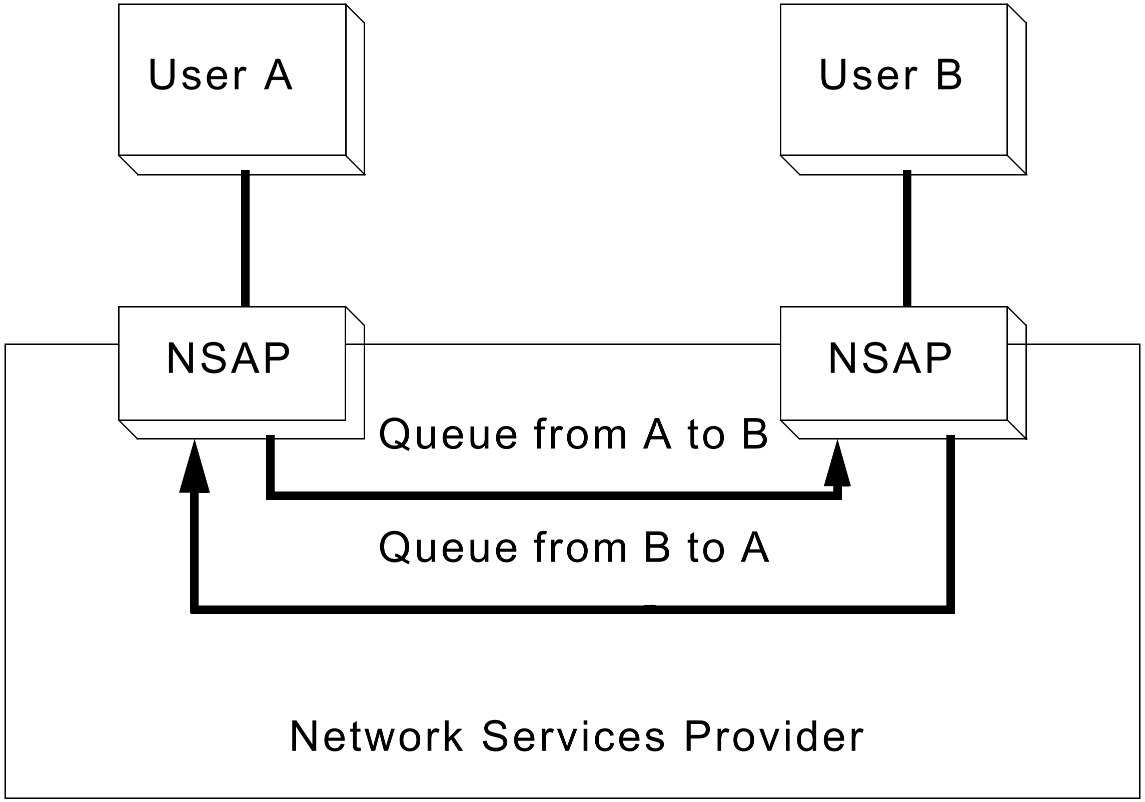
- **virtual circuit**
- **datagram**

Comment: service not equal to implementation!

Examples for communication architectures:

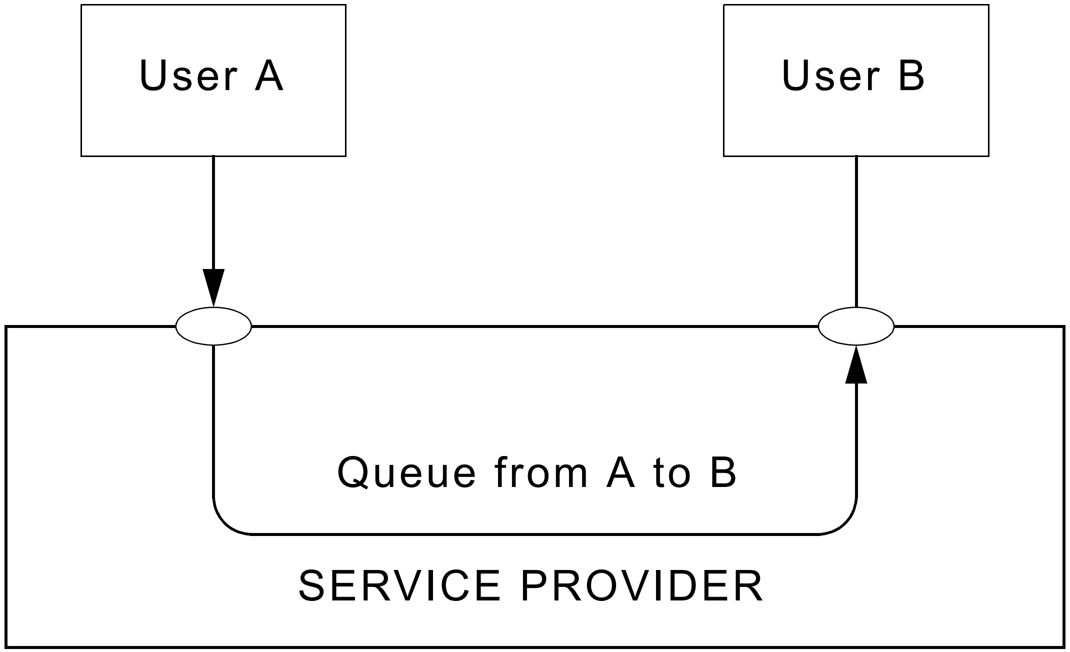
		Service (upper layer/s)	
		connectionless	**connection-oriented**
L3 Implementation	**Datagram**	**typically**: UDP via IP	TCP via IP
	virtual circuit	UDP/IP via ATM	**typically**: ATM AAL1 via ATM

Service ISO CONS: Model



NSAP: Network Service Access Point

Service ISO CLNS: Model



Service provider can

- **delete objects in a queue**
- **duplicate objects in a queue and**
- **change the object sequence within a queue**

Virtual Circuit

Connect phase:

- **select a path**
- **Intermediate systems (IS) store path information**
- **network reserves all resources required for the connection**

Data transfer phase: all packets follow the selected path

- **packet contains VC_number**
  (identification of connection, no complete address information)
- **IS uses the stored path information to determine the successor**

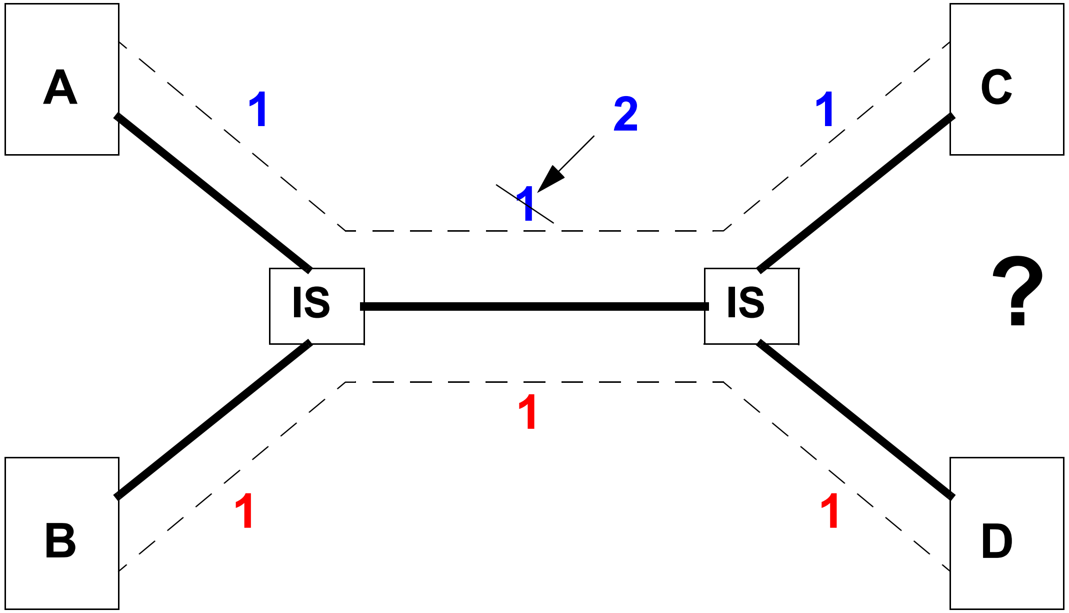
Disconnect phase:

- **network forgets the path**
- **releases reserved resources**

Virtual Circuit (2)

End systems ES allocate VC-identifiers (VC-numbers) independently

Problem: the same VC-identifiers may be allocated to different paths

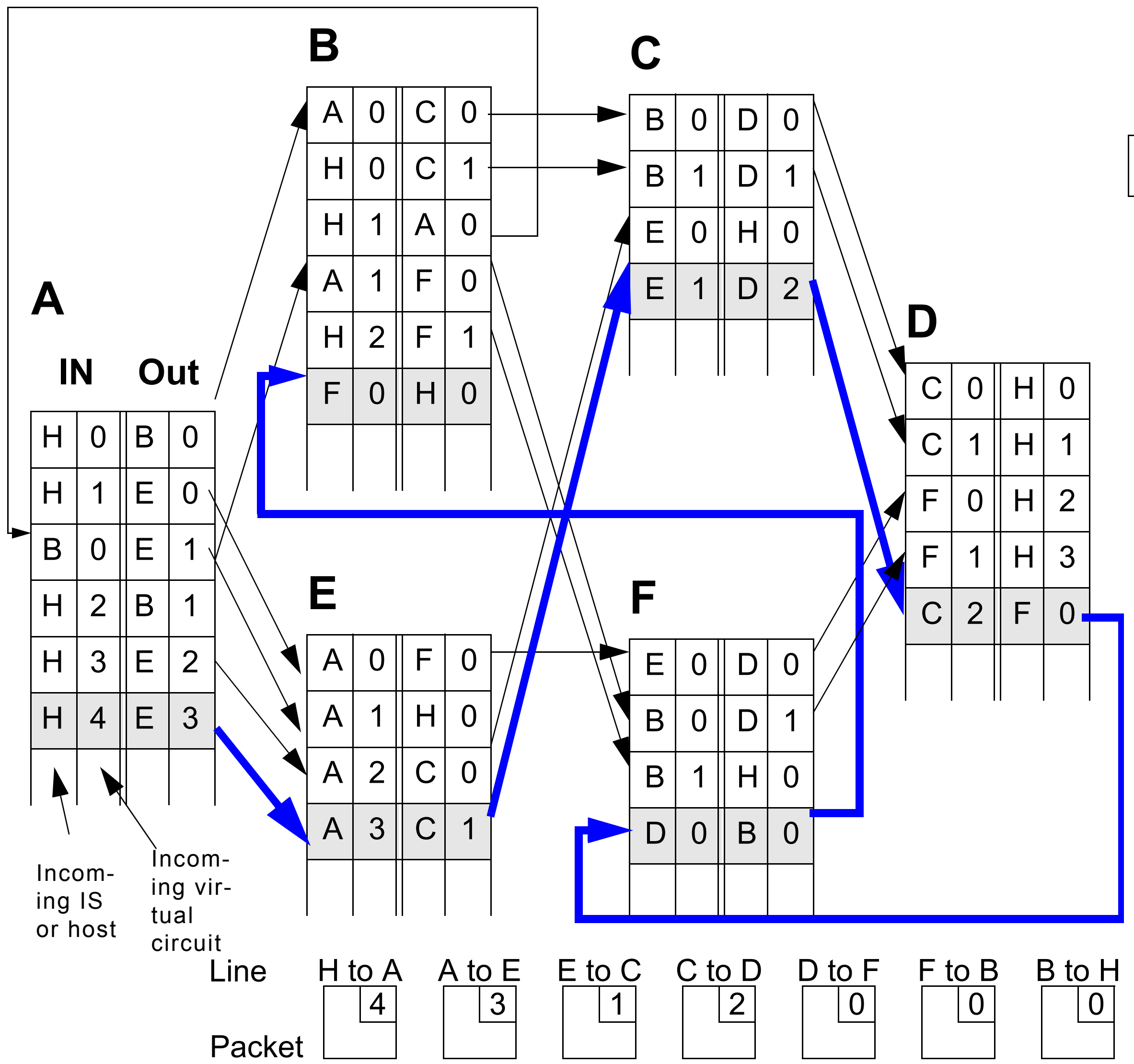


Solution: allocate VC-numbers for virtual circuit segments

- **IS differentiates between incoming and outgoing VC-number**
  1. **IS receives incoming VC-number in CONNECT.ind**
  2. **IS creates outgoing VC-number (unique between IS and successor(IS))**
  3. **IS sends outgoing VC-number in CONNECT.req**

Virtual Circuit (3)

Example:



A

IN		Out	
H	0	B	0
H	1	E	0
B	0	E	1
H	2	B	1
H	3	E	2
H	4	E	3

Incoming IS or host

Incoming virtual circuit

B

A	0	C	0
H	0	C	1
H	1	A	0
A	1	F	0
H	2	F	1
F	0	H	0

C

B	0	D	0
B	1	D	1
E	0	H	0
E	1	D	2

D

C	0	H	0
C	1	H	1
F	0	H	2
F	1	H	3
C	2	F	0

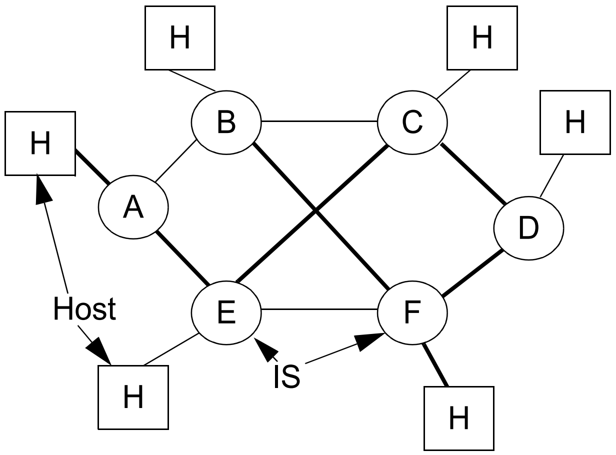
E

A	0	F	0
A	1	H	0
A	2	C	0
A	3	C	1

F

E	0	D	0
B	0	D	1
B	1	H	0
D	0	B	0

Line	H to A	A to E	E to C	C to D	D to F	F to B	B to H
Packet	4	3	1	2	0	0	0



8 simplex virtual circuits

Originating at A	Originating at B
0 - ABCD	0 - BCD
1 - AEFD	1 - BAE
2 - ABFD	2 - BF
3 - AEC	
4 - AECDFB	

Datagram

Every 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**

Datagram vs. Virtual Circuit: Some Comparison

virtual circuit: destination address defined by connection

- + packets contain short VC-number only
- + low overhead during transfer phase
- + "perfect" channel throughout the net
- + resource reservation: "Quality of Service" guarantees possible

- **but:**
  - - overhead for connection setup
  - - memory for VC tables and state information needed in every IS
  - - sensible to IS and link failures
  - - resource reservation: potentially poor utilization

Datagram: IS routing table specifies possible path(s)

- + no connection setup delay
- + less sensible to IS and link failures
- + route selection for each datagram: quick reaction to failures

but:

- - each packet contains the full destination and source address
- - route selection for each datagram: overhead
- - QoS guarantees hardly possible

4. Routing

Task:

- **define the route of packets through the network from the source to the destination system**

ROUTING ALGORITHM:

- **define on which outgoing line an incoming packet will be transmitted on**

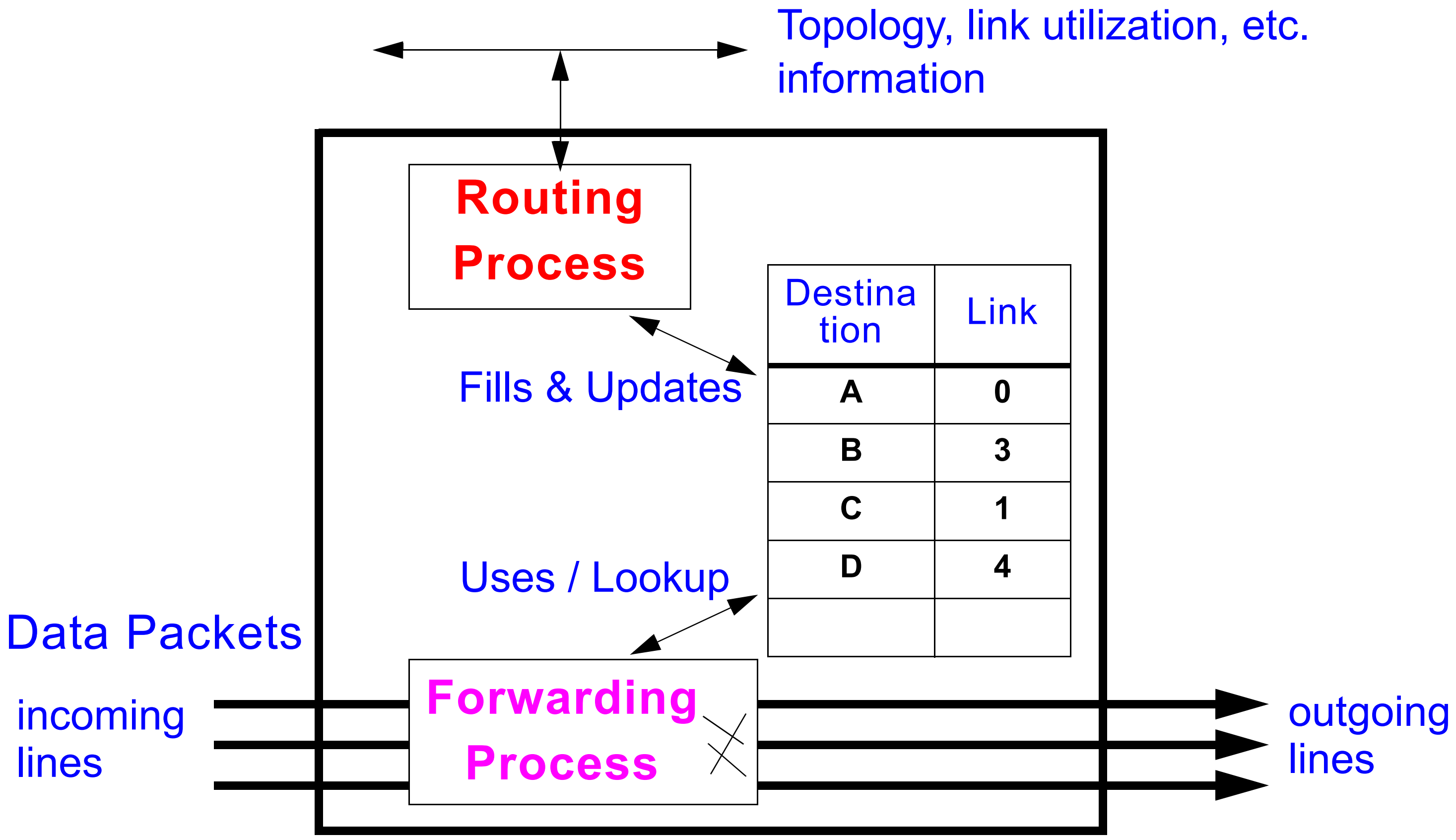
Route determination:

- **datagram:**
  - individual decision for each packet
- **virtual circuit:**
  - routing only during connect (session routing)

Routing & Forwarding

Distinction can be made

- **Routing: make decision which route to use**
- **Forwarding: what happens when a packet arrives**



Topology, link utilization, etc. information

Routing Process

Fills & Updates

Destination	Link
A	0
B	3
C	1
D	4

Uses / Lookup

Data Packets

incoming lines

Forwarding Process

outgoing lines

Desirable Properties of a Routing Algorithm

correctness

simplicity

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

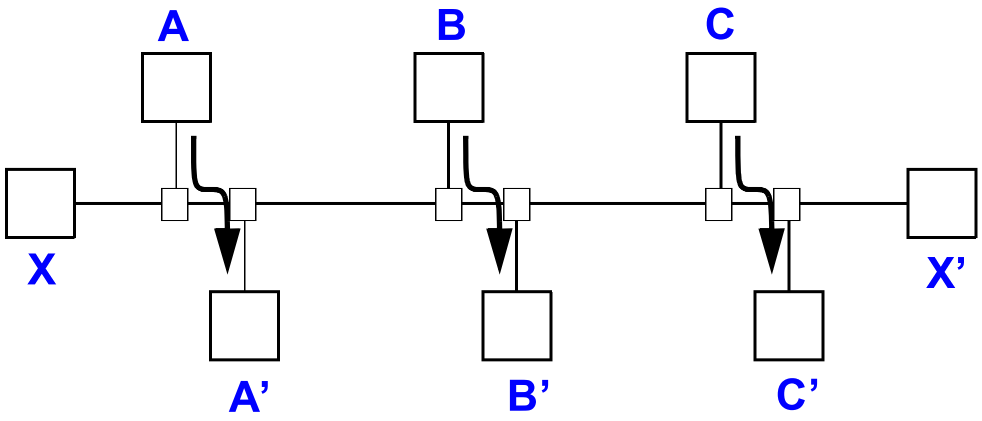
optimality

Routing Algorithms: Conflicting Properties

Often conflicting: fairness and optimization

Example:

- **Communication among A → A', B → B', C → C' uses full capacity of horizontal line**



- **optimized throughput, but**
- **no fairness for X and X'**

⇒ **tradeoff between fairness and optimization**

some different optimization criteria

- **average packet delay**
- **total throughput**
- **individual delay**

⇒ **conflict**

therefore often

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

Classes of Routing Algorithms

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

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

Enhancements (adaptive and non-adaptive algorithms)

- **multiple routing and hierarchical routing definition**

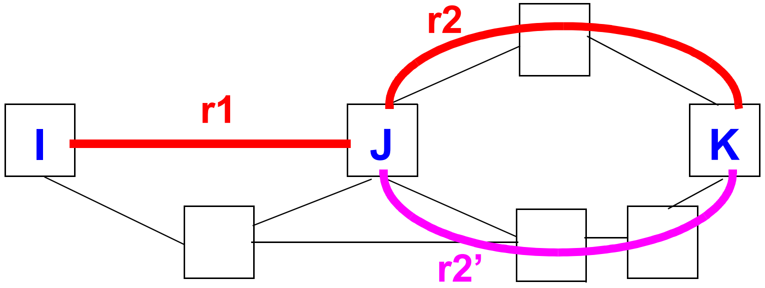
Optimality Principle and Sink Tree

General statement about optimal routes:

> ***if router J is on optimal path from router I to router K***
> ***then the optimal path from router J to router K uses the same route***

Example:

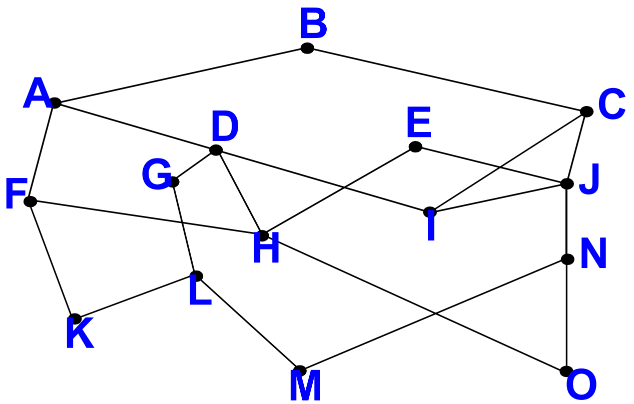
- **r1: route from I to J**
- **r2: route from J to K**
- **if better route r2' from J to K would exist then concatenation of r1 and r2' would improve route from I to K (contradiction)**



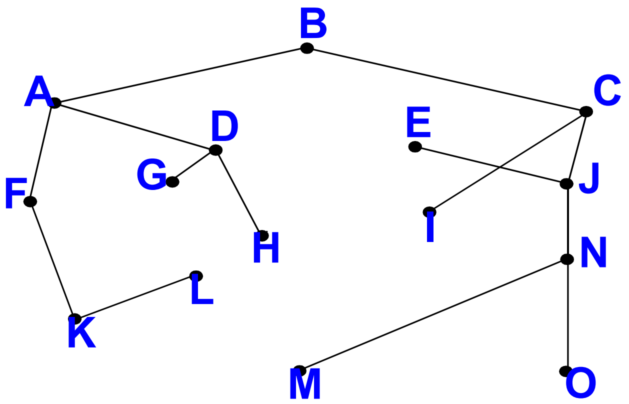
⇒ **set of optimal routes from all sources to a given destination form a tree rooted at the destination: SINK TREE**

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

- **discover and 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

Methodology & Metrics

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)**

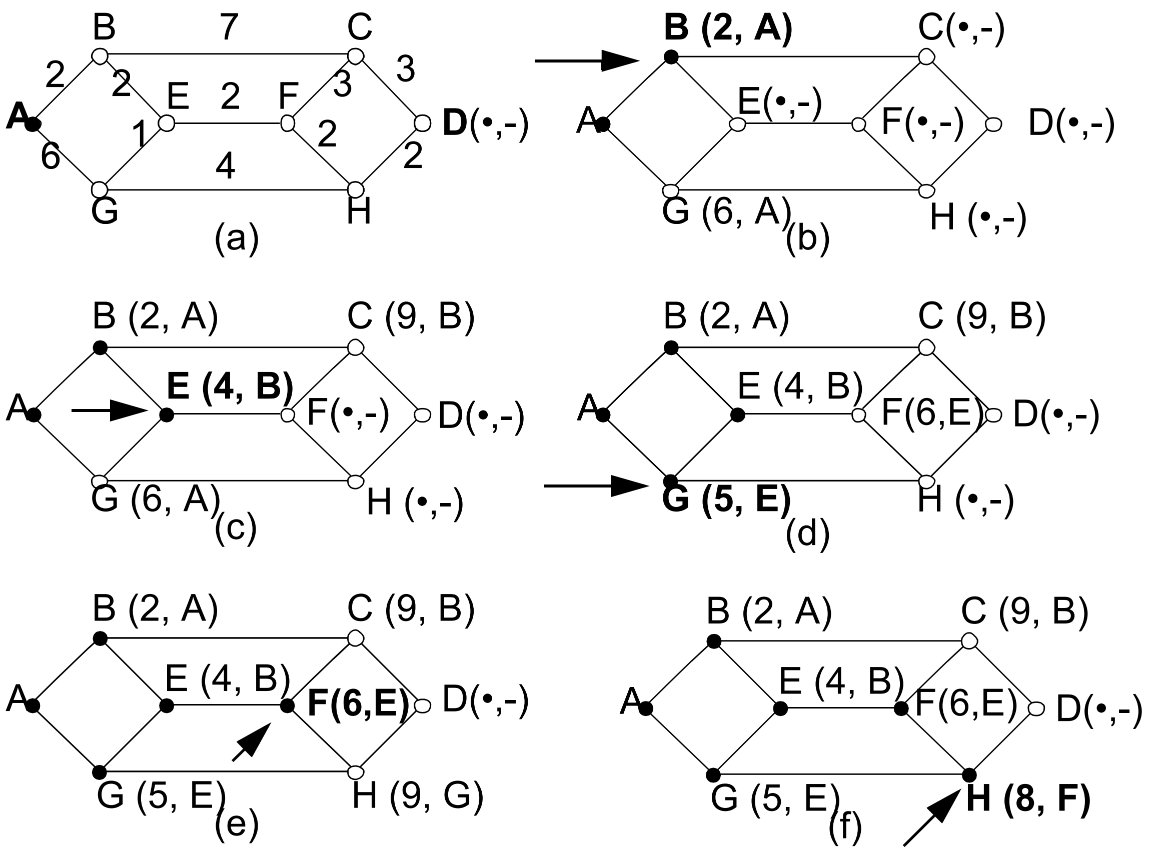
Metrics for the "ideal" route, e.g., a "short" route

- **number of hops**
- **geographical distance**
- **bandwidth**
- **average data volume**
- **cost of communication**
- **delay in queues**
- **...**

4.1 Shortest Path Routing

Example:

- **link is labeled with distance / weight**
- **node is labeled with distance from source node along best known path (in parentheses)**



Shortest Path Routing (2)

Procedure: e.g., according to Dijkstra

find the shortest path from A to D:

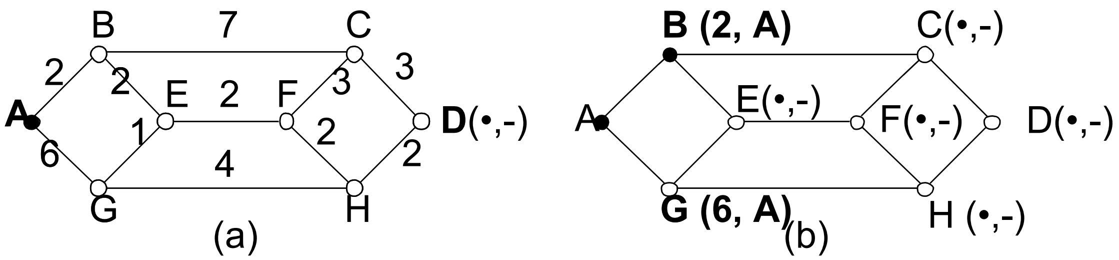
- **labels may be permanent or tentative**
- **initially, no paths are known → all nodes are labeled with infinity (tentative)**
- **discovery that a label represents shortest possible path from source to node:**
  → label is made permanent

1. **Node A labeled as permanent (filled-in circle)**
2. **relabel all directly adjacent nodes with the distance to A**
   (path length, nodes adjacent to source):
   - e.g. B(2,A) and G(6,A)
3. **examine all tentatively labeled nodes;**
   make the node with the smallest label permanent
   - e.g. B(2,A)
4. **this node will be the new working node for the iterative procedure**
   (i.e., continue with step 2.)

Shortest Path Routing (worksheet 1) (3)

Example:

- **link is labeled with distance**
- **node is labeled with distance from source along best known path**



Procedure: e.g., according to Dijkstra

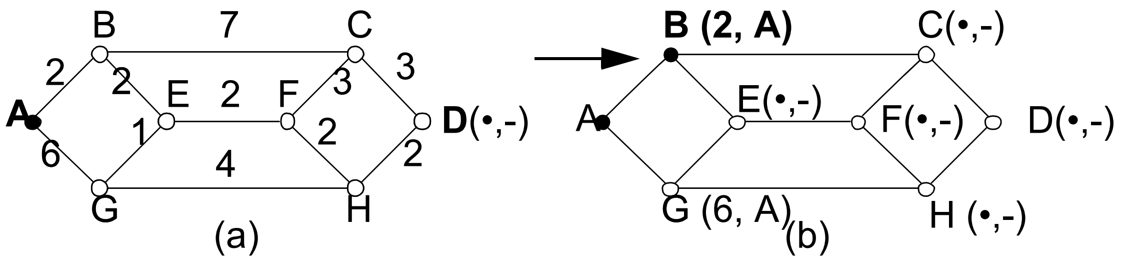
find the shortest path from A to D:

1. **Node A labeled as permanent (filled-in circle)**
2. **relabel all directly adjacent nodes with the distance to A (path length, nodes adjacent to source):**
   - e.g. B(2,A) and G(6,A)

Shortest Path Routing (worksheet 2) (4)

Example:

- **link is labeled with distance**
- **node is labeled with distance from source along best known path**



Procedure: e.g., according to Dijkstra

find the shortest path from A to D:

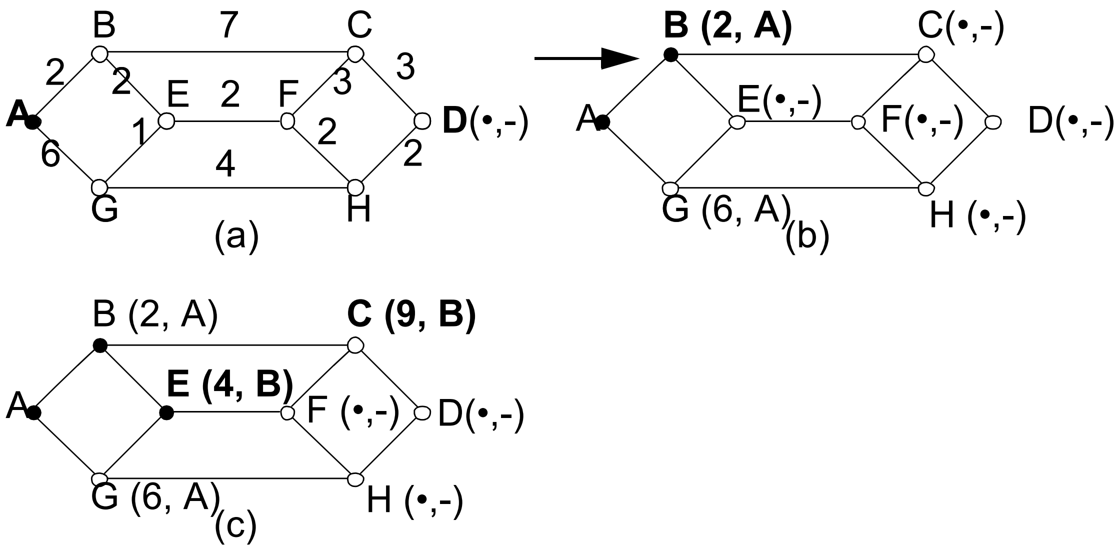
...

3. **examine all tentatively labeled nodes;**
   make the node with the smallest label permanent
   - e.g. B(2,A)
4. **this node will be the new working node for the iterative procedure (i.e., continue with step 2.)**

Shortest Path Routing (worksheet 3) (5)

Example:

- **link is labeled with distance**
- **node is labeled with distance from source along best known path**



Procedure: e.g., according to Dijkstra

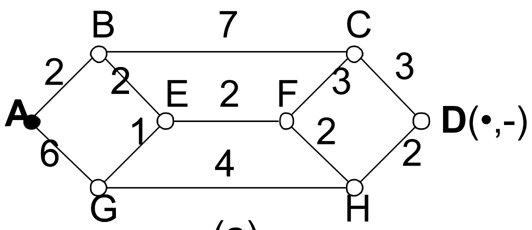
find the shortest path from A to D:

1. **Node B has been labeled as permanent (filled-in circle)**
2. **relabel all directly adjacent nodes with the distance to B (path length, nodes adjacent to source):**
    - A (does not apply, because it is the origin),
    - i.e. E (4,B), C (9,B)

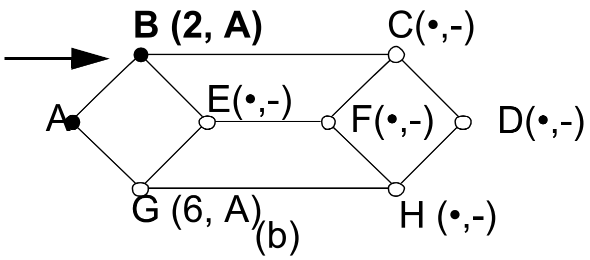
Shortest Path Routing (worksheet 4) (6)

Example:

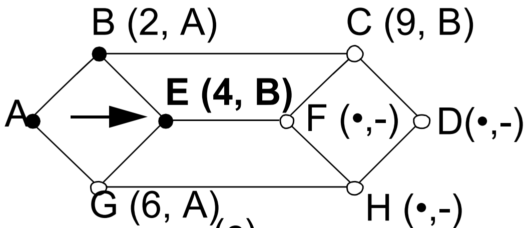
- **link is labeled with distance**
- **node is labeled with distance from source along best known path**



(a)



(b)



(c)

Procedure: e.g., according to Dijkstra

find the shortest path from A to D:

1. **....**
2. **....**
3. **examine all tentatively labeled nodes;**
   make the node with the smallest label permanent: e.g. E(4,B)
4. **this node will be the new working node for the iterative procedure ...**

4.2 Flow-Based Routing

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

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

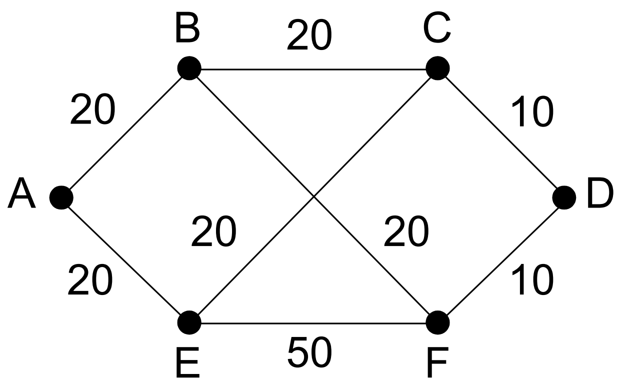
   includes

   - serving time (occurs also during no load, λ_i=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 overall delays result from selecting different paths;**
   - subnetwork with **MINIMAL OVERALL DELAY** used for routing

Flow-Based Routing (2)

Example: requirements

- **network with fully duplex channels,**
- **stating TOPOLOGIES and CAPACITIES**



Destination

Source

	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	

- **stating the paths to be selected including the 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

Flow-Based Routing (3)

Example: initial computation information

i	Line	λ_i(pkts/sec)	C_i(kbps)	μC_i(pks/sec)	T_i(msec)	Weight
1	AB		20			
2	BC		20			
3	CD		10			
4	AE		20			
5	EF		50			
6	FD		10			
7	BF		20			
8	EC		20			

Example: computation results

i	Line	λ_i(pkts/sec)	C_i(kbps)	μC_i(pks/sec)	T_i(msec)	Weight
1	AB	14	20	25	91	0.171
2	BC	12	20	25	77	0.146
3	CD	6	10	12.5	154	0.073
4	AE	11	20	25	71	0.134
5	EF	13	50	62.5	20	0.159
6	FD	8	10	12.5	222	0.098
7	BF	10	20	25	67	0.122
8	EC	8	20	25	59	0.098

IBR (Institut für Betriebssysteme und Rechnerverbund) – TU Braunschweig

Kommunikationssysteme: Network Layer

Flow-Based Routing (4)

λ_i **Average load:**
the sum of all median packets/sec at the respective edge

- example: AB = AB (AB=9) + AC (ABC=4) + AD (ABFD=1) = 14

C_i **Capacity of each edge in kbps (known from the graph)**

μC_i **Capacity of each edge at given median packet size**

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

$$\mu C_i = \frac{20 \text{ kbit/sec}}{800 \text{ bit/packet}} = 25 \text{ packets/sec}$$

T_i **Average delay on each path**

$$T_i = \frac{1}{\mu\ C_i - \lambda_i}$$

- example $T_1 = \dfrac{1}{25 \text{ packets/sec} - 14 \text{ packets/sec}} = 90{,}909\ ...\ \text{msec/packet}$

Flow-Based Routing (5)

Weight: the relative traffic of data using this path

- (in relation to the overall traffic)
- example

$$\text{Weight (AB)} = \frac{\text{(average load AB)}}{\sum\limits_{\text{all paths xy}} \text{average load xy}} = \frac{14}{82} = 0,1707$$

⇒ **Average overall delay for the subnetwork:**

- example

$$\sum_{\text{all path ij}} \text{Weight(ij)} \times \text{average delay (ij)} = 86 \text{ msec}$$

4.3 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

- **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
- **each station remembers the packets that have already been transfered and deletes them upon recurrence**
  - source router inserts sequence number into packets received from hosts
  - each router needs a ’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
- with ’regular’ topologies this makes sense and is an optimization
- but some topologies do not fit well to this approach

Comment:

- geographically-oriented routing got recent interest for mobile scenarios

Flooding: Evalution 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

Summary: Static Routing Procedures

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

- **but:**
  - poor performance if traffic volume or topologies change over time

4.4 Centralized Adaptive 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 neighbours
    - actual queue lengths
    - line utilization, etc.
- **RCC**
    - collects information
    - calculates the optimum path for each IS pair
    - generates routing tables and distributes these to the ISs

Example: TYMNET

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

Centralized Adaptive Routing (2)

Characteristics:

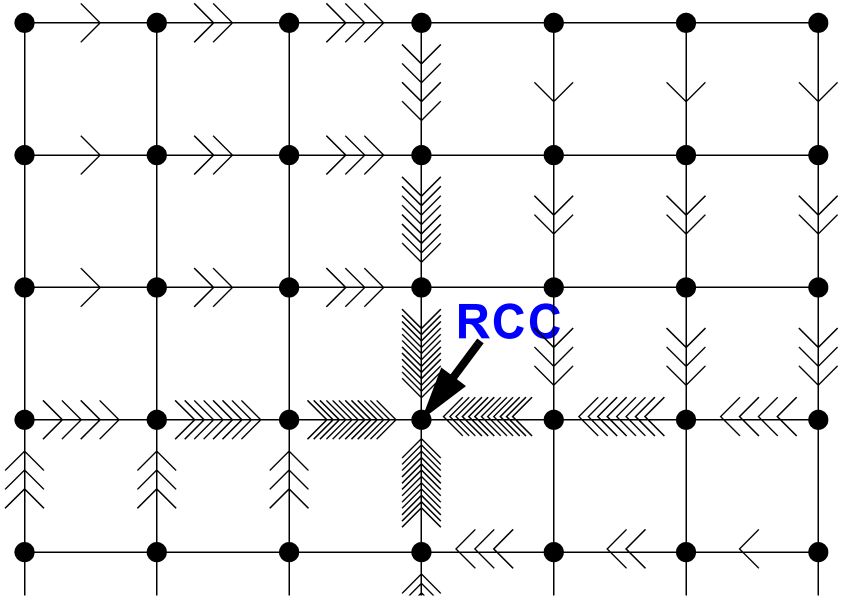
- **RCC has complete information**

⇒ perfect decisions

- **and 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**
- **ISs receive tables at different times**
- **traffic concentration in RCC proximity**



4.5 Isolated Procedures: Backward-Learning Algorithm

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**

Packet of source S is received on line L after C hops

⇒ **S is reachable on L within C hops**

Routing table in IS

- **L - table (destination - IS, outgoing line, C_{min})**
- **update of the routing table**

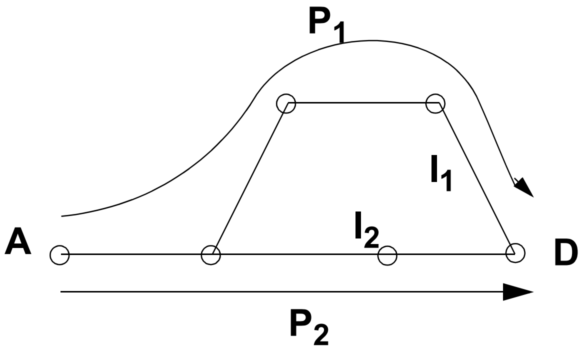
IS receives packet (..., S, C, ...) on L

```
if not (S in L-Table)
 then  Add(S,L,C)
 else if C < $C_{min}$
     then Update(S,L,C)
```

Isolated Procedures: Backward-Learning Algorithm (2)

Example:

- **packet (..., source - IS, hop counter, ...)**
  - P_1 (..., A, 4, ...)→ Add (A, I_1, 4)
  - P_2 (..., 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

4.6 Distributed Routing: Distance Vector Routing

DISTANCE VECTOR ROUTING ALGORITHM

- **also known as distributed Bellman-Ford algorithm, Ford-Fulkerson algorithm**
- **was the original ARPANET routing algorithm**
- **has been used in the Internet as RIP Routing Information Protocol**

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 neighbours**

Distance Vector Routing: Procedure

Each IS maintains routing table with one entry per router in the subnet

- estimate of the distance (hops, delay, packets queued, ...) to destination
- outgoing line to be used for that destination

Each IS is assumed to know the "distance(s)" to each neighbour

- number of hops (= 1)
- delay (echo packets)
- queue length (e.g., used in the ARPANET),...

IS sends lists with estimated distances to each destination periodically to its neighbours Y

- e.g., Internet RIP every 30 sec, maximum distance 15 hops

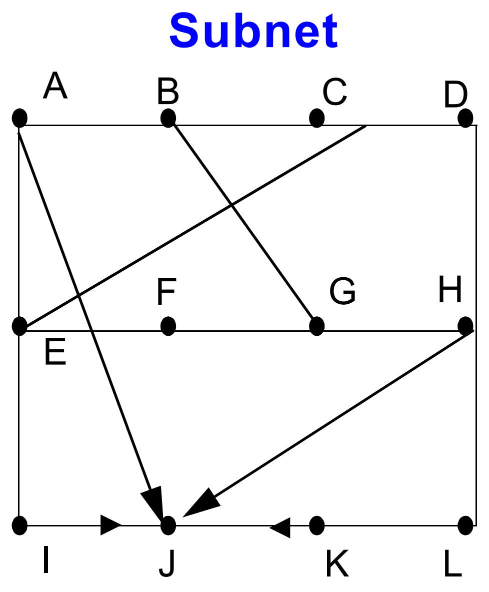
X receives list E(Z) from neighbour Y

• distance X to Y:	e
• distance Y to Z:	E(Z)
• i.e. distance X to Z (via Y):	E(Z) + e

IS calculates a new routing table from the received lists, containing

- destination IS, prefered outgoing path, "distance"

Distance Vector Routing: Exampe



delays at and of nodes A/I/H/K/. (row).
to nodes A;B;C;D.. (column)

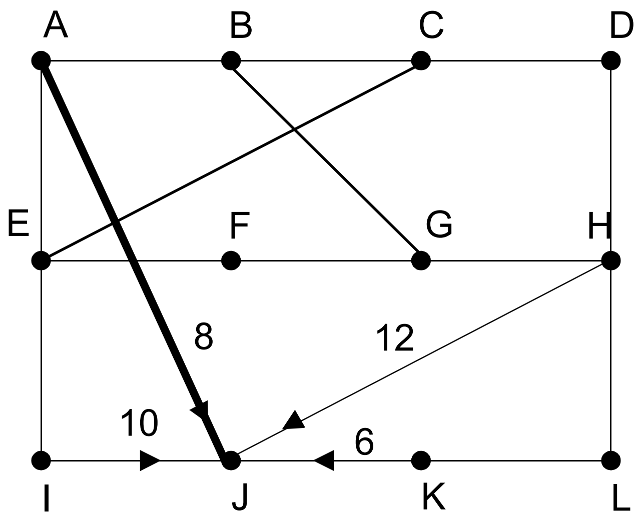
To	A	I	H	K	new estimated delay from J	line
A	0	24	20	21	8	A
B	12	36	31	28	20	A
C	25	18	19	36	28	I
D	40	27	8	24	20	H
E	14	7	30	22	17	I
F	23	20	19	40	30	I
G	18	31	6	31	18	H
H	17	20	0	19	12	H
I	21	0	14	22	10	I
J	9	11	7	10	0	-
K	24	22	22	0	6	K
L	29	33	9	9	15	K
	JA Delay= 8	JI Delay= 10	JH Delay= 12	JK Delay= 6	new routing table for J	

Previous routing table will not be taken into consideration

⇒ **Reaction to deteriorations**

Distributed Routing: Distance Vector Routing (3)

Example: defining a section



.**B**.sends information to node J

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

from B via **A**: costs (JA) + costs path (AB) = 8 + 12 = 20
from B via **I**: costs (JI) + costs path (IB) = 10 + 36 = 46
from B via **H**: costs (JH) + costs path (HB) = 12 +31 = 43
from B via **K**: costs (JK) + costs path (KB) = 6 + 28 = 34

Select Minimum, Min (JAB, JIB, JHB, JKB) = JAB = 20

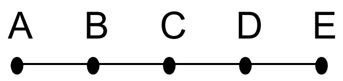
Distance Vector Routing: Feature "Count to Infinity"

Information distribution over new

- short paths (with few hops): fast
- long paths with many hops: SLOW

Example: route improvement

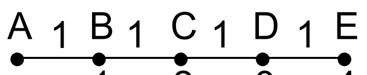
- previously: A unknown
- later: A connected with distance 1 to B, this will be announced
- Note: Synchronous update used here for simplification
- distribution proportional to topological spread



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

Example: deterioration, (here: connection destroyed)

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



B	C	D	E	
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!

B	C	D	E	
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges
7	8	7	8	After 6 exchanges
	⋮			
∞	∞	∞	∞	

Comment

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

Distance Vector Routing: Variant "Split Horizon Algorithm"

Objective - based on the Distance Vector principle

- but improve the "count to infinity" property

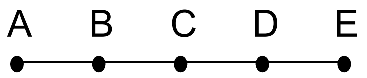
Principle

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

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

Example:
deterioration,
e.g., connection destroyed

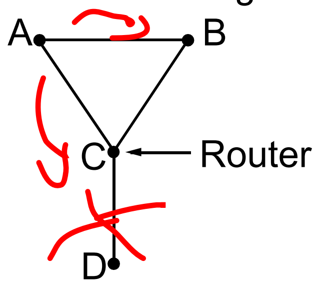
- B to C: A = ∞ (real),
  C to B: A = ∞ (because A is on path), ...



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

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)**

4.7 Link State Routing

also "distributed routing"

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**

Use

- **introduced into the ARPANET in 1979, nowadays most prevalent**
- **IS-IS (Intermediate System-Intermediate System)**
  - developed by DECNET
  - also used as ISO CLNP in NSFNET
  - Novell Netware developed its own variation from this (NLSP)
- **OSPF (Open Shortest Path First)**
  - since 1990 Internet RFC 1247

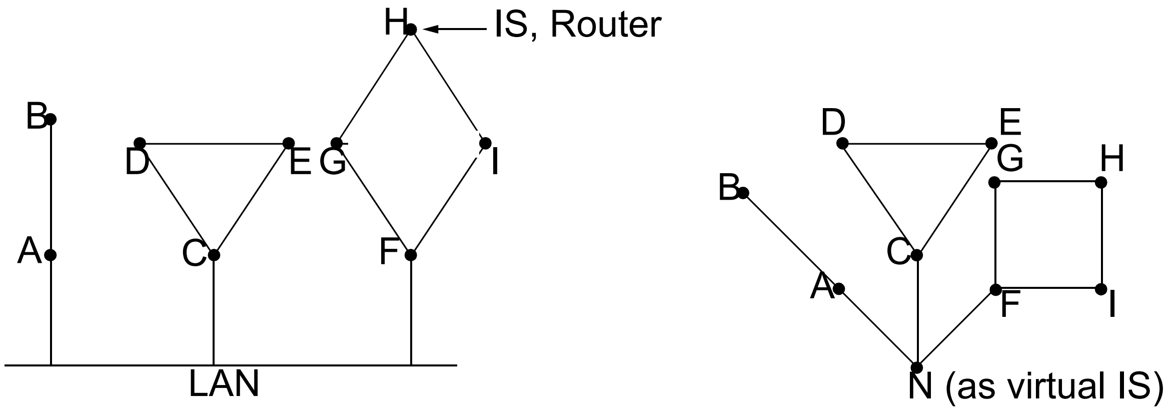
Link State Routing (2)

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

Example:

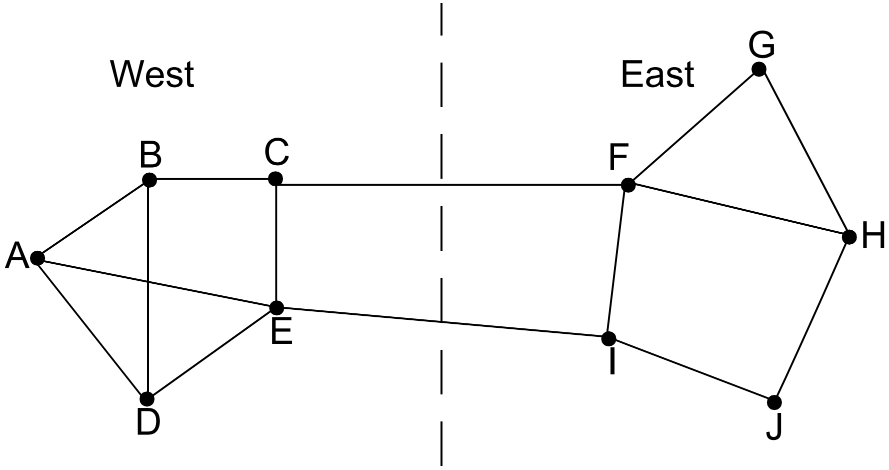
- **with LAN (as virtual IS)**



Link State Routing (3)

2. Phase: 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:

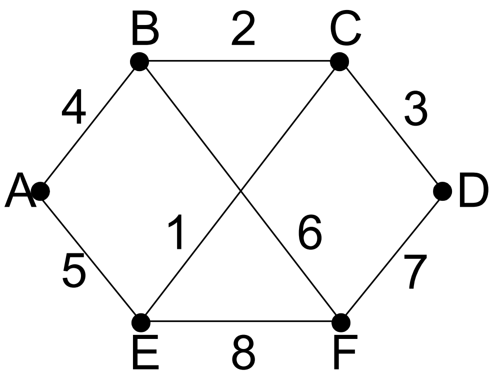


  - after each new routing table
    the other link CF or EI is charged

Link State Routing (4)

3. **Phase: organizing the information as link state packet**

- **including own address, sequence number, age, "distance"**



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

- **timing problems: validity and time of sending**
  - periodically
  - in case of major changes

Link State Routing (5)

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**

- OSPF zur Berechnung

4.8 Multipath Routing

Principle:

- **using alternative routes between the IS pairs**
- **usage frequency 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 Ai will be used:

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

Multipath Routing (2)

Example:

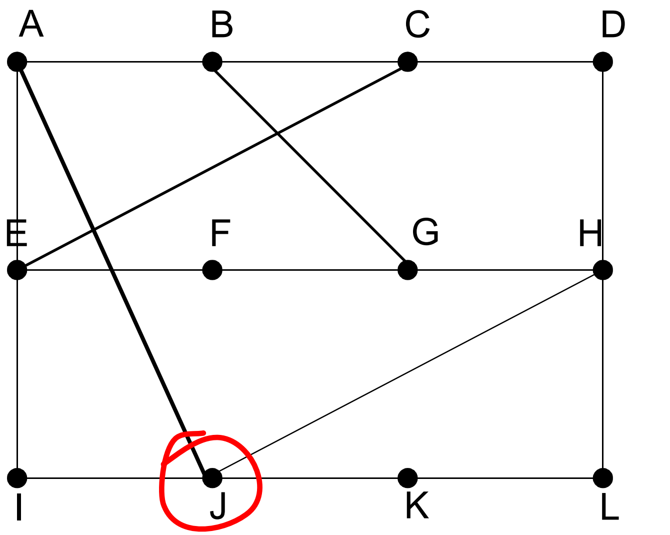


Table from J →

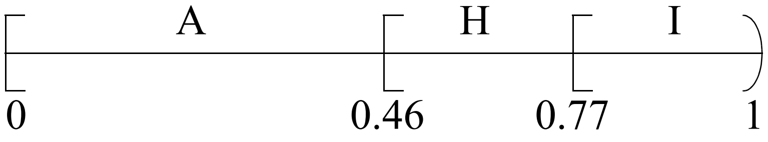
dest.	1st choice		2nd choice		3rd choice	
A	A	0.63	I	0.21	H	0.16
B	A	0.46	H	0.31	I	0.23
C	A	0.34	I	0.33	H	0.33
D	H	0.50	A	0.25	I	0.25
E	A	0.40	I	0.40	H	0.20
F	A	0.34	H	0.33	I	0.33
G	H	0.46	A	0.31	K	0.23
H	H	0.63	K	0.21	A	0.16
I	I	0.65	A	0.22	H	0.13
K	K	0.67	H	0.22	A	0.11

G_1 G_2 G_3

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

$$A_1: 0 \leq z < G_1$$
$$A_2: G_1 \leq z < G_1 + G_2$$
$$....$$
$$A_n: G_1 + G_2 + ... + G_n-1 \leq z < 1$$

Example: destination B



IBR (Institut für Betriebssysteme und Rechnerverbund) – TU Braunschweig

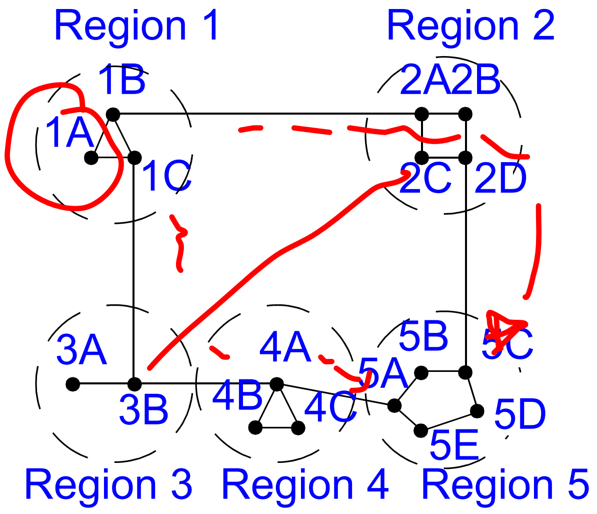
Kommunikationssysteme: Network Layer

4.9 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

Full table for 1A

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

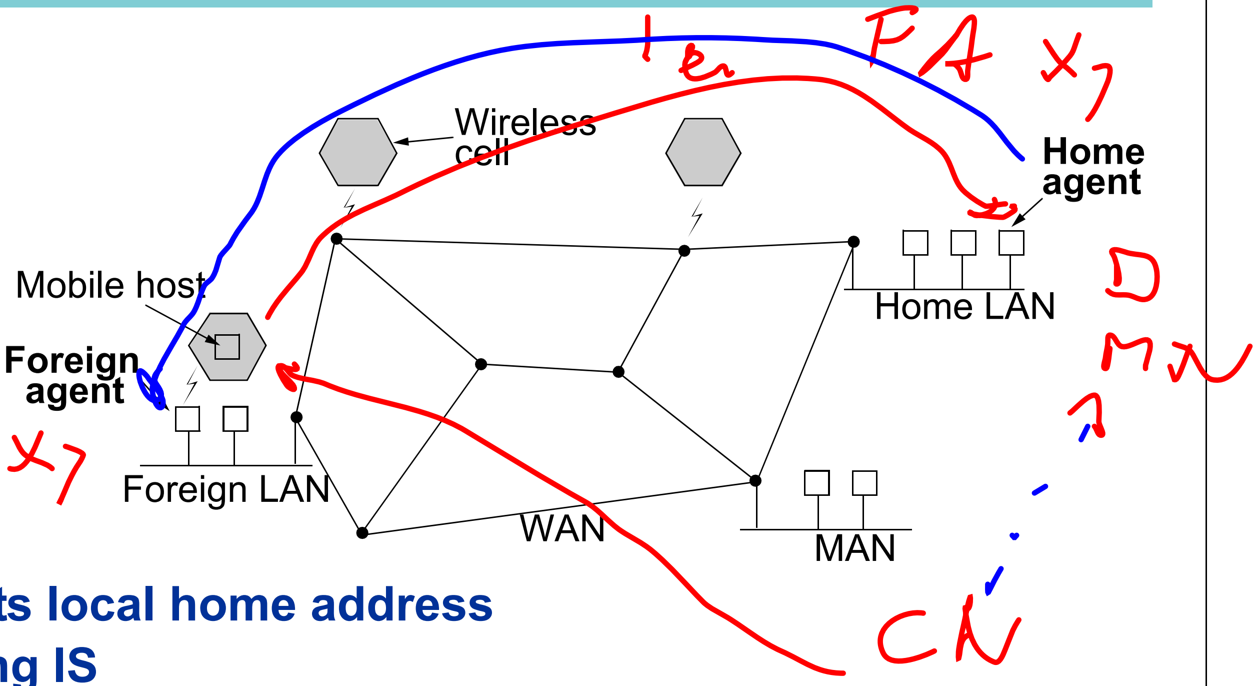
Region {

Comparison

- **but**
  - the best path is not always calculated
- **design:**
  - number of layers

4.10 Routing with Mobility

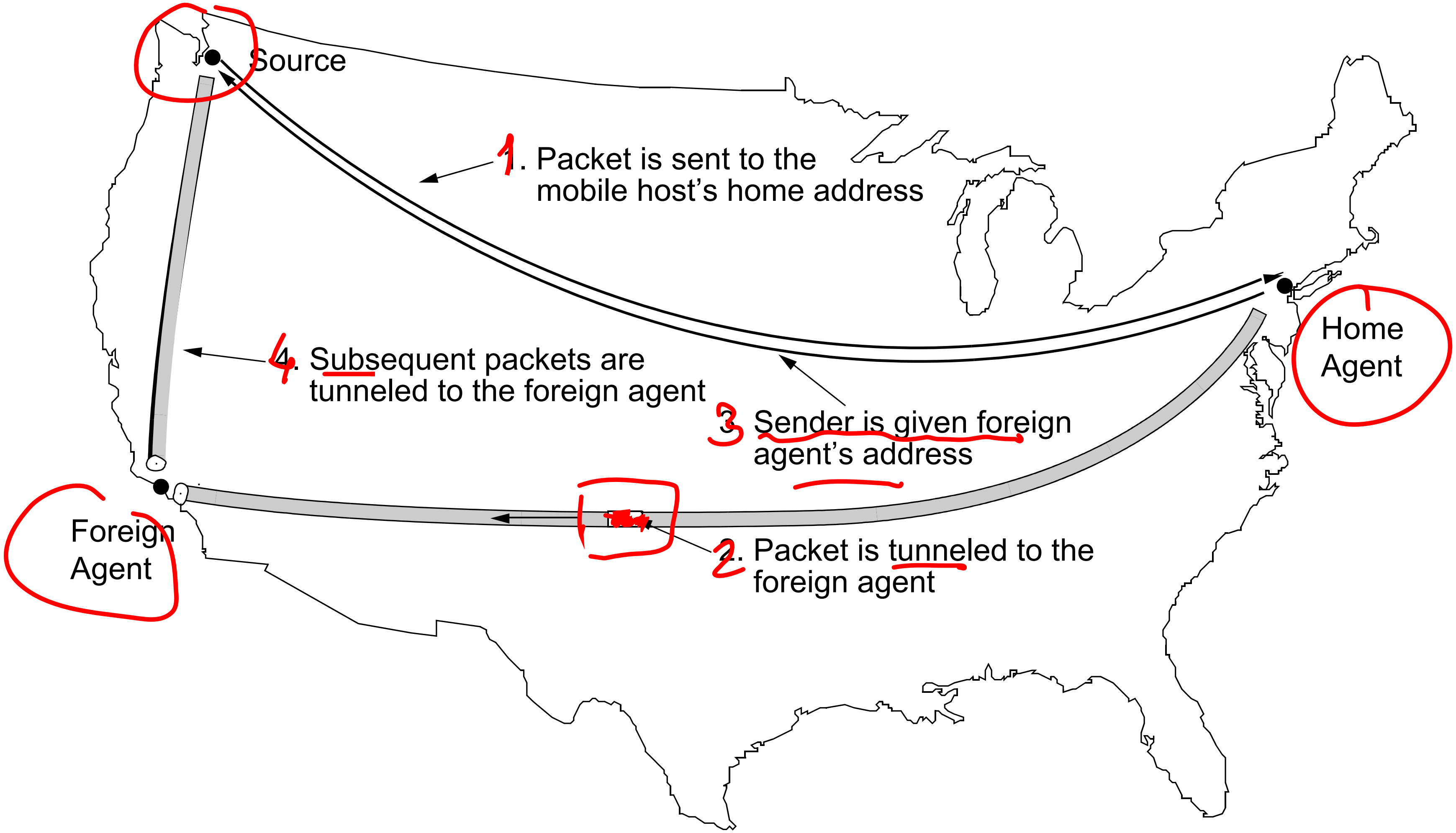
Principle



- **end system identified by its local home address**
- **no modifications in existing IS**
- **i.e.,**
  - Home-Agent: stationary address
  - Foreign Agent: knows mobile end system

Routing with Mobility (2)

Tunneling and Rerouting Procedures



1. Packet is sent to the mobile host’s home address

2. Packet is tunneled to the foreign agent

3. Sender is given foreign agent’s address

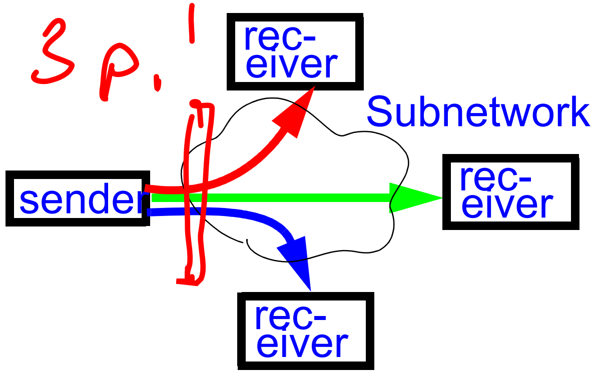
4. Subsequent packets are tunneled to the foreign agent

5. Broadcast Routing

Terminology

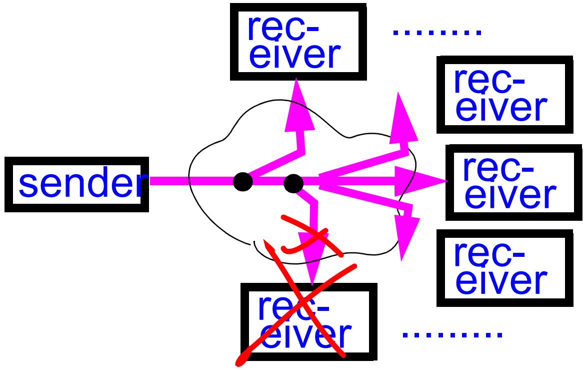
- **Unicast: 1 : 1 communication**

3 p.!



multipeer m:n communication

- **Multicast: 1 : n communication**
- **Broadcast: 1 : all communication**



5.1 Broadcast Routing: Methods

Several methods have been proposed for broadcasting

Simple approaches:

1. Individual sending to every destination (distinct packets)

- + requires no special feature from the network
- waste of bandwidth ⚡
- sender has to know all destinations ⚡

2. Flooding

- too many duplicates ⚡

5.2 Broadcast Routing: Multidestination Routing

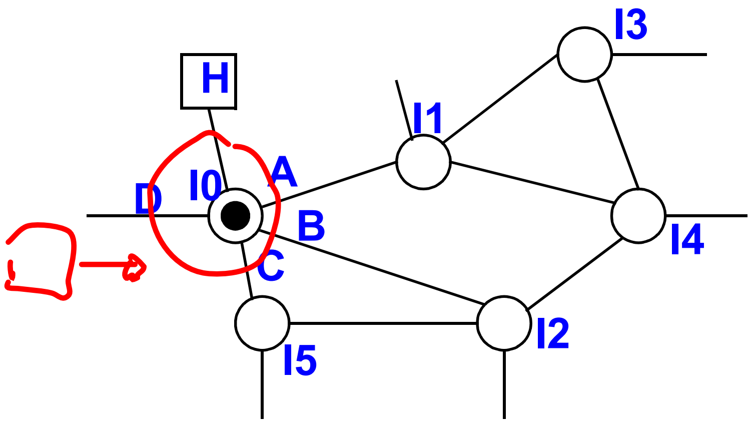
Each Packet CONTAINS A LIST OF DESTINATIONS

Steps performed at each IS

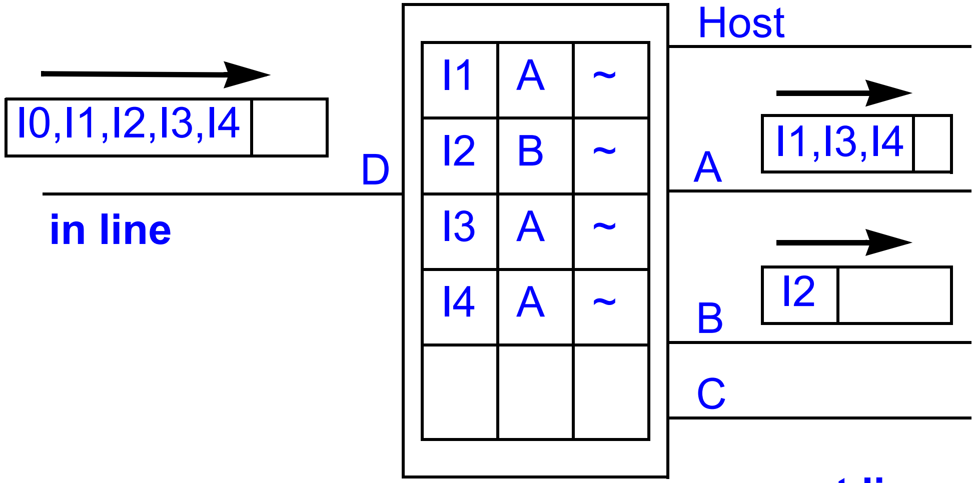
- **examine which outgoing links are required**
- **generates a packet copy for each REQUIRED outgoing link**
  - packet copy contains **ONLY** destinations which can be reached via this line

Example:

- **network with I_0 as the considered IS**



network



I1	A	~
I2	B	~
I3	A	~
I4	A	~

I0 routing table

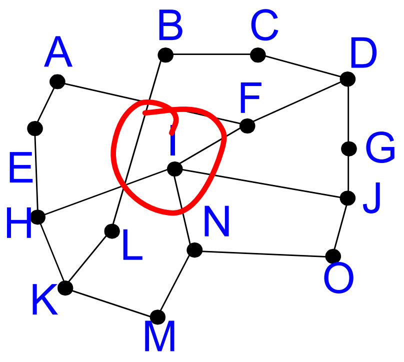
5.3 Broadcast Routing: Spanning Tree

Idea:

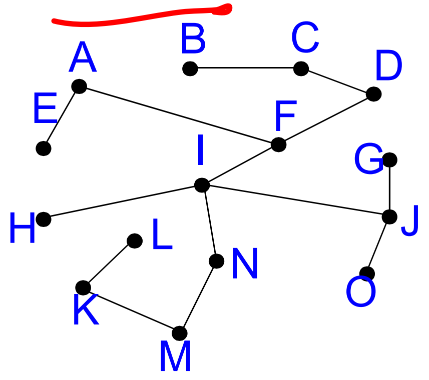
- **use sink tree for router initiating broadcast or other spanning tree**
  - subset of subnet including all routers with no loops

Spanning tree: subset of subnet including all routers with no loops

Example network, IS "I" as the sender



Net



a Spanning Tree

Prerequisite:

- **Spanning Tree is known to the IS**

- **IS generates minimum number of packet copies**
- **IS generates a packet copy for each required outgoing line**
  - all spanning tree lines except incoming one

Main issue:

- **how to determine a Spanning Tree?**
  - sometimes available, e.g., from link state routing
  - sometimes not, e.g., with distance vector

5.4 Broadcast Routing: Reverse Path Forwarding (RPF)

Also called "Reverse Path Flooding" (RPF)

- **Variation of the Spanning Tree**

Principle

- **each sender has his own Spanning Tree**
- **but IS do not need to know the Spanning Trees**

Considerations

- **each router has information which path it would use for (unicast)-packets**
  - because of the unicast routing algorithms

Algorithm (for a packet arriving at an IS)

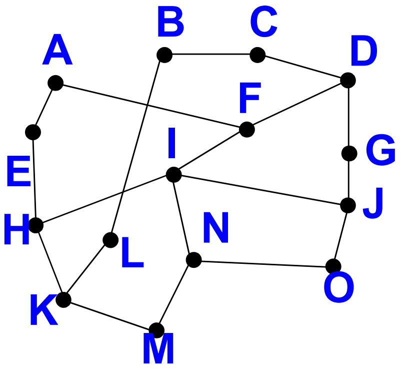
- **has this packet arrived at THE IS entry over which the packets for this station/source are usually also sent?**

  Yes:
  - packet used the **BEST** route until now,
  - action: resend over all edges (not including the incoming one)

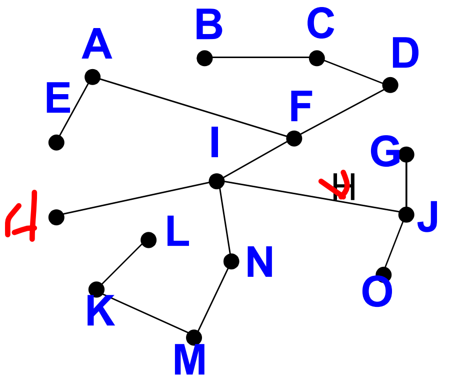
  No:
  - action: discard packet (most likely duplicate)

Broadcast Routing: Reverse Path Forwarding (RPF) (2)

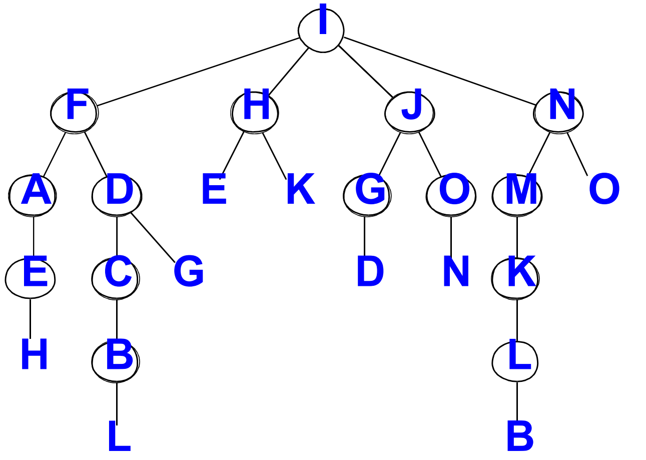
Example



Net



a Spanning Tree



Characteristics

- based on the assumption of SYMMETRIC DUPLEX CHANNELS
- simple implementation (no global conditions, ...)
- metrics
  - consist only of distance

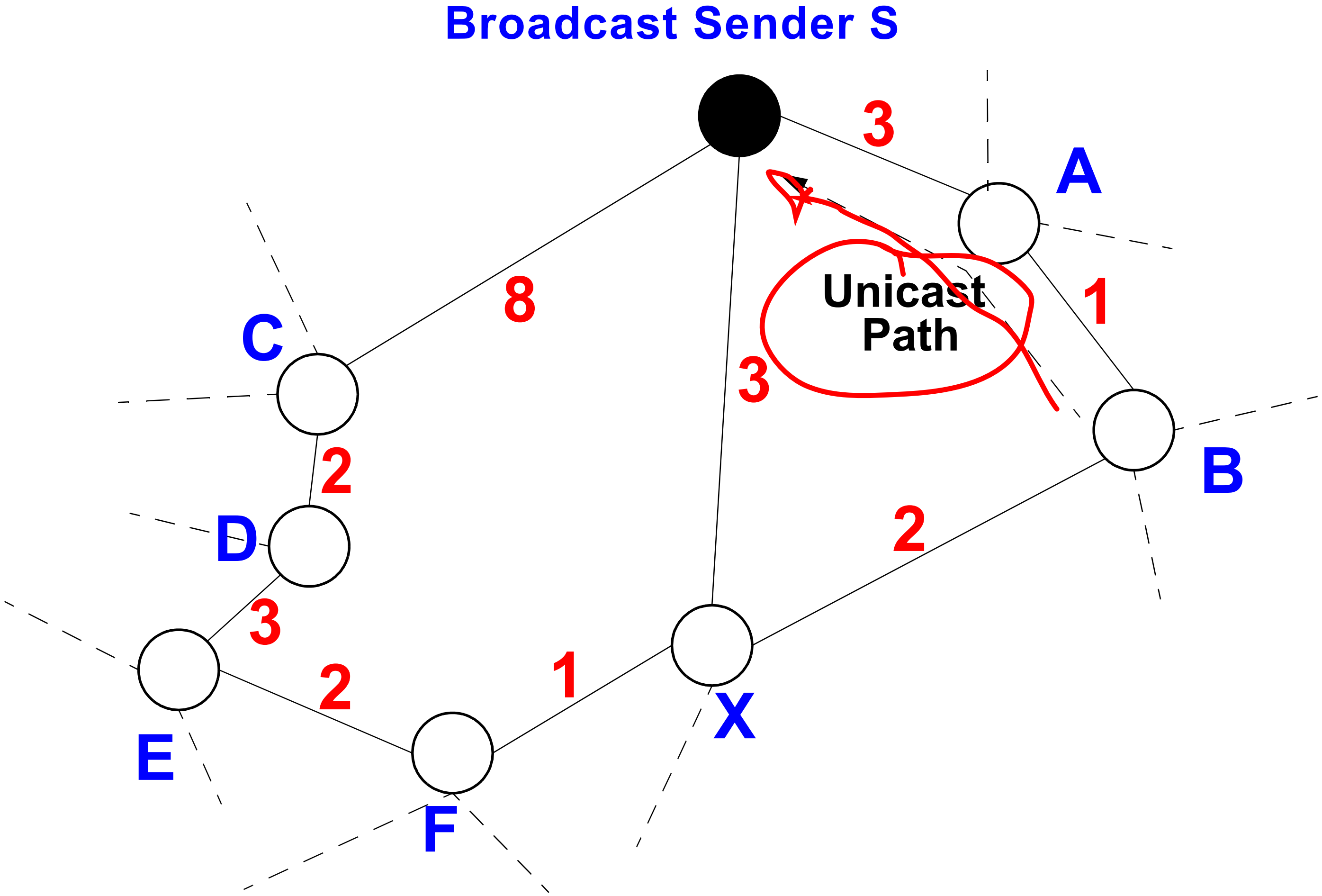
Application:

MBone Multicast Backbone

- between MBone islands

Broadcast Routing: Reverse Path Forwarding (RPF) (3)

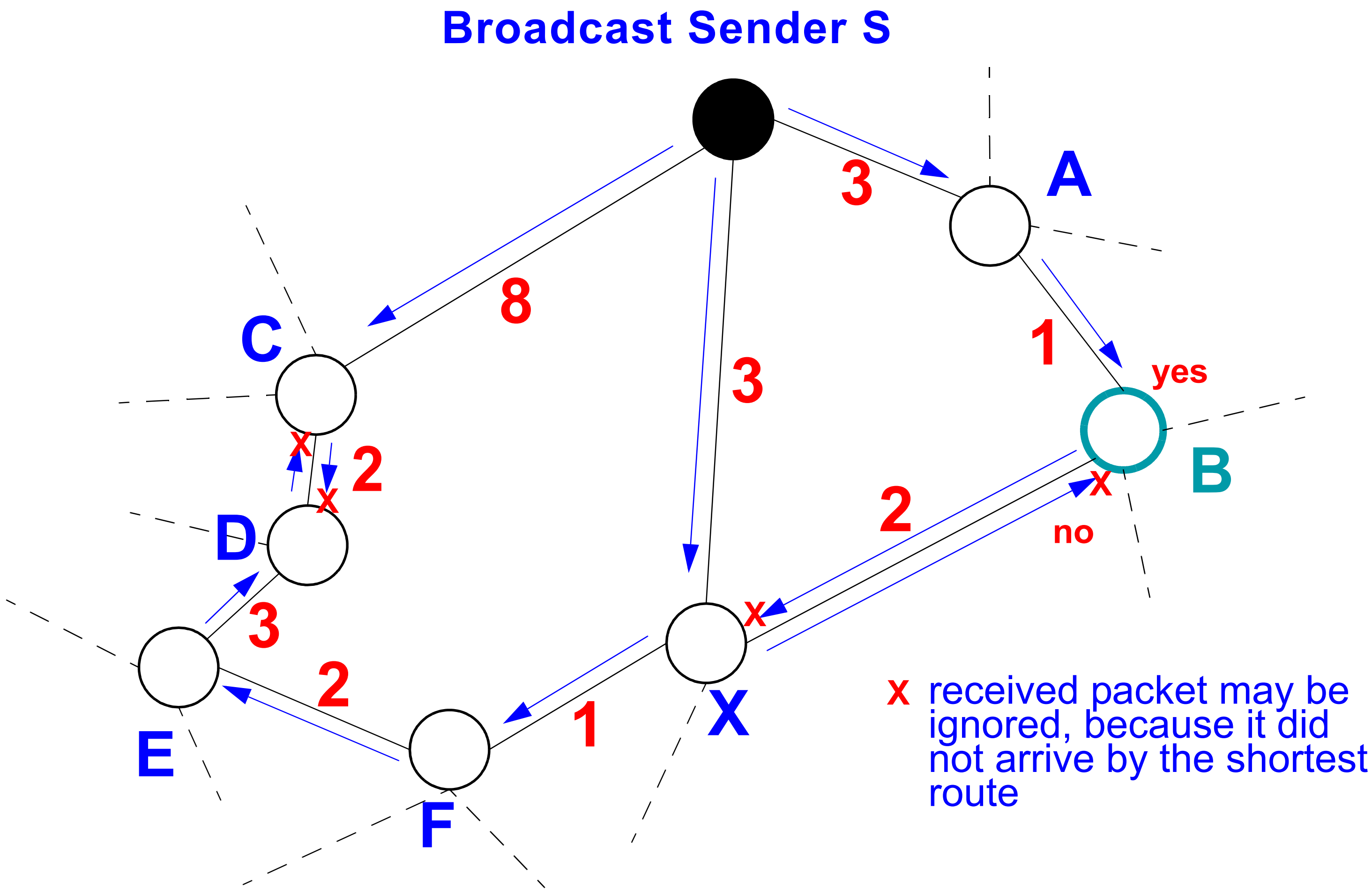
Example:



- **in the example**
  - B will send its unicast packets to S via A (shortest route).

Broadcast Routing: Reverse Path Forwarding (RPF) (4)

Example:



x received packet may be ignored, because it did not arrive by the shortest route

- **within the RPF algorithm of the above example**
  - router B uses the unicast routing information to ignore all broadcast packets received from S, **WHICH DID NOT ARRIVE VIA NODE A**

5.5 Broadcast Routing:Reverse Path Broadcast (RPB)

Motivation: disadvantages of Reverse Path Forwarding:

- **when packets are forwarded,
they are forwarded over ALL edges (not including the incoming one)**
- **better if over only one SUITABLE edge**

Algorithm: packet from S(ource) to D(estination)

- **like REVERSE PATH FORWARDING**
  - with specific selection of the outgoing links
- **has this packet arrived via an IS entry over which packets may also be sent to station/source S?**

Yes:

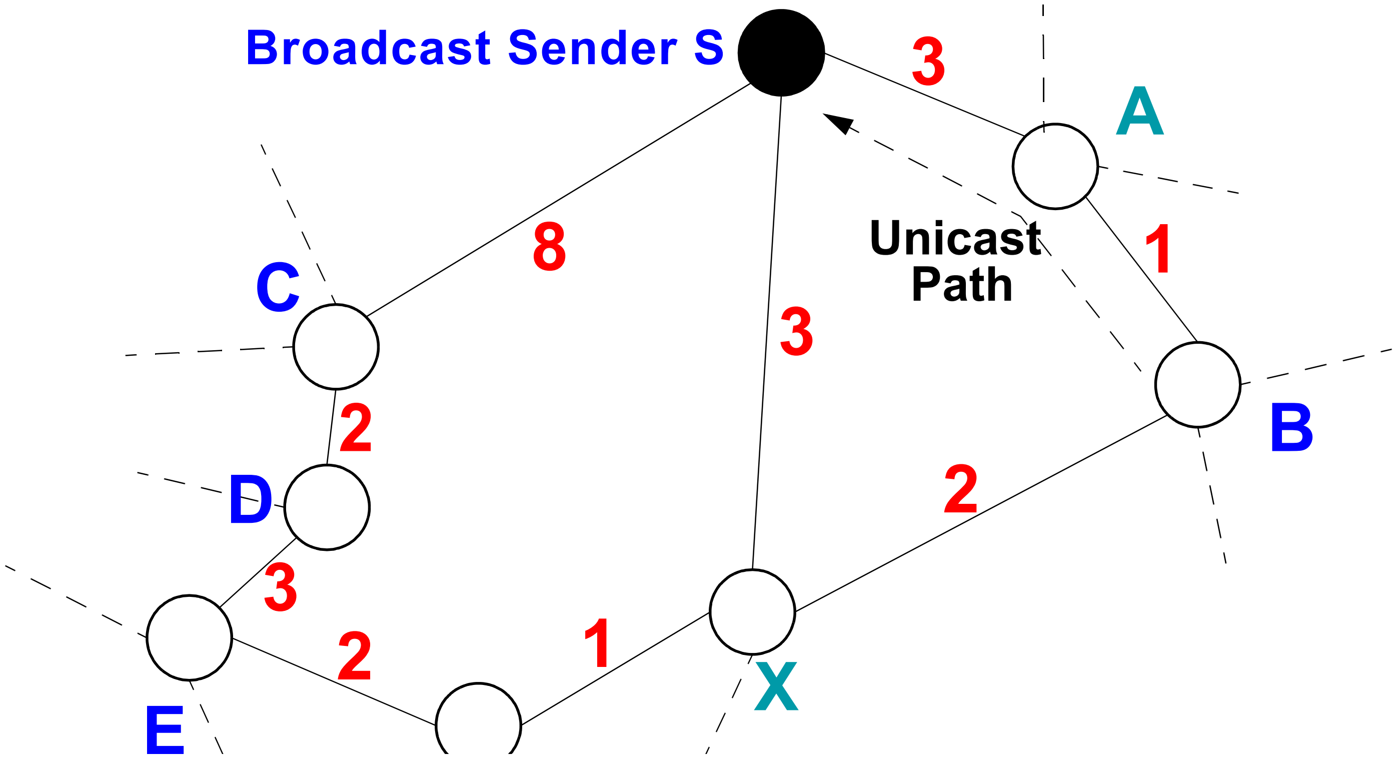
- packet used the **BEST** route until now,
- **THEN:** select the edge at which the packets arrived and from which they are then rerouted to source **S** (in reversed direction)
- **THEN DO NOT** i.e. not as in Reverse Path Forwarding (RPF) ~~send over all edges (without the incoming one)~~

No:

- discard packet (is most likely a duplicate)

Broadcast Routing:Reverse Path Broadcast (RPB) (2)

Example:



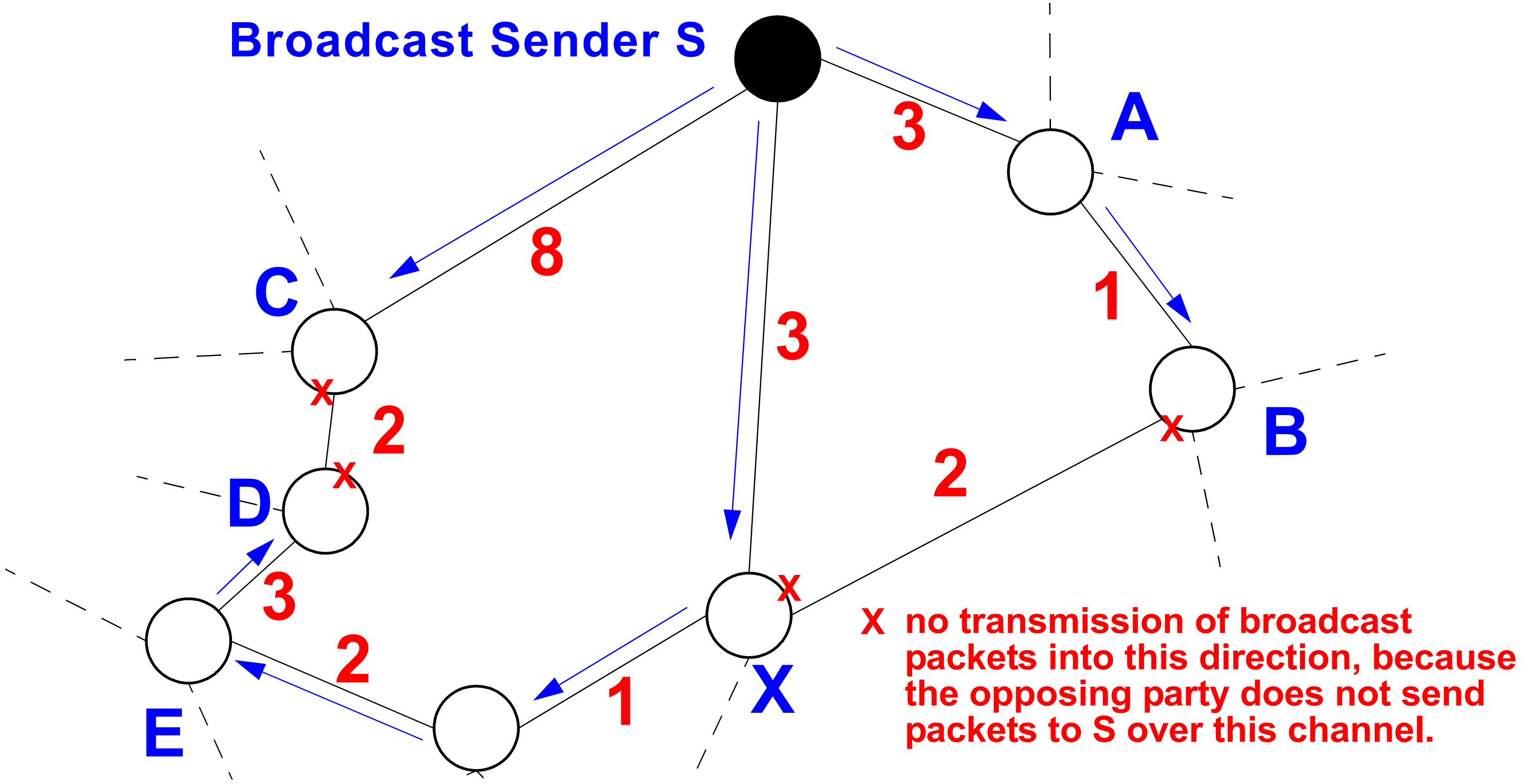
In the example

- **A can learn by inspecting the unicast packets**
  - that it is located on the unicast path from B to S
- **X can learn by packets failing to appear**
  - that it is not located on the unicast path from B to S

⇒ **This information is used by the RPB algorithm**

Broadcast Routing:Reverse Path Broadcast (RPB) (3)

Example:



X no transmission of broadcast packets into this direction, because the opposing party does not send packets to S over this channel.

In the example with the RPB algorithm

- **X does not forward a broadcast packet from S to B, because X knows**
  - that B does not receive unicast packets via X
  - but sends them over a different node instead with
  - this other node then receiving the broadcast packet

⇒ **Connection X-B relieved in comparison to the RPF algorithm**

Broadcast Routing:Reverse Path Broadcast (RPB) (4)

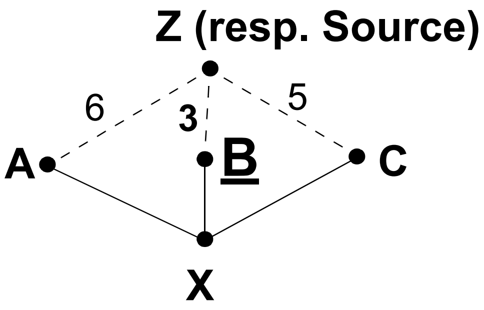
Comment:

- **when distance is the same:**
  - IS with the shortest address is selected
- **IS utilize the routing information,**
  - to exploit this parent-child relationship

(BELOW ANOTHER EXPLANATION)

Principle

- **as in REVERSE PATH FORWARDING**
  i.e., only packets which arrived over the "best" path are forwarded, but...
  - collision avoidance
    (additional discarding of packets)
    by defining a **PARENT-CHILD RELATIONSHIP**
  - provided that knowledge of the Spanning Tree exists
- **or parent-child relationship:**
  - IS B is the parent of the adjacent IS X,
    IF its distance to source Z
    is shorter than the distances of all other neighbours of X
    - (in the example: B is parent of X)



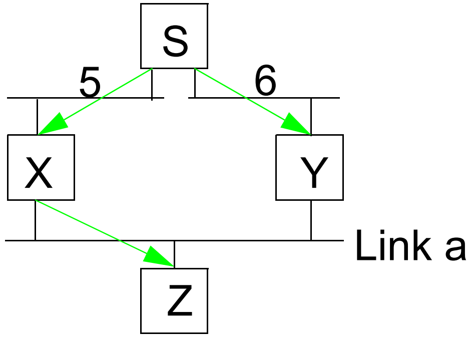
Broadcast Routing:Reverse Path Broadcast (RPB) (5)

Algorithm for selecting the outgoing links

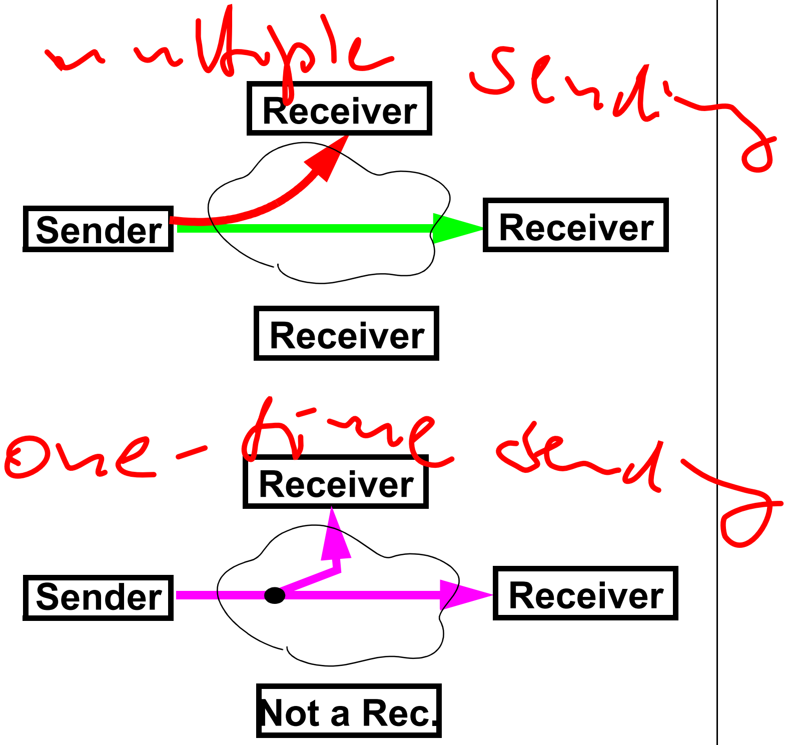
- **X is the PARENT of a link,**
  - IF its distance to the source is shorter than Y’s distance (or than all other ones)
- **if distance is the same:**
  - decision is based on the shorter address
- **router exchange routing informationen with each other to determine parent-child relationship**

Example

- **link a is the child of X, not of Y**
- **PACKETS FORWARDED ONLY OVER CHILD LINKS (this results in the Spanning Tree)**



6. Multicast Routing



Multicast Definition

- **Unicast: 1:1 communication**
- **Multicast: 1:n communication**

Tasks

- **to send data to a group of end systems**
  - one-time sending instead of
  - multiple sending
- **to maintain the overall load at a low level**

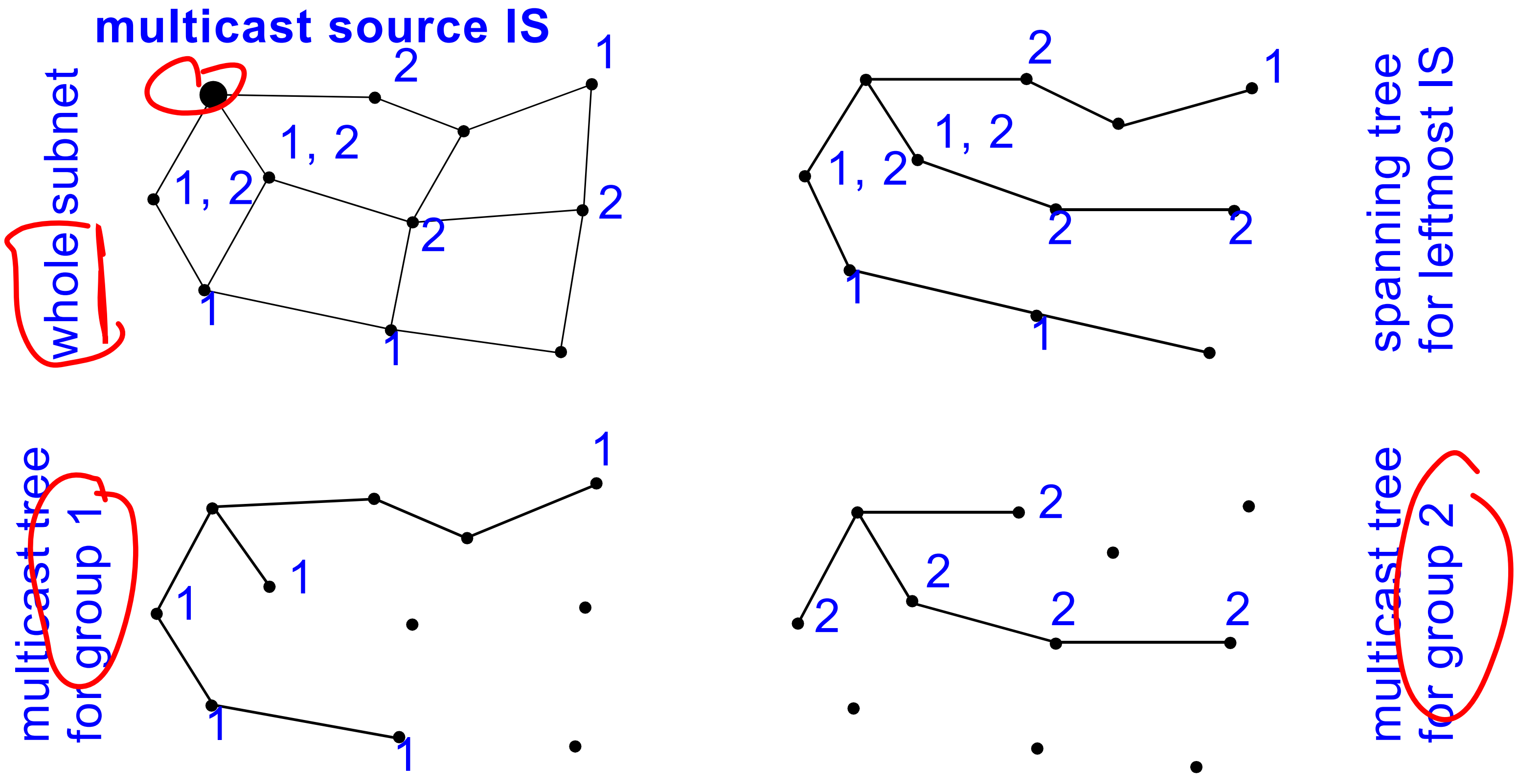
Results

- **lower network load**
- **lower load on the sender**

Condition: group addressing

- **group membership may change, managed for example by:**
  - Internet Group Management Protocol (IGMP)
    - group management (create, destroy, join, leave)
  - somehow related protocols for session maintenance
    - Session Description Protocol (SDP)
    - Session Announcement Protocol (SAP)
    - Session Initiation Protocol (SIP)

6.1 Multicast Routing: Spanning Tree


multicast source IS
whole subnet
spanning tree for leftmost IS
multicast tree for group 1
multicast tree for group 2
1, 2
1, 2
2
1
2
2
1
1
1, 2
1, 2
2
1
2
2
1
1
1
1
1
1
1
2
2
2
2
2

Principle

- **global knowledge of the multicast group's spannig tree (Multicast Tree),**
- **initially only local knowledge**

Distribution of Information

- **first IS adapts spanning tree to the specific group i.e. aligning (propagating) the spanning tree by**
  - distance vector routing or link state routing

Multicast Routing: Spanning Tree with Link State Routing

Principle: all IS must know the multicast tree

- **i.e. each IS**
  - **KNOWS TO WHICH GROUP IT BELONGS**
  - but **DOES NOT** know (initially) which other IS belong to the group as well
- **distribution of this information**
  - depends on the underlying routing protocol
  - here: Link State Routing

Link State Routing

- **all IS send link state packets periodically**
  - containing information
    - distance to neighbours
    - **EXPANDED** by information on multicast groups
  - by broadcast to all the others
- **each IS calculates a multicast tree**
  - from the now locally available and complete state information
- **based on the information about the multicast tree**
  - IS determines the outgoing lines
  - on which packets have to be transmitted

Multicast Routing: Spanning Tree with Distance Vector R.

Principle: all IS have to know the multicast tree

- **i.e. each IS**
  - **KNOWS WHICH GROUP IT BELONGS TO**
  - but **DOES NOT** know (inititially) which other IS also belong to the group
- **distribution of this information**
  - depends on the underlying routing protocol
  - here: Distance Vector Multicast Routing Protocol DVMRP

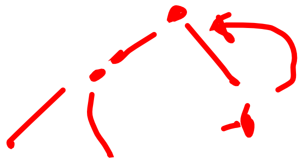
Method: REVERSE PATH FORWARDING WITH PRUNING

- **(Pruning: feedback in order to stop data transfer)**

Multicast Routing: Spanning Tree with Distance Vector R.(2)

Principle

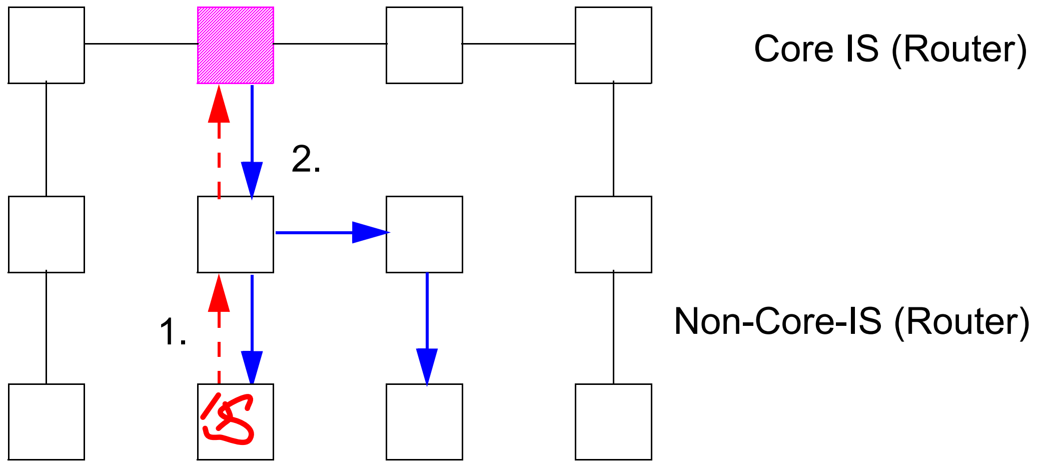
- **sender sends first multicast packet to everybody, using the broadcast method Reverse Path Forwarding RPF**
- **then apply adaptation (PRUNING)**
  - because broadcasting too resource consuming
- **from broadcast communication to the multicast structure**
- **originating from the leaves of the spanning tree:**
  - if multicast packet arrives from IS leaf **NOT** belonging to multicast group:
    - send a **NON-MEMBERSHIP-REPORT** (NMR) to the immediate predecessor
    - propagate a Non-Membership-Report NMR, if
      1. IS receives Non-Membership-Reports NMRs from all descendents
      2. but does not belong to the group itself
  - if multicast packet arrives from IS leaf which **DOES BELONG TO** multicast group:
    - nothing happens on the IS side
      (i.e., the following multicast packets are also send there again)



Benefit:

- **pruning only on trees that are actually used**
  - unused trees are cut coarsely
- **optimized for many receivers**

6.2 Multicast Routing: Core-Based Tree



- **also know as "Trees with Rendezvous Points"**

Principle

- **the CORE is selected (an IS which is central to the group)**
- **the group's spanning tree from this node/IS is determined**
- 1 **the sender transmits a packet to this central IS**
- 2 **the core transmits this packet via the spanning tree**

Properties

+ **simple central calculation**
+ **one tree common to all n senders (instead of n trees)**
- **route to the central IS may not be optimized**

IBR (Institut für Betriebssysteme und Rechnerverbund) – TU Braunschweig

Kommunikationssysteme: Network Layer

6.3 Multicast Routing: Additional Procedures & Topics

Variants (some additional ones)

- **Truncated Reverse Path Forwarding (TRPB)**
  - enhancement of broadcast procedure "Reverse-Path-Broadcast"
- **Steiner Trees (optimizing network resources)**
- **Reverse Path Multicast (RPM)**
- **Distance Vector Multicast Routing Protocol (DVMRP)**
  first version of DVMRP (RFC 1075) based on RPM
- **hierarchical DVMRP**
  - two-tiered, non-overlapping domains/subnetworks
- **Multicast Open Shortest Path First (MOSPF)**
  - based on link state routing OSPF
- **Protocol Independent Multicast (PIM)**
  - for groups with small spatial density

PIM - DM
PIM - SM

Multicast Routing: Additional Procedures & Topics (2)

Objectives: optimizations / constraints

- edge optimization: e.g., path with largest bandwidth
- edge limited: e.g., find a path that adheres to the constraints at every edge
- path optimization: e.g., path with the lowest overall costs
- path limited: e.g., path which does not exceed certain overall delay

Reserving resources

- Resource Reservation Protocol (RSVP)
- Stream Protocol Version 2 (ST-2)

Quality of Service

- negotiation
- with heterogenous receivers (filtering)
- adaptation (scaling)

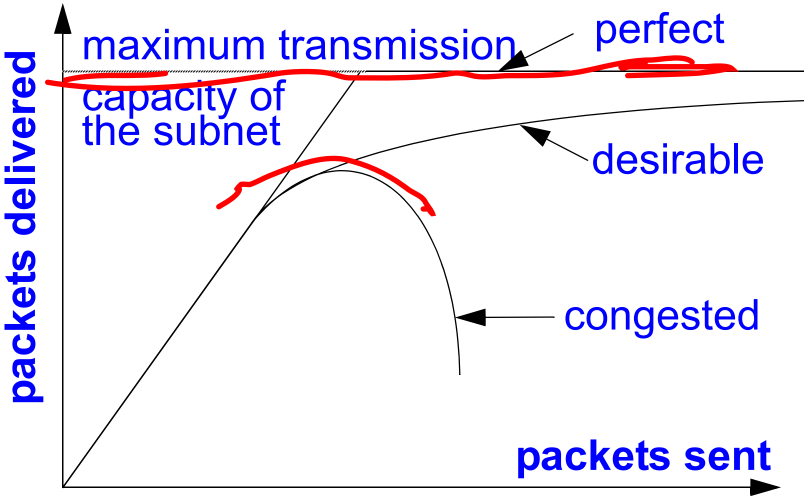
7. Congestion Control

When too much traffic is offered:

- **congestion sets in**
- **performance degrades sharply**

Reasons for congestion, among others

- **IS too slow for routing algorithms**
- **incoming traffic overloads outgoing lines**



Congestions tend to amplify themselves

Example: IS drops packet due to congestion

- **packet has to be retransmitted**
  - additional bandwidth used
- **sender cannot release the buffer**
  - thereby additionally tying up resources

congestion control vs.	flow control
managed by subnet (L3)	concatenated point-to-point (L2)
global issue	more an end-to-end issue
if possible, avoid from the beginning	reduce effects
may use flow control	

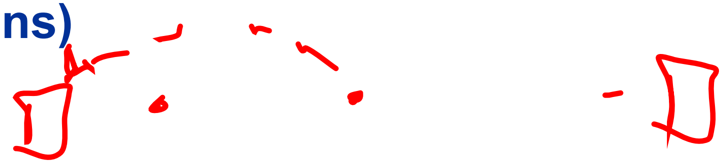
Congestion Control (2)

General methods of resolution

- **increase capacity**
- **decrease traffic**

Taxonomy according to Yang/Reedy 1995

- **open loop: avoid (before congestion happens)**
  - initiate countermeasures at the sender
  - initiate countermeasures at the receiver



- **closed loop: repair**
  - explicit feedback (packets are sent from the point of congestion)
  - implicit feedback (source assumes that congestion occured due to other effects)

Strategies

1. **avoidance**
   - traffic shaping, leaky bucket, token bucket, reservation (multicast), isarithmic congestion control
   - flow control (not discussed herein)
2. **repair**
   - drop packets, choke packets, hop-by-hop choke packets, fair queuing,...

7.1 Congestion Avoidance

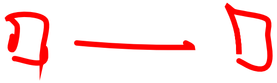
Principle

- appropriate communication system behavior and design

Policies at various layers can affect congestion

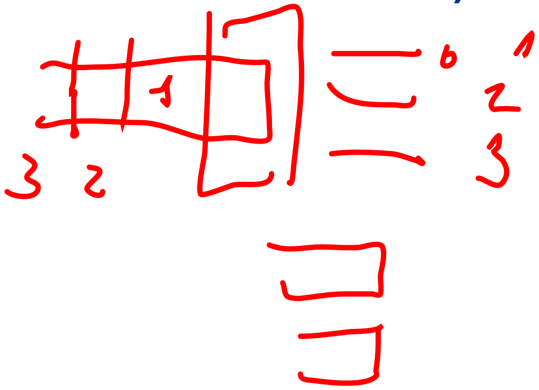
Data link layer

- flow control
- acknowledgements
- error treatment / retransmission / FEC



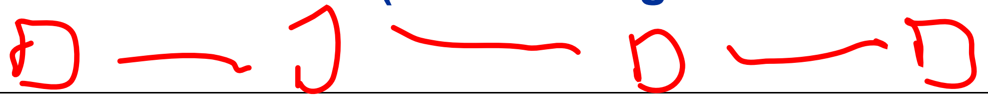
Network layer

- datagram (more complex) vs. virtual circuit (more procedures available)
- packet queueing and scheduling in IS
- packet dropping in IS (including packet lifetime)
- selected route

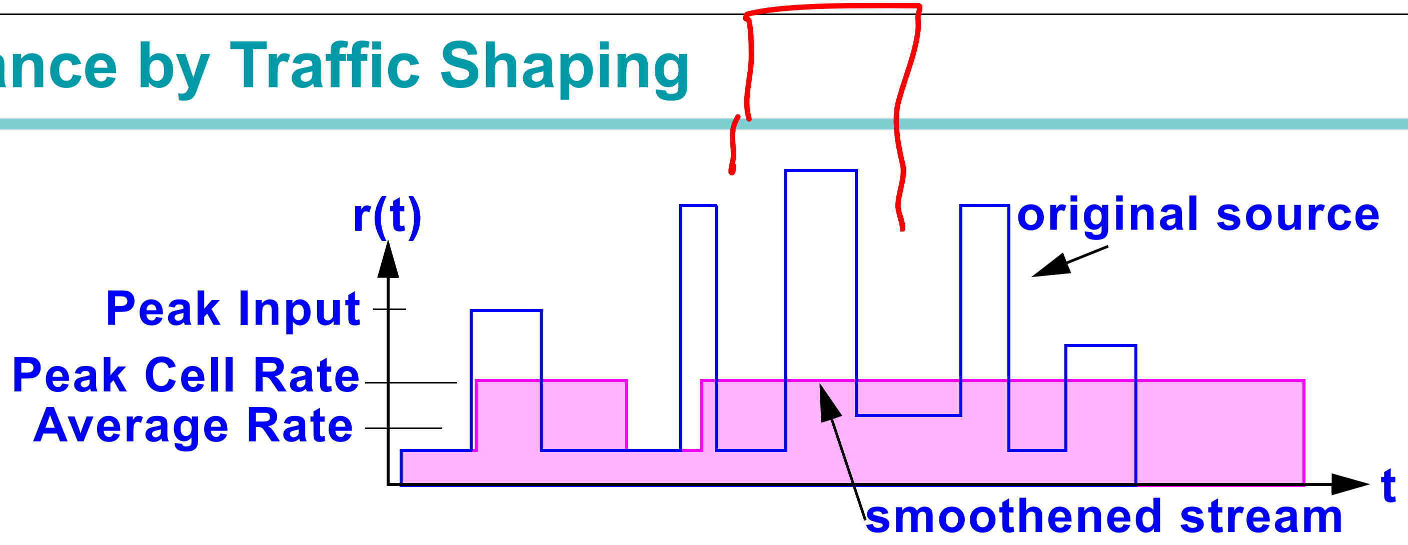


Transport layer

- basically the same as for the data link layer
- but some issues are harder (determining timeout interval)



Avoidance by Traffic Shaping



Motivation

- **congestion is often caused by bursts**
- **bursts are relieved by smoothening the traffic (at the cost of a delay)**

Procedure

- **negotiate the traffic contract beforehand (e.g., flow specification)**
- **the traffic is shaped by the end device**
  - average rate and
  - burstiness

Application: virtual circuits in ATM

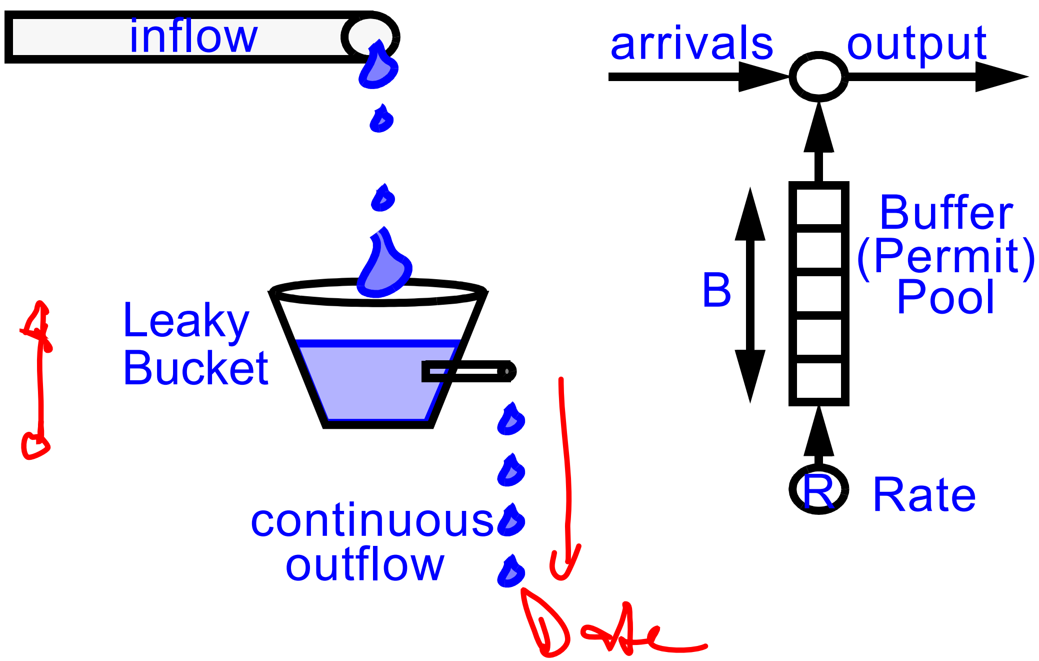
- **”traffic shaper” smoothens extremely fluctuating traffic**
- **trade-off: loss of cells vs. delay**

Traffic Shaping with Leaky Bucket

Principle

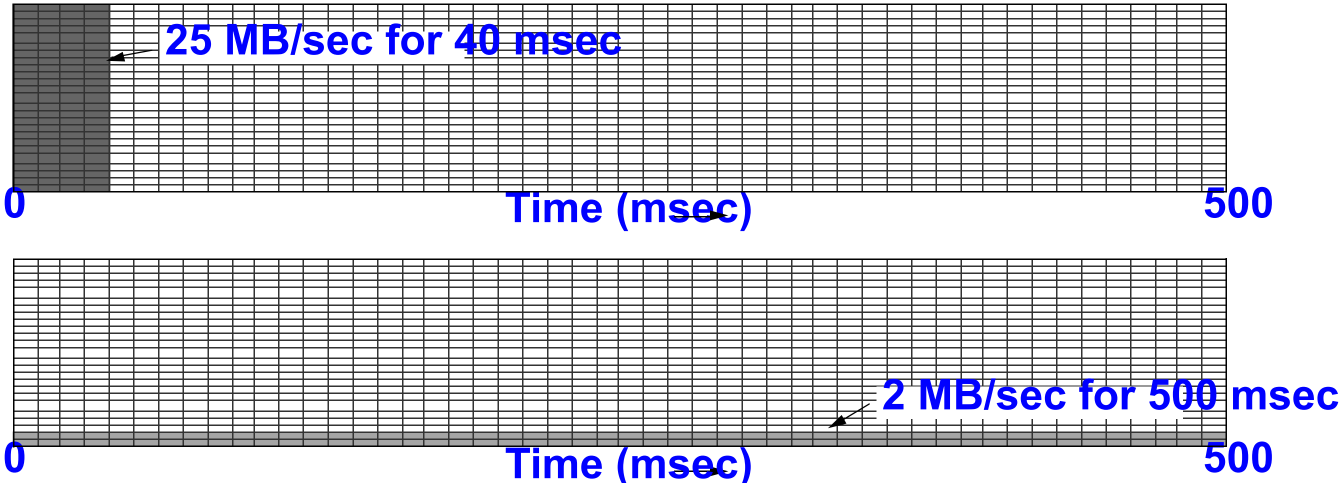
- **continuous outflow**
- **congestion corresponds to data loss**
- **1986: Turner**

Data



Implementation

- **easy if packet length stays constant (like ATM cells)**
- **example**



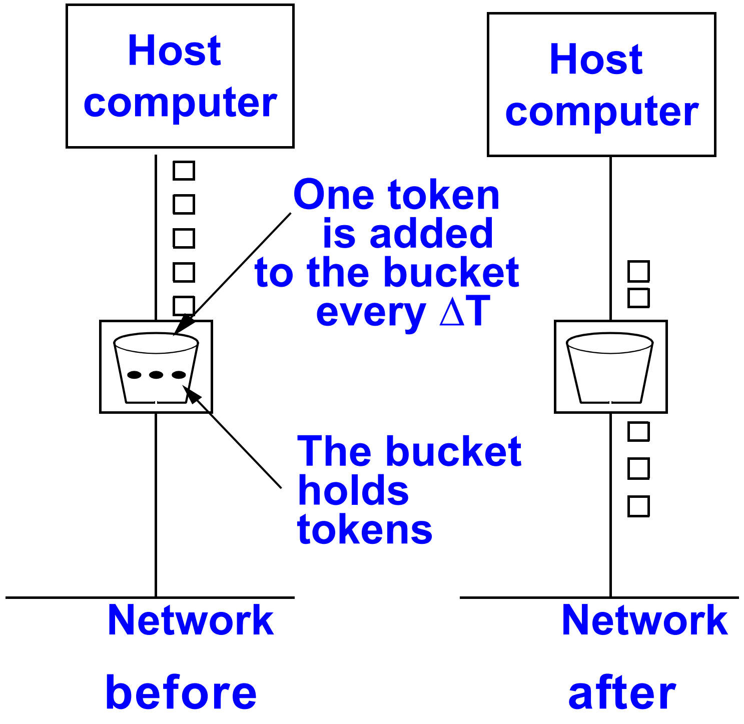
Traffic Shaping with Token Bucket

Principle

- permit a certain amount of data to flow off for a certain amount of time
- controlled by "tokens"
- number of tokens limited

Implementation

- add tokens periodically (until maximum has been reached)
- remove token: depending on the length of the packet (byte counter)



Comparison

- Leaky Bucket
  - max. constant rate (at any point in time)
- Token Bucket:
  - permits a limited burst

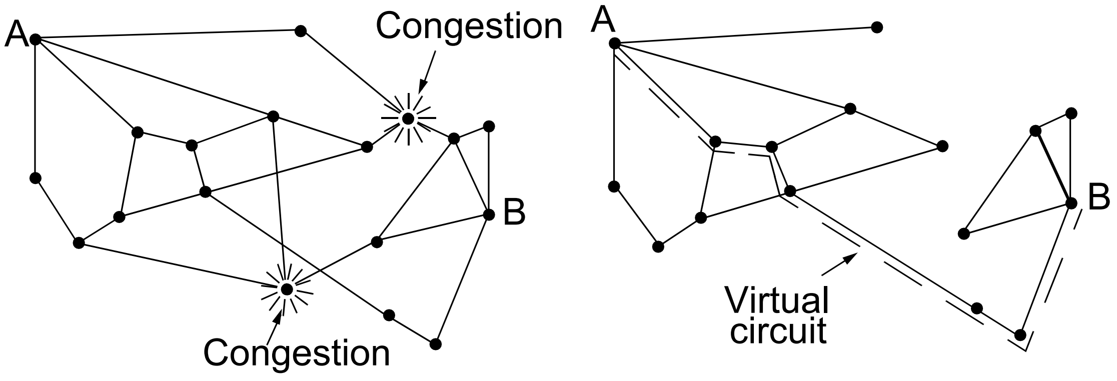
Avoidance by Reservation: Admission Control

Principle:

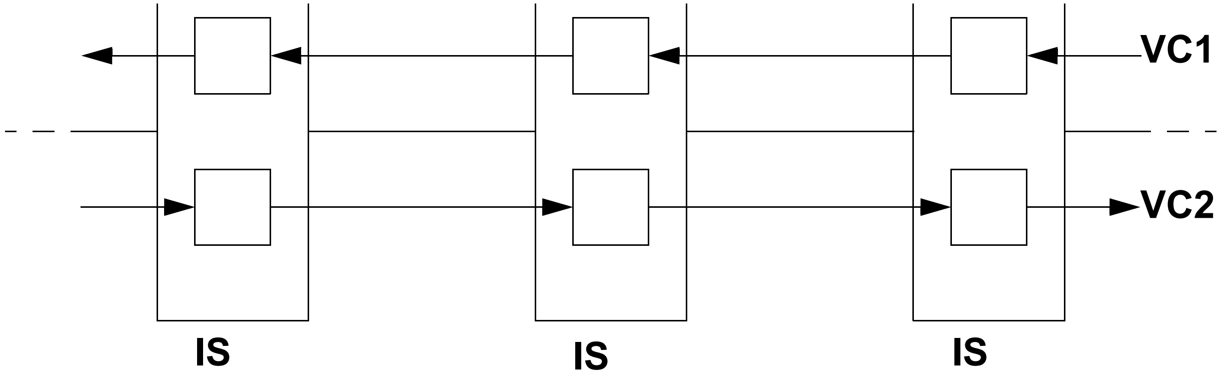
- **prerequisite: virtual circuits**
- **reserving the necessary resources (incl. buffers) during connect**
- **if buffer or other resources not available**
  - alternative path
  - desired connection refused

Example:

- **network layer may adjust routing based on congestion**
- **when the actual connect occurs**



Avoidance by Buffer Reservation



Principle:

- buffer reservation

Implementation variant: Stop-and-Wait protocol

- one buffer per IS and connection (simplex, VC=virtual circuit)

Implementation variant: Sliding Window protocol

- m buffer per IS and (simplex-) connection (m corresp. to the window size)

Properties:

- + congestion not possible
- – buffers remain reserved, even if there is no data transmission for some periods
- therefore, usually only for applications that require low delay and high bandwidth (e.g., digital voice transmission)

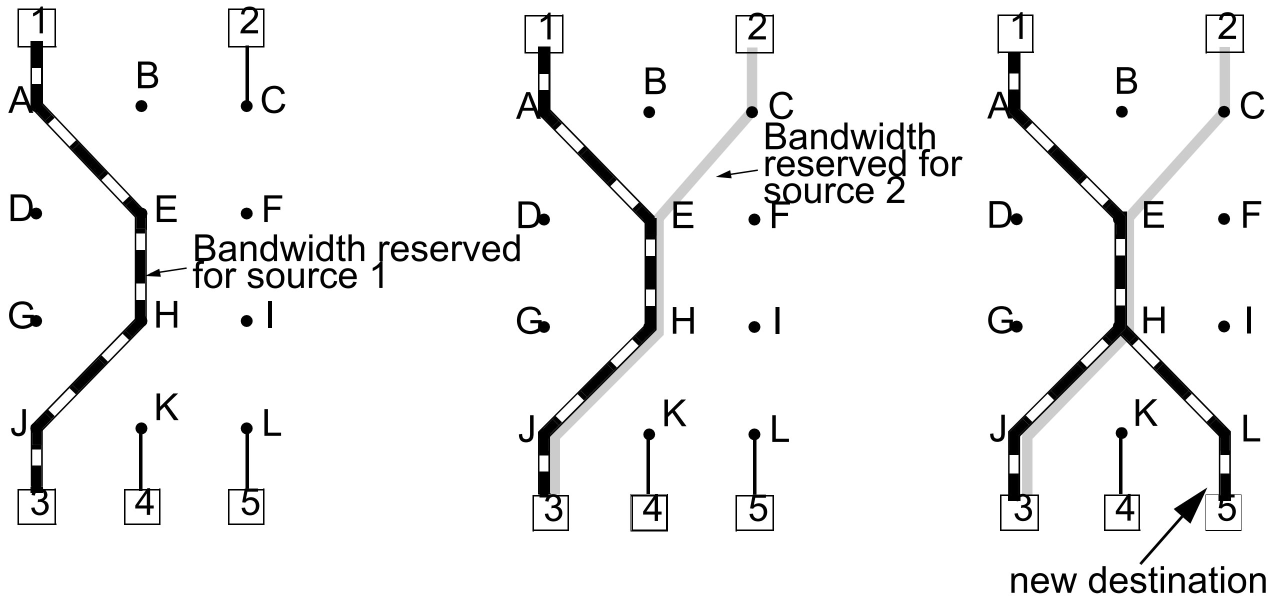
Avoidance by Reservation: Multicast and Time Guarantees

Reservation protocols

- **Resource Reservation Protocol (RSVP)**
- **Stream Type Protocol Version 2 (ST-2)**

Searching for the most ideal IS to connect to an MC group

Example



Avoidance by Isarithmic Congestion Control

Principle

- **limiting the number of packets in the network by assigning "permits"**
  - amount of "permits" in the network
  - a "permit" is required for sending
    - when sending: "permit" is destroyed
    - when receiving: "permit" is generated

Problems

- **parts of the network may be overloaded**
- **equal distribution of the "permits" is difficult**
- **additional bandwidth for the transfer of "permits" necessary**
- **bad for transmitting large data amounts (e.g. file transfer)**
- **loss of "permits" hard to detect**

7.2 Congestion Correction

Principle

- **no resource reservation**
- **necessary steps**
  - detect congestion
  - introduce appropriate procedures for reduction

Packet dropping

Principle:

- **incoming packet is dropped, if it cannot be buffered**

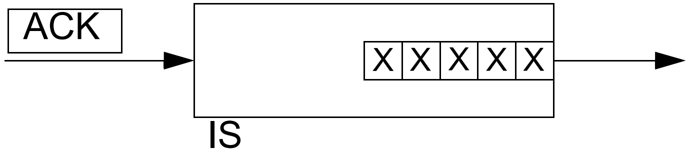
Preconditions for

- **datagram:**
  - no preparations necessary
- **connection-oriented service:**
  - packet will be buffered until receipt has been acknowledged

Buffer assignment methods

1. Permanent buffers per incoming line

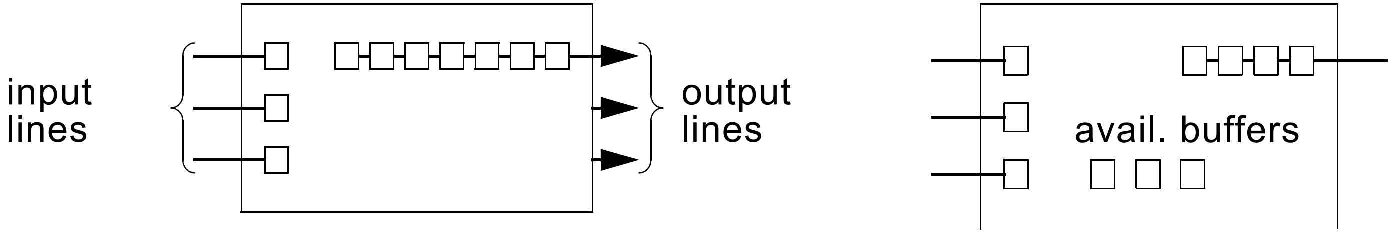
- **ACK**
  - would have to be discarded
  - during overwrite, ACK would release buffer
  - (comment: in the above picture the ACK refers to the 5 buffer slots on the right)



Packet dropping (2)

2. Maximum number of buffers per output line

- **example:**



  - packet dropped despite there are free lines

- **heuristic rule [Irland]**

$$m = \frac{k}{\sqrt{s}}$$

```
m     ... max. number of buffers per output line
k     ... total number of buffers
s     ... number of output lines
```

3. Minimal number of buffers per output line

- **line cannot be starved**

Example ARPANET

- **a combination of 2) and 3)**

Packet dropping (3)

4. Content-related dropping: relevance

- **reference**
  - data connection as a whole
    (or all single data packets from one end system to another end system)
  - single packets,
    examples
    - WWW document
      images vs. text and structural information
    - file transfer:
      old packets more important than new ones
      (algorithm to initiate correction process should start as late as possible)
- **implementation of priorities**
  - in virtual circuits or datagrams
  - example: ATM

Packet dropping (4)

Properties:

- **very simple**

but

- **retransmitted packets waste bandwidth:**
- **packet has to be sent 1 / (1 - p) times before it is accepted**
  - (p ... probability that packet will be dropped)

Optimization necessary to reduce the wastage of bandwidth

- **dropping packets that have not gotten that far yet**

Choke Packets

Principle:

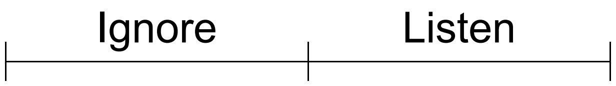
- **reduce traffic during congestion by telling source to slow down**

Procedure for IS: each outgoing line (OL) has one variable = utilization u

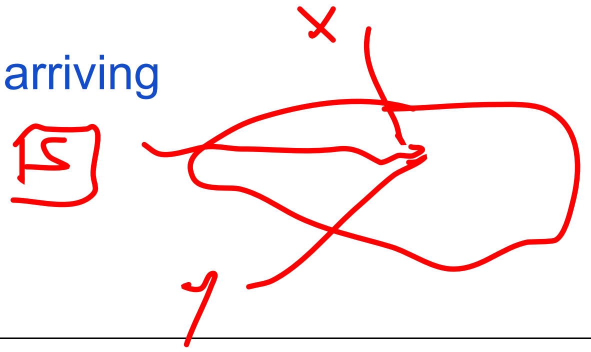
- **calculating u (0 ≤ u ≤ 1) (u: UTILIZATION)**
  - IS checks the line usage f periodically (= 0 or 1)
  - $\underset{t}{u} = a * \underset{t-1}{u} + (1 - a) * f$
  - $0 \leq a \leq 1$ determines to what extent "history" is taken into account
- **u > threshold: OL changes to condition "warning"**
  - send **CHOKE PACKET** to source (indicating destination)
  - tag packet (to avoid further choke packets from down stream IS) and forward it

Procedure for source

- **source receives the choke packet**
  - reduces the data traffic to the destination in question by X_1%
- **source recognizes 2 phases:**
  (gate time so that the algorithm can take effect)



  - Ignore: ES ignores further Choke packets
  - Listen: ES listens if more Choke packets are arriving
    - yes: further reduction by X_2%; go to Ignore phase
    - no: increase the data traffic



Choke Packets (2)

Enhancements

- **varying choke packets depending on state of congestion**
  - warning
  - acute warning
- **for u instead of utilization**
  - queue length
  - ....

Properties

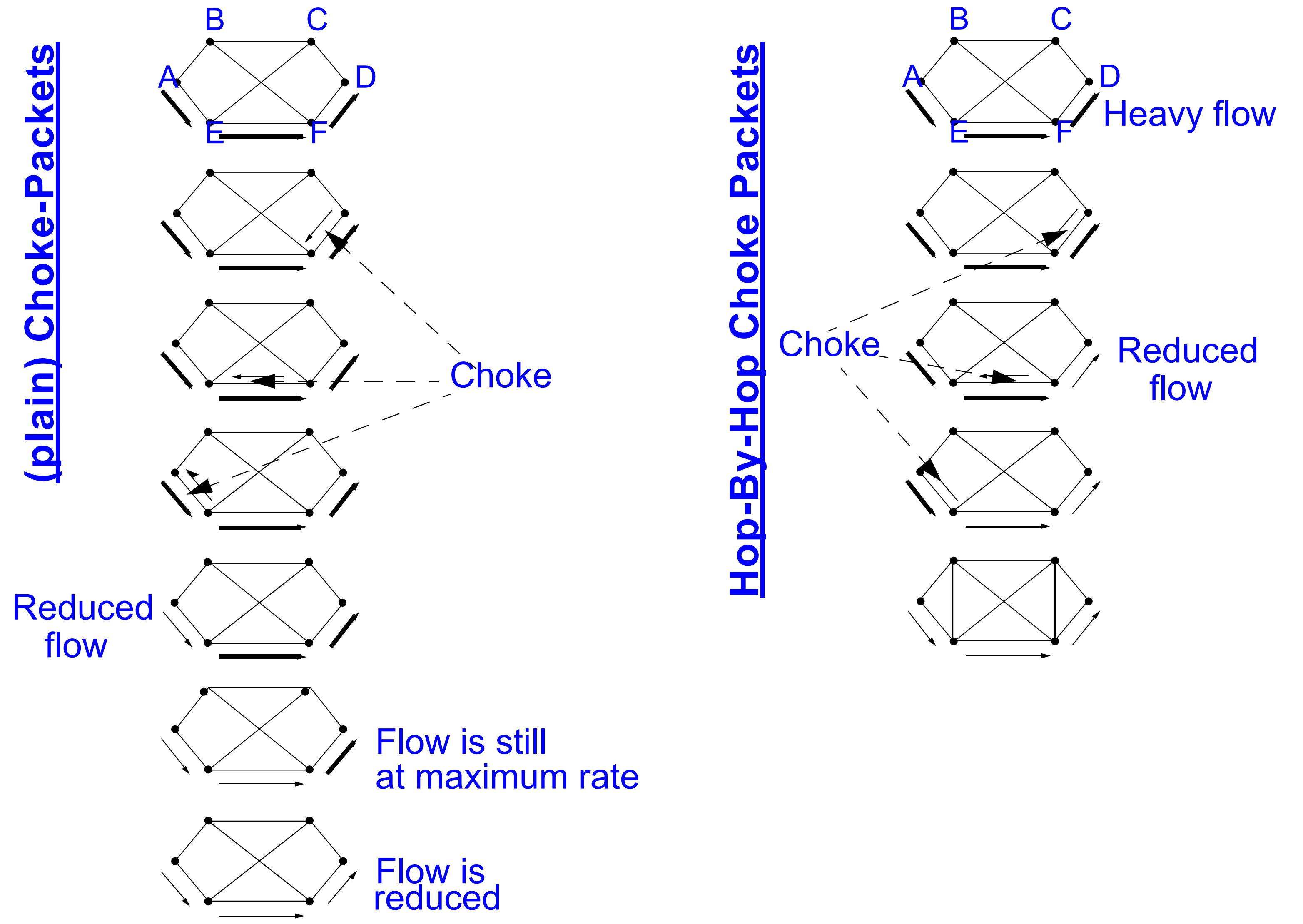
- **effective procedure**
- **but**
  - possibly many choke packets in the network
    - even if Choke bits may be included in the data at the senders to minimize reflux
  - end systems can (but do not have to) adjust the traffic
  - superimposed by mechanisms
    - L2 flow control, ...
    - L4 TCP, ..

Choke Packets: Hop-by-Hop Choke Packets (3)

Principle

- reaction to Choke packets already at IS (not only at ES)

Example



Random Early Detection (RED)

Idea:

- **Congestion should be attacked as early as possible**
- **some transport protocols (e.g., TCP) react to lost packets by rate reduction**

IS drops some packet before congestion significant (i.e., early)

- **gives time to react**

Dropping starts when moving avg. of queue length exceeds threshold

- **small bursts pass through unharmed**
- **only affects sustained overloads**
- **Packet drop probability is a function of mean queue length**
  - prevents severe reaction to mild overload

Can mark packets instead of dropping them

- **allows sources to detect network state without losses**

RED improves performance of a network of cooperating TCP sources

No bias against bursty sources

Controls queue length regardless of endpoint cooperation

8. Addressing

3 types of identifiers: names, addresses and routes [Shoch 78]

> **"The NAME of a resource indicates WHAT we seek,**
>
> **an ADDRESS indicates WHERE it is, and**
>
> **a ROUTE says HOW TO GET THERE."**

Objectives:

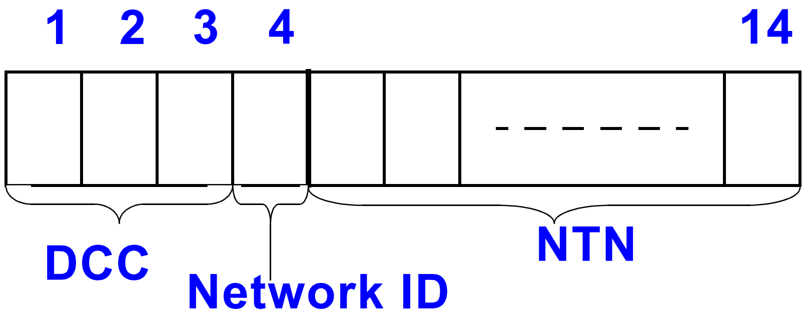
- **global addressing concept for ES**
- **simplified address allocation**
- **addresses independent from**
  - type and topology of the subnetworks
  - number and type of the subnetworks to which the ES have been connected
  - location of a source ES

8.1 X.121 Addressing

CCITT/ITU "numbering scheme"

- **addressing concept for public data networks**
- **a.o., used by X.25**

X.121 address:



- **a maximum of 14 digits**
- **consisting of**
  - Data Network Identification Code (4 digits)
    - Data Country Code (digits 1 - 3)
    - Network Identification (digit 4)
  - Network Terminal Number (max. 10 digits)

Example:

DCC for USA: 310 - 329, i. e. max. 200 networks

DCC for Tonga: 539, i. e. max. 10 networks

8.2 OSI Addressing

Objective:

- **global addressing concept for both existing and new subnetworks**

Situation: different concepts exist for

- **public networks:**
  - X.121: data networks
  - F.69: telex
  - E.163: telephone network
  - E.164: ISDN, ...
- **private networks**

⇒ **i.e., a flexible and expandable concept is necessary**

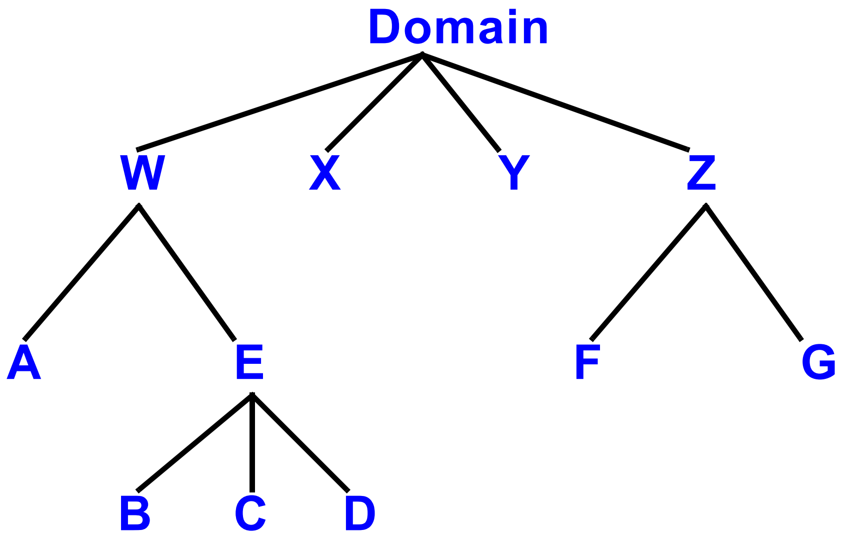
OSI method: unique NSAP identification

OSI method: hierarchic addresses

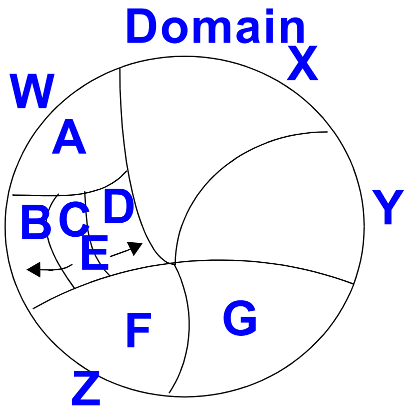
- **OSI defines the ADDRESSING DOMAINS**
- **the domain contains the ADDRESSING AUTHORITY**
- **Addressing Authority**
  - allocates addresses
  - creates new domains and delegates authority

OSI Addressing (2)

graphic representation of the domain hierarchy:



or



A domain may be

- **networks of one type**
- **networks of a geographical region**
- **networks of an organisation**
- **...**

OSI Addressing: Structure (3)

Address length: 20 bytes (binary) or 40 digits

Address structure:

- **Initial Domain Part (IDP) with**
  - **AUTHORITY AND FORMAT IDENTIFIER** (AFI)
    - specifies how to interpret the IDI (syntax and semantics)
    - e.g. the format of the DSP (binary or digits)

IDI Format	DSP SYNTAX	
	Decimal	Binary
X.121	**36**	**37**
ISO DCC	**38**	**39**
F.69	**40**	**41**

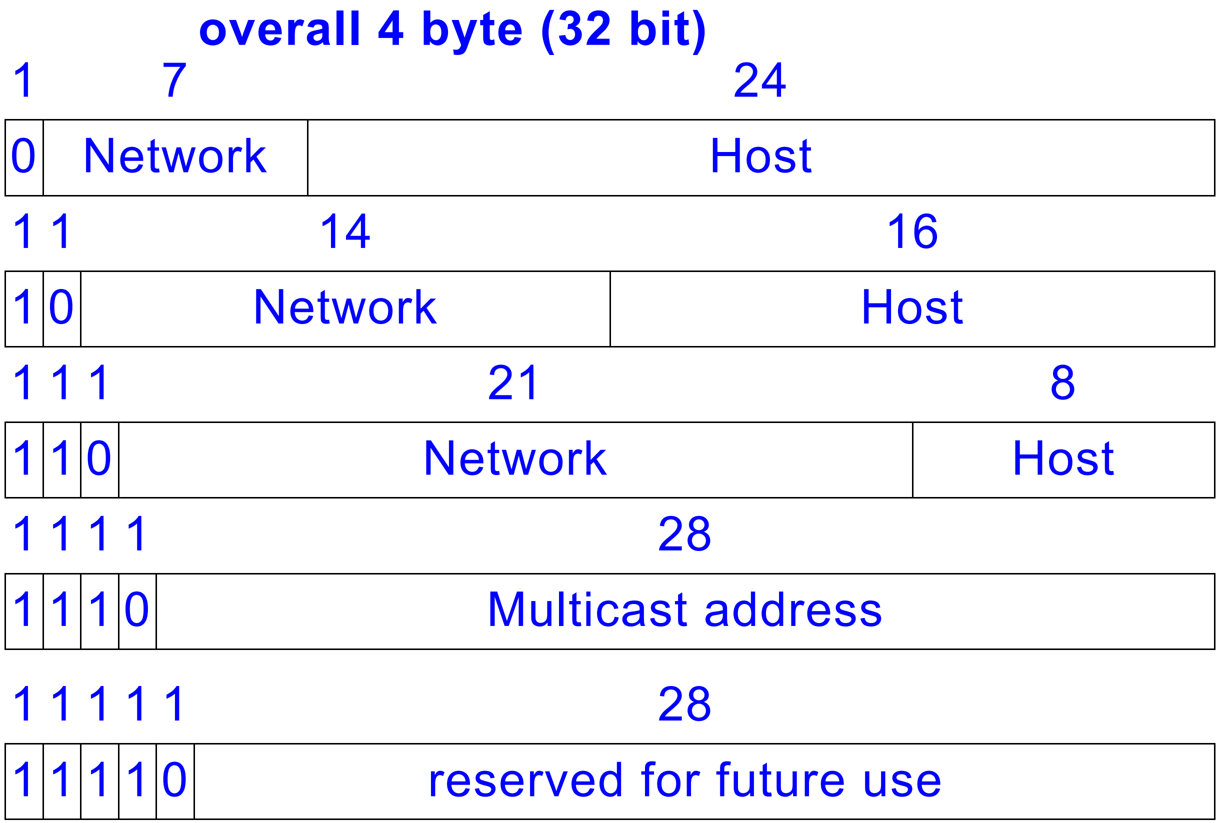
Character	National Character
50	51

  - **INITIAL DOMAIN PART** (IDI)
    - identifies the Addressing Authority (AA), responsible for **ALLOCATING THE NSAP ADDRESSES**
    - identifies the domain
- **Domain Specific Part (DSP)**
  - contains the address clearly identifying the ES within the domain

8.3 Internet Addresses (IP)

Global addressing concept for ES (and IS) in the Internet

- **32 bit address (amount is limited!)**
- **each address is unique worldwide**
- **structure: Net-ID (Subnet-ID), ES-ID**



overall 4 byte (32 bit)

1 7 24

0 | Network | Host

1 1 14 16

1 | 0 | Network | Host

1 1 1 21 8

1 | 1 | 0 | Network | Host

1 1 1 1 28

1 | 1 | 1 | 0 | Multicast address

1 1 1 1 1 28

1 | 1 | 1 | 1 | 0 | reserved for future use

Internet Addresses (IP) (2)

Notation

- decimal value for each byte (0...255)
- subdivided by dots
- value range: 0.0.0.0 ... 255.255.255.255

Formats: 5 classes

A:	1.0.0.0	up to 127.255.255.255
B:	128.0.0.0	up to 191.255.255.255
C:	192.0.0.0	up to 223.255.255.255
D:	224.0.0.0	up to 239.255.255.255 (Multicast)
E:	240.0.0.0	up to 247.255.255.255

Broadcast addresses: (convention: 11...1 for Host-ID)

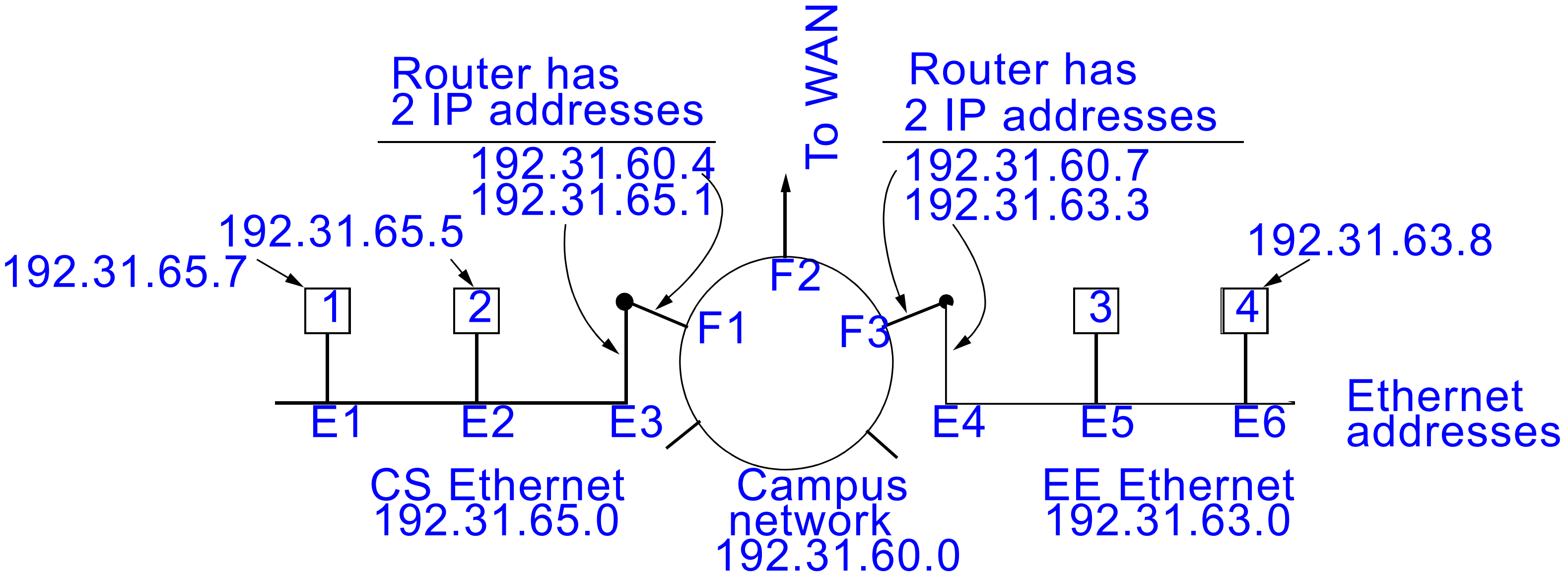
Internet Addresses (IP) (3)

Address allocation

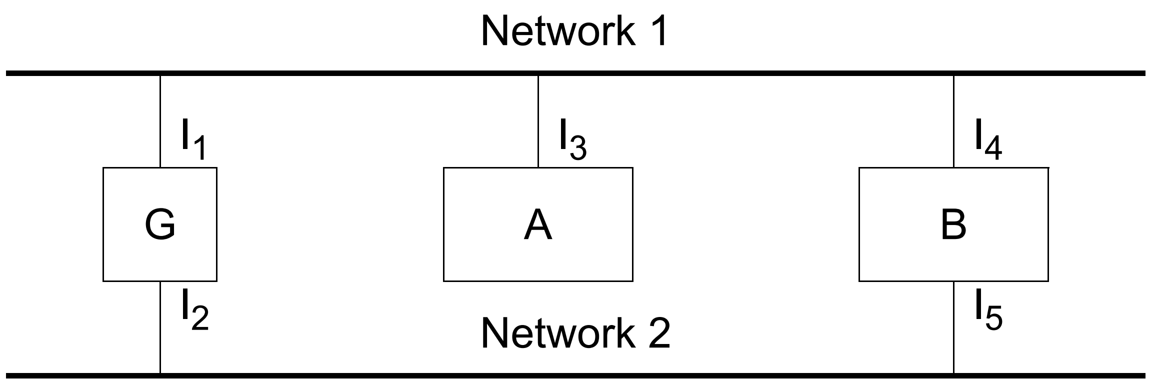
- **class allocation and network range:**
  - by a central authority
  - Network Information Center NIC
- **end system**
  - local
  - possibly forming a subnetwork

Example

- **network**



Internet Addresses (IP): A Critical Review (4)



Addresses IDENTIFIY "NETWORK CONNECTIONS", not the ES

- **"multi-homed" ES have more than one address**
- **a change of the connection forces the modification of the address**
- **the address has an impact on the chosen route (constitutes a problem in the mobile area)**

Example: A cannot reach B via address I_5 if G fails

- **comment: is also valid for X.121**

Amount of addresses

- **limited**

Internet Addresses (IP): The Future (5)

IP Version 6 (IPv6)

- 16 byte length (instead of 4 byte length, i.e. approx. 3 x 10^{38})

Distribution

- provider-based: approx. 16 mio. companies distribute addresses
- geographic-based: distribution as it is today
- link, site-used: address relevant only locally (security, Firewall concept)

e.g. new: Anycast

- sending data to an individual of a group
- e.g. the one who is geograhically the closest

Internet Addresses (IP): The Future (6)

Prefix (binary)	Usage	Fraction
0000 0000	Reserved (including IPv4)	1/256
0000 0001	Unassigned	1/256
0000 001	OSI NSAP addresses	1/128
0000 010	Novell Netware IPX addresses	1/128
0000 011	Unassigned	1/128
0000 1	Unassigned	1/32
0001	Unassigned	1/16
001	Unassigned	1/8
010	Provider-based addresses	1/8
011	Unassigned	1/8
100	Geographic-based addresses	1/8
101	Unassigned	1/8
110	Unassigned	1/8
1110	Unassigned	1/16
1111 0	Unassigned	1/32
1111 10	Unassigned	1/64
1111 110	Unassigned	1/128
1111 11100	Unassigned	1/512
1111 111010	Link local use addresses	1/1024
1111 111011	Site local use addresses	1/1024
1111 1111	Multicast	1/256