Foundations …

… at a glance

General requirements

- So, you want to build a network ...
  - First you need to know the **requirements** the network must satisfy
  - The requirements vary depending on who you ask (different views)
- Requirements from different views:
  - Application programmer: service specific needs, e.g., packets sent should not get lost – without errors – within a certain amount of time - and should arrive in the same order.
  - Network designer: “cost effective” design, efficient and fair usage of network resources
  - Network provider: a system that is easy to administer and manage, reliable, faults can be easily isolated
  - Users expect services: e-mail, tele- and videoconferencing, e-commerce, video-on-demand, ...

Computer network characteristics

- Typically communications networks optimized for some service
  - telephone network
  - television/radio broadcast network
  - user terminals special purpose devices
- Modern computer networks are more general:
  - terminals are general purpose PCs/workstations
  - networks can carry any kind of data
  - support many different applications
- Topics in this lecture
  - How computer networks **provide connectivity** (Requirement 1)
  - How efficient **resource sharing** is achieved (Requirement 2)
  - How applications **“talk”** to each other (Requirement 3)
  - How **network performance** affects the system (Requirement 4)

Requirements are **reflected in network architectures**
Basically, we get a **“snap shot”** of the issues covered in this course

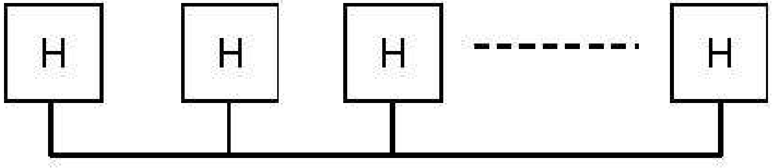
Outline

- Achieving connectivity
- Methods for resource sharing
- Enabling application level communication
- Performance issues
- Network architecture

Basic building blocks

- A network, in principle, consists of **nodes** and **links** connecting the nodes.
- Network nodes: PCs, servers, special purpose hardware
  - Internet terminology
    - **hosts**, end-systems: PCs and servers running network applications
    - **routers** (switches): store and forward packets through the network
    - **Links**: optical fiber, coaxial cable, twisted pair copper, radio, etc.
  - point-to-point
    - hosts directly connected

  - multiple access (LANs, etc.)
    - hosts share the common transmission medium



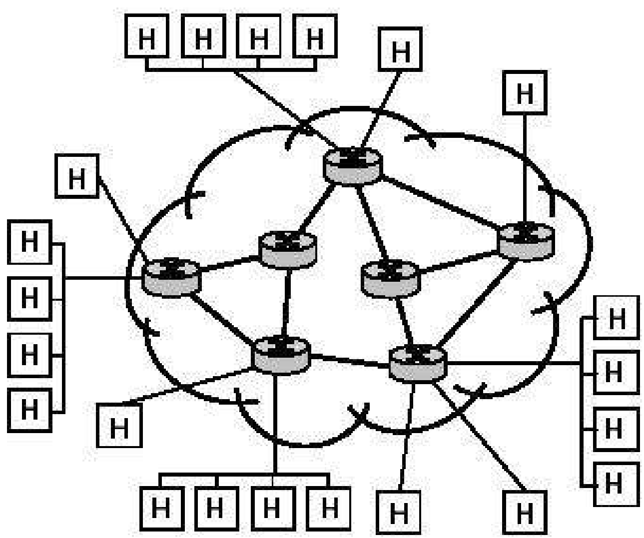
Building larger networks

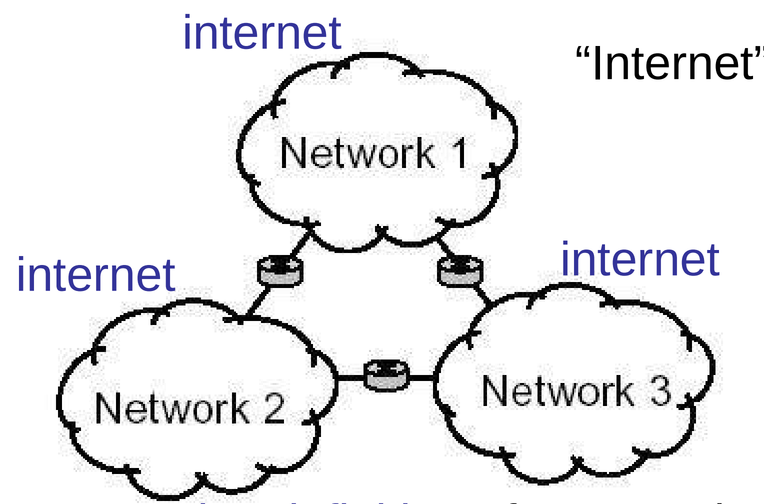
- Large networks can not be built based on point-to-point connectivity

=> use routers (switches) to interconnect hosts to each other

Nodes connected together through switches to form connected networks

Networks connected together through gateway routers to form bigger entities



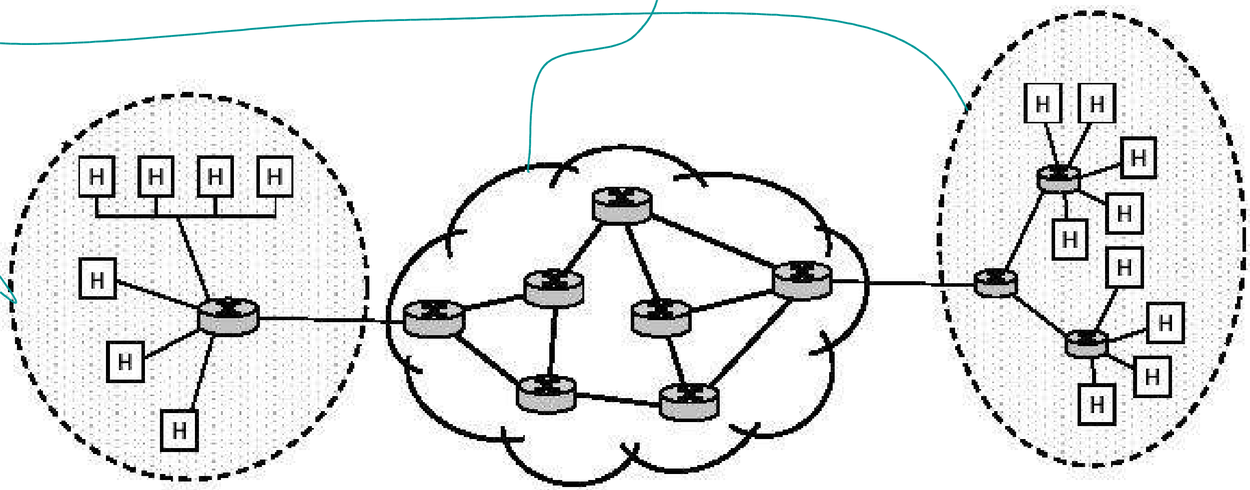


Recursive definition of a network
Starting at the bottom: physical link

Network edge vs. core network in the Internet

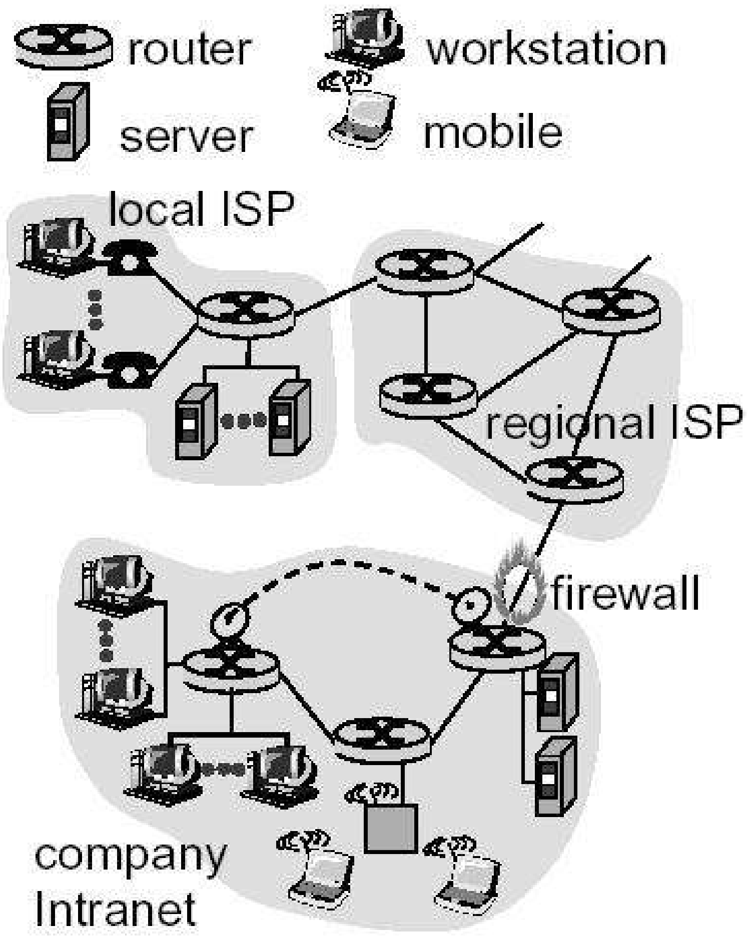
- **Access network**
  - customers are connected to the core network by the **access network**
  - link speeds comparably low
- **access technologies:** dial up (*modem over twisted pair*), xDSL, cable modem, ...
  - may contain **billing functionality**, traffic management for each access
  - tree topology

- **Core network**
  - no end users directly connected to the core
  - high link speeds
- **SDH/SONET** over fiber based technologies
  - simple functionality (forwards packets)
  - mesh topology



Internet

- Consists of millions of hosts (end systems) connected by links and routers
- **Hosts** exchange messages by using **protocols** offering e.g.
  - – reliable transfer
  - – packet sequence integrity
- **Routers** forward data
  - – based on best effort service
  - – *no guarantees on loss or timeliness*
- "**Network** of networks"
  - – loosely hierarchical
  - – public Internet vs. private intranets
  - – Internet access provided by ISPs (Internet Service Providers)



Issues of scale

Easy to build and manage a network supporting 100 users, but what if the number of users is 100 million ...

- A system allowing unlimited growth in size is said to **scale**.
  - Scalability a very desirable property for networking technologies
- Scalability of networks is often influenced very much by …
  - the nature of the guarantees regarding service quality
  - the amount of information that the network has about the users
- One reason for the success of Internet technology is its scalability
  - The networking paradigm is based on best-effort service (*no guarantees* are made *about the service quality*) and the network is *connectionless*
  - The **nodes** of the network **do not store any state information** of the users/connections
  - New **nodes and users can be added** to the network (*almost*) without any complexity increases
  - **Only the routing is affected** by the increase in the number of nodes (route computation complexity grows with the number of nodes)

? Switching modes

- **Circuit switching** *quick transmission, must arrive in sequencing order, constant arrival rate, real time data, such as audio and video*
  - telephone networks (PSTN)
  - (mobile telephone networks)
- **Packet switching**
  - public data networks (PDN).
  - two possibilities
    - **connectionless** (d*atagram packet switching*): e.g. Internet (IP), SS7 (MTP)
    - **connection oriented** (*virtual circuit networks*): e.g. X.25, Frame Relay

  ATM technology is also based on *virtual circuit switching*

    - fast packet switching with fixed length packets (cells): ATM
    - integration of different traffic types (voice, data, video)

    =>*multiservice networks*

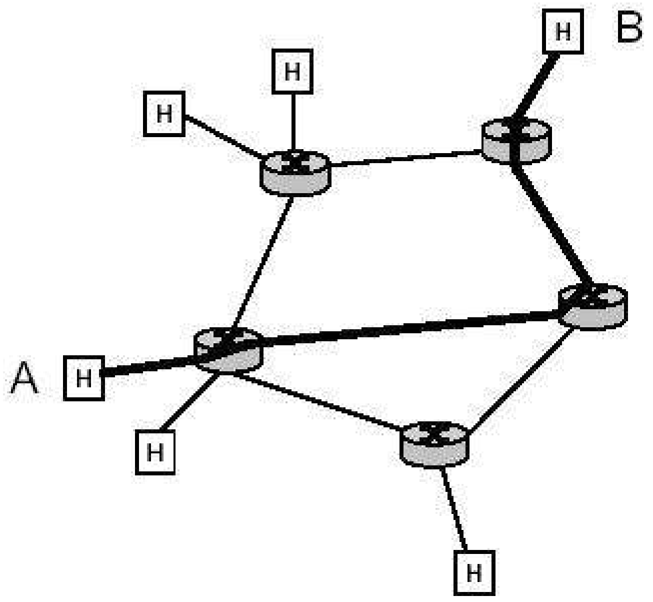
Circuit switching

Connection oriented:

– connections **set up** end-to-end before information transfer

– resources **reserved** for the whole duration of connection

- Information transfer as a **continuous stream**
- Before information transfer
  – delay (to set up the connection)
- During information transfer
  – no overhead
  – no extra delays



(Connectionless) packet switching

Connectionless:

– no connection set-up

– no resource reservation

- Information transfer by using **discrete packets**

– varying length

– global address (of the destination)

- Before information transfer

– no delays

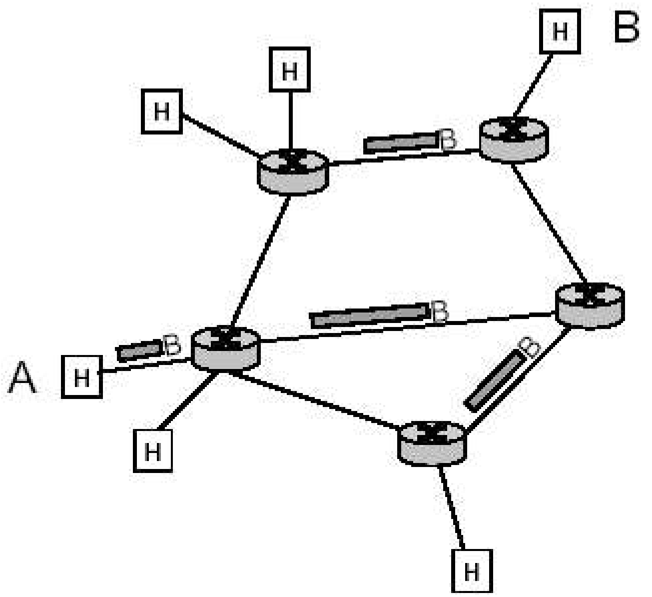
- During information transfer

– overhead (header bytes)

– packet processing delays

– queuing delays (since packets compete for shared resources)

– routers “store-and-forward”



Addressing and routing

- Hosts need to distinguish each other when wishing to communicate
- Each host is assigned a unique byte-string known as **address**
- When a sender "A" communicates with some destination "B", in a packet switched network
  - the address of the destination "B" **is attached** to each packet, and
  - each router determines how to forward the packet based on the destination address
  - **routing** is the systematic **process of determining where a packet is sent** (which output port) based on the destination address

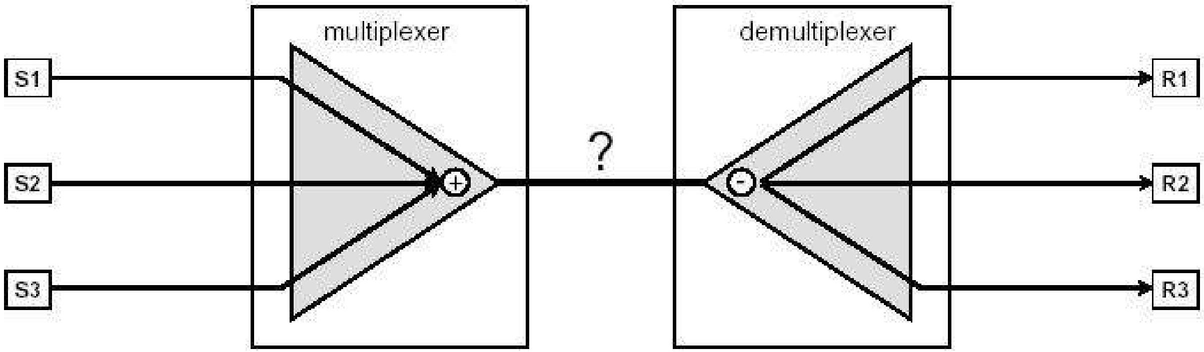
- Different addressing and routing scenarios
  - **unicast**: between a single sender and destination pair
  - **broadcast**: from a single user **to all other** users (e.g. network control messages)
  - **multicast**: from a single user **to a subset of all users** (e.g. distribution of files)

Outline

- Achieving connectivity
- Methods for resource sharing
- Enabling application level communication
- Performance issues
- Network architecture

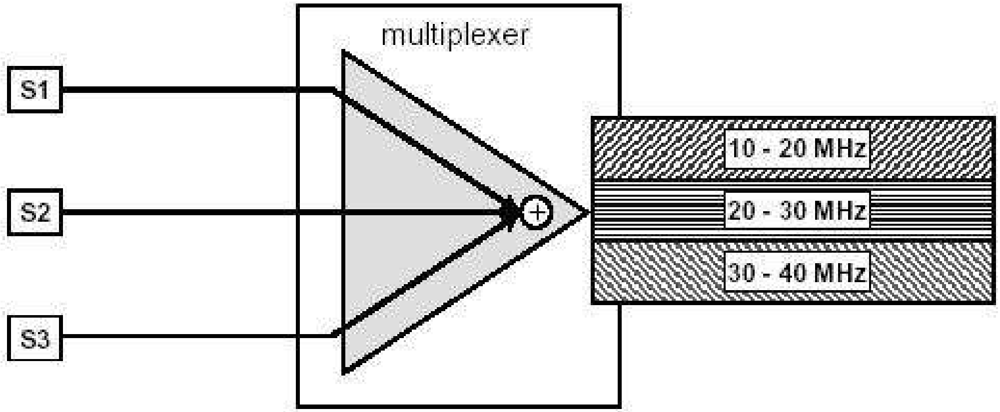
Multiplexing

- Multiplexing
  - mechanism for achieving **resource sharing**, i.e., sharing of **link bandwidth**
- Problem:
  - How can the link bandwidth be shared among n different senders
- 1st approach: partition the bandwidth **strictly** for all users
  - FDM and TDM



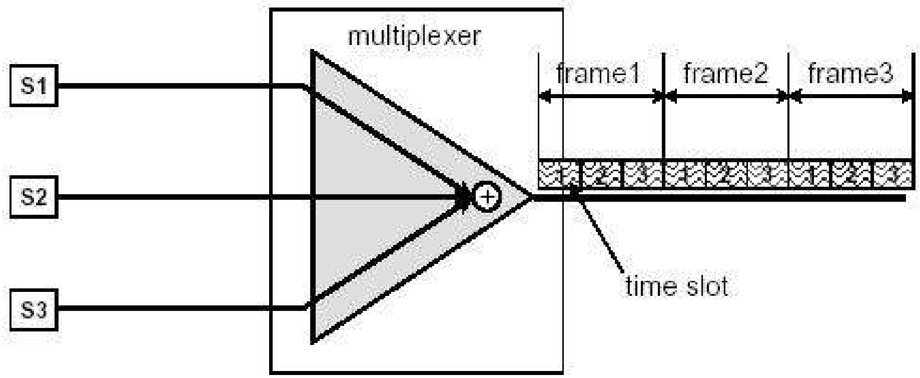
Frequency Division Multiplexing (FDM)

- FDM
  - oldest multiplexing technique
  - used e.g. in analogue circuit switched systems
  - fixed portion (frequency band) of the link bandwidth reserved for each channel
- FDM multiplexer is **lossless**
  - input: *n* 1-channel physical connections
  - output: 1 *n*-channel physical connection



Time Division Multiplexing (TDM)

- TDM
  - used in digital circuit switched systems and digital transmission systems
  - information conveyed on a link transferred in **frames** of fixed length
  - fixed portion (*time slot*) of each frame reserved for each channel
- TDM multiplexer is lossless
  - input: *n* 1-channel physical connections
  - output: 1 *n*-channel physical connection



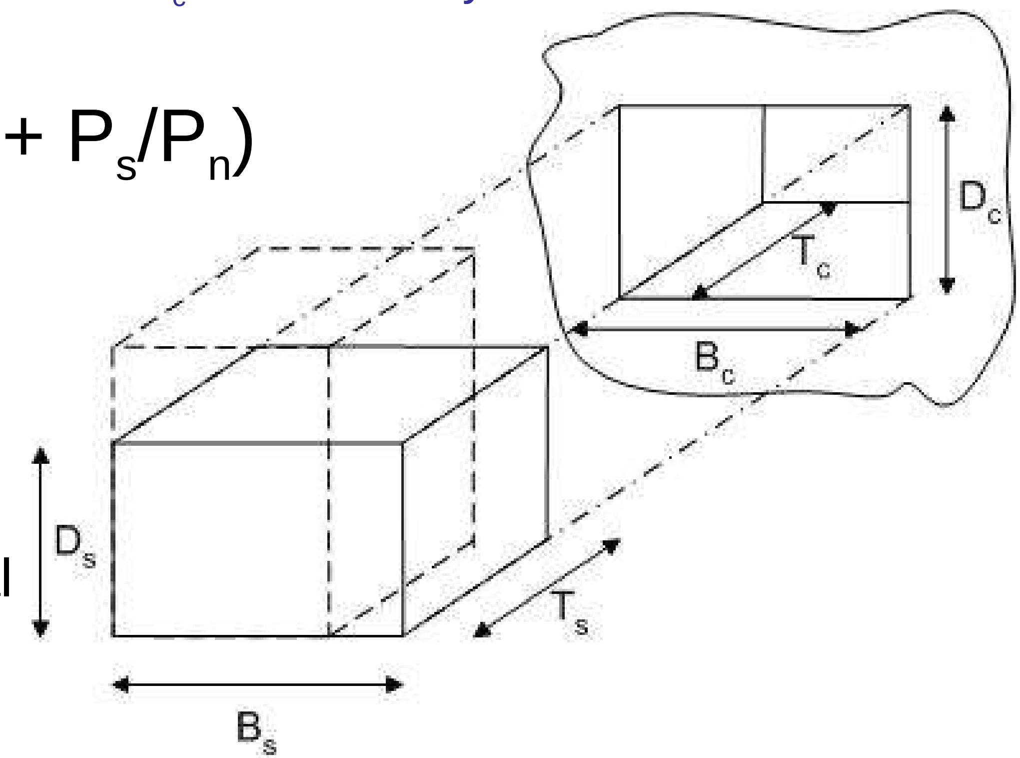
Multiplexing (Shannon)

- $I = C.T_c = B_c.T_c.\log_2(1 + P_s/P_n) =$

  $I = B_c.T_c.D_c$ [bit]

D_c … channel dynamic

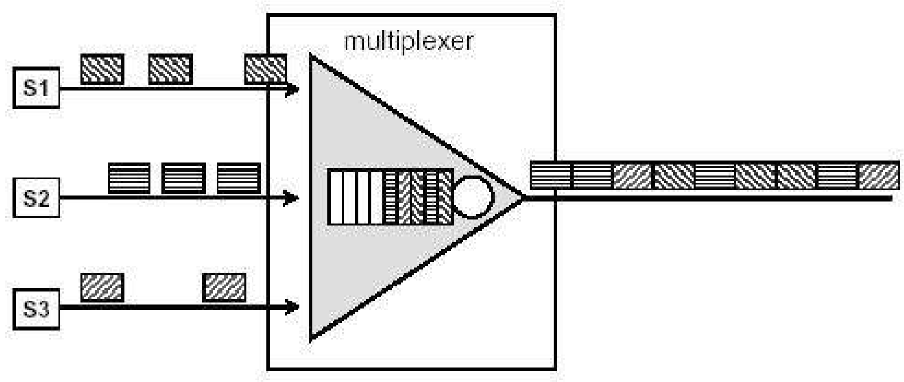
- with $C = B_c.\log_2(1 + P_s/P_n)$



B_s … signal bandwidth
T_s … signal duration
D_s … dynamic range of signal
B_c … channel bandwidth
T_c … transmission duration
D_c … dynamic range of signal

? Statistical multiplexing

- FDM and TDM are inefficient
  - If a sender has no data to transmit, the bandwidth allocated to the sender can not be used by others => **statistical multiplexing**
- In statistical multiplexing
  - basic transmission *unit is called a packet*
  - physical link is shared over time (cf. TDM) but **on-demand** (per each packet)
  - simultaneous packet arrivals are buffered (contention)
- as a result, packets from multiple senders are **interleaved** at the output
  - *buffer space is finite*, thus buffer overflow is possible (***congestion***)



Statistical multiplexing

- Statistical multiplexer is (typically) **lossy**
  - input: *n* physical connections with link speeds R_i ($i = 1,\ldots,n$)
  - output: 1 physical connection with link speed $C \leq R_1 + \ldots + R_n$
- However, the *loss probability* can be *decreased by enlarging the buffer*
  - with an "infinite" buffer enough that *C* exceeds **average aggregated** input rate
  - possible to dimension the size of the buffer such that *a given loss probability is achieved* (under some assumptions regarding the traffic)
- Statistical multiplexer and **QoS** (Quality of Service)
  - determining which packet to transmit from the buffer is called **scheduling**
- FIFO: packets are served in the arrival order
- Round robin: each connection (class) has own queue and they are served cyclically according to some weights
- Manymoreexist…
  - by using different scheduling mechanisms, some connections can be given "preferential" treatment (e.g., weighted round-robin) => **QoS** enabled networks

Outline

- Achieving connectivity
- Methods for resource sharing
- Enabling application level communication
- Performance issues
- Network architecture

Communication needs of applications

- Applications (processes) running on hosts need to communicate
  - different applications have different needs
- Typical *application requirements/considerations*
  - reliability?
  - packet sequence order?
  - security?
- Network *design challenge*
  - identify the *set of common services –* what are the application *needs?*
  - hide the complexity of the network without imposing too many constraints on the applications
- Network provides “logical channels”
  - IPC = Inter Process Communication
  - fills in the “logical gap”

Application requirement classification

1. Client/server applications (request/reply applications)
   – **client** process makes a *request* and the **server** process *replies*
   – strict requirements on packet loss (*no loss*), may have security requirements
   – Examples: file transfer (FTP), file systems (NFS), HTML documents on the web, digital libraries.
2. Streaming applications
   – sender generates a **continuous** stream of packets
   - the stream can correspond to, e.g., digitized audio or video

   – applications have *relatively tight requirements* on the **timeliness** of packet delivery, but they can tolerate packet loss to some degree:
   - *videoconferencing* has *tighter demands* than *video on-demand*
   - Security? *Conferencing* may require, e.g., encrypted transmission…

…, but what can go wrong?

Reliable transfer - what can go wrong?

- Error types
  - **Bit errors**: bit or burst of bits is corrupted
    - *Error correction detection* may be able to fix the problem
  - **Packet errors**: complete packet is lost
    - Due to *unrecoverable bit errors*, *congestion* (most likely reason), *software errors* (misplaced packets, relatively rare)
    - Problem: Not easy to distinguish between *packets that are excessively late* (due to e.g. severe overload) and *actually lost packets*.
  - **Node/link failures**:
    - A *physical link* is damaged/cut, *router crashes* ...
    - Can cause massive service disruptions
    - In Internet routing protocols can recover from link failures
    - Problem: Not easy to determine if a router is e.g. *completely down* or *just congested*.

Reliable transfer: one of the most important service properties
  - "network hides certain failures to make the network seem more reliable"

Outline

- Achieving connectivity
- Methods for resource sharing
- Enabling application level communication
- Performance issues
- Network architecture

? Performance measures (1): bandwidth

- Bandwidth = throughput
  - # bits that can be transmitted over the network in a given time
  - unit: bits per second (bps), e.g. 100 Mbps (cf. MB = megabytes = 8 Mb)
- Link bandwidth vs. end-to-end bandwidth
  - bandwidth of a physical link has a deterministic value, e.g. 155 Mbps (ATM)
  - link bandwidths are *constantly improving*: link bandwidths in the backbone
    - 1980's: 2 Mbps, 1990's: 155 Mbps, 2000: 1 Gbps
  - end-to-end (the received) bandwidth of an application depends on:
    - other traffic in the network (*congestion*)
    - application limitations (*CPU speed of the computer*)
    - protocol overhead (each bit sent by the application is "wrapped" in possibly several "envelopes" until the bit is transmitted on a physical link)

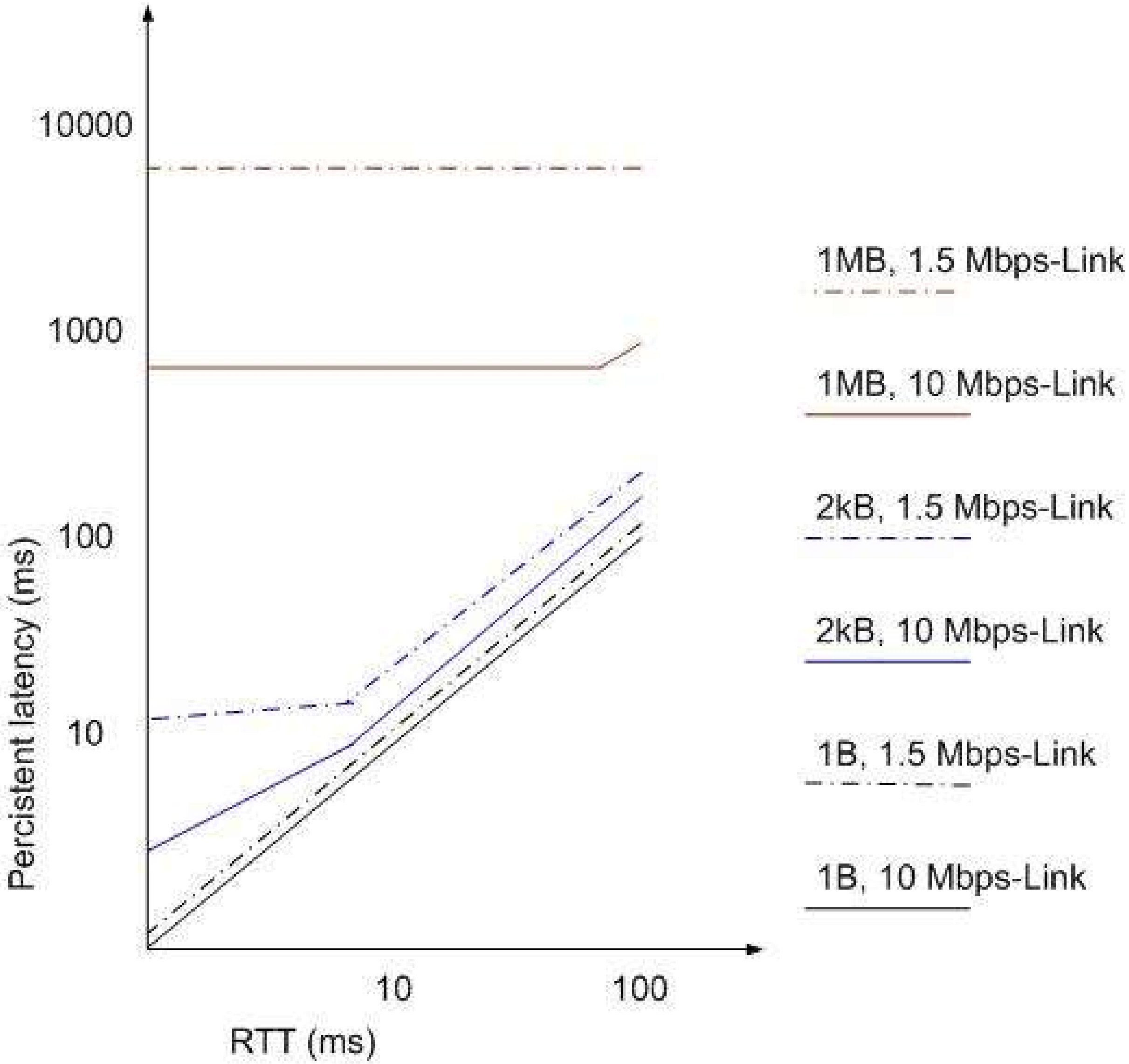
? Performance measures (2): latency

- Latency = delay
  - How long it takes a message to travel from one end of the nw to another
  - Measured in units of time, e.g., *latency across US* continent 24 ms
  - RTT (round trip time): time it takes a message to reach its destination and come back to the sender
- Components: *propagation delay, transmission delay, queuing delay*

Latency = Propagation + Transmit + Queue
Propagation = Distance / Speed of Light
Transmit = Size / Bandwith

  - Speed of Light: 2.3 E8 m/s in cable, 2.0 E8 m/s in fiber
- Applications can be either *bandwidth-* or *latency bound*
  - Telnet sessions are *latency bound* but large FTP transfers are *BW bound*

? Performance measures (3): latency



- 1B (keystroke) latency is equal to RTT
- 2kB (e-mail) negligible difference at 1ms RTT but no difference at 100ms RTT
- 1MB (digital image) RTT makes no difference – link speed dominates performance

Performance measures (4): latency

- It is sometimes useful to think in terms of ***instruction per miles***
- Example:

  Consider a computer that is able to execute 200 E6 instructions / s

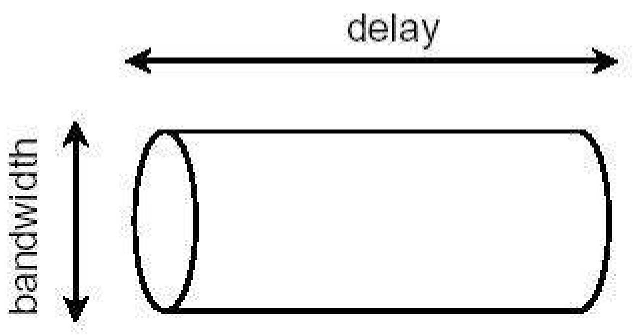
  Channel with 100ms RTT (over a distance of ~5000 miles)

  For 1 RTT the computer can execute 20 E6 instructions

  … or 4000 instructions per mile

Performance measures (5): latency

- The **product of RTT and bandwidth** determines
  - the amount of information transmitted by the user before any feed-back from the destination can be received
- In broadband wide-area-networks (WAN) this product can be very large
  - the sender can overload the receiver
  - if the sender does not "fill in the pipe", the network utilization may be low
- Example 1:
  - Assume that
    - distance is 1500 km
    - transmission rate C = 100 Mbps
  - The two-way propagation delay is
    - 2*1500/300,000 s = 0.01 s
  - Thus, the product of RTT and C is
    - 0.01*100,000,000 bits
    = 1,000,000 bits = 1 Mbit = 125 kB



? Delay x bandwidth product in high speed networks

Example 2:

- Assume RTT = 100 ms, we aim to transmit a file of size 1 MB
  - 1 Mbps network: time to transmit = 80 x RTT
    - 80 pipes full of data (stream of data to send)
    - clearly, the network design solution would be to increase the bandwidth
  - 1 Gbps network: time to transmit = 0.08 x RTT
    - only 8 % of the pipe is filled (the file has become a single "packet")
    - now, the latency dominates the network design
- Another way to think about this situation:

More data can be transmitted during each RTT on high speed networks (RTT becomes a significant amount of time)

101 RTTs vs. 100 RTTs -> relative difference of 1%

1RTT vs. 2RTTS -> relative difference of 100%

Delay x bandwidth product in high speed networks

- Thus, *dealing with the bandwidth* seems to be the main design issue in future high speed networks
- Applications have other performance requirements than *delay and bandwidth*
  - Applications may have an upper bound on required bandwidth:

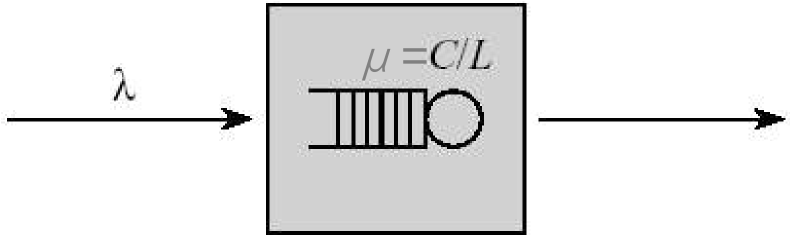
    Video application: 128kB frames 30 times per second

    -> requests a throughput or 32 Mbps
  - Real time applications have requirements on delay variation (jitter) caused by queuing in the network routers

    Compressed video appl.: average bandwidth requirement = 20Mbps
    Burst size depends on buffer size

? Performance of a statistical multiplexer (1)

- Internet is based on the use of statistical multiplexing
  - the output port of a router operates as a statistical multiplexer
- A statistical multiplexer can be modeled as a waiting system (= queue)
- Traffic consists of packets
  - each packet is transmitted with the full link speed C
  - packets arrive at a rate λ and let L denote the average packet length
  - packet service rate μ will be $\mu = C/L$
  - let $\rho = \lambda / \mu$, stability requirement: packet arrival rate $\lambda < \mu$ => $\rho < 1$



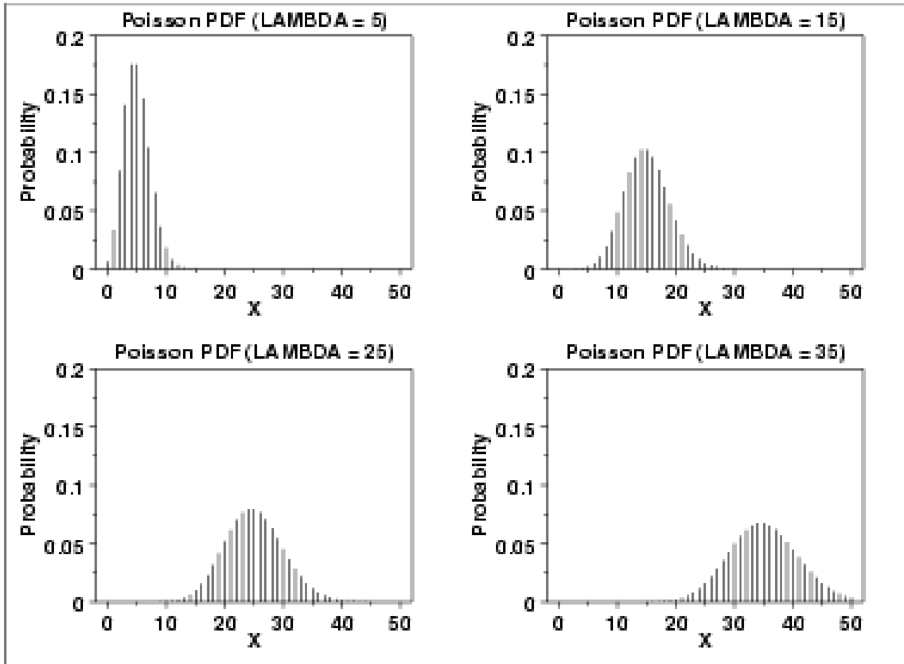
Performance of a statistical multiplexer (2)

- Poisson packet arrivals

  The Poisson distribution is used to model the number of events occurring within a given time interval.

  $P(x, \lambda) = (e^{-\lambda} \cdot \lambda^{x}) / x!$

  λ … is the shape parameter which indicates the average number of events in the given time interval.



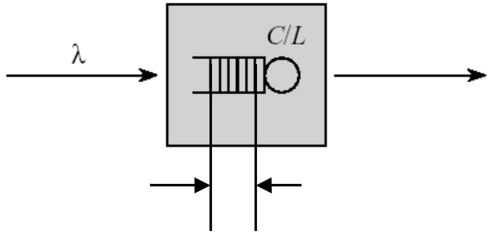
Performance of a statistical multiplexer (2)

- Assume Poisson packet arrivals with exponentially distributed sizes
  - M/M/1 queuing system

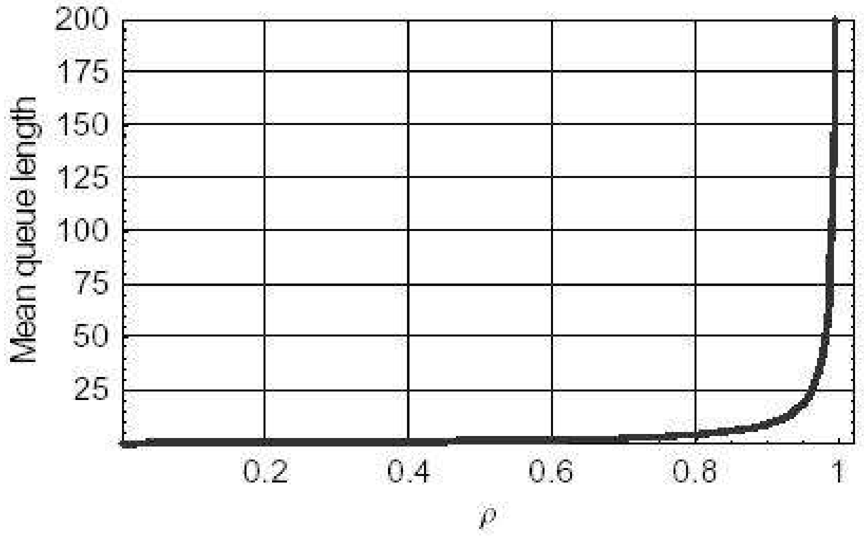
    M … negative exponential distribution

    G … general independent arrivals or service time

    D … deterministic arrivals or fixed-length service



- Load vs. mean queue length
  - mean queue length (and delay) rises sharply as load approaches 1



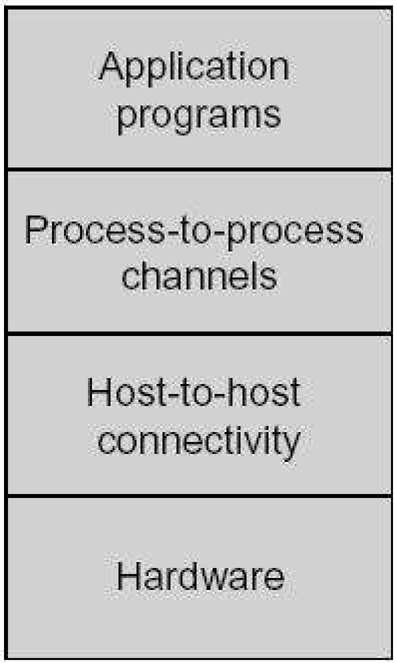
- Reasonable to design the network s.t. load < 0.9
  - link utilization always < 100%
  - **congestion control needed!!!**

Outline

- Achieving connectivity
- Methods for resource sharing
- Enabling application level communication
- Performance issues
- Network architecture

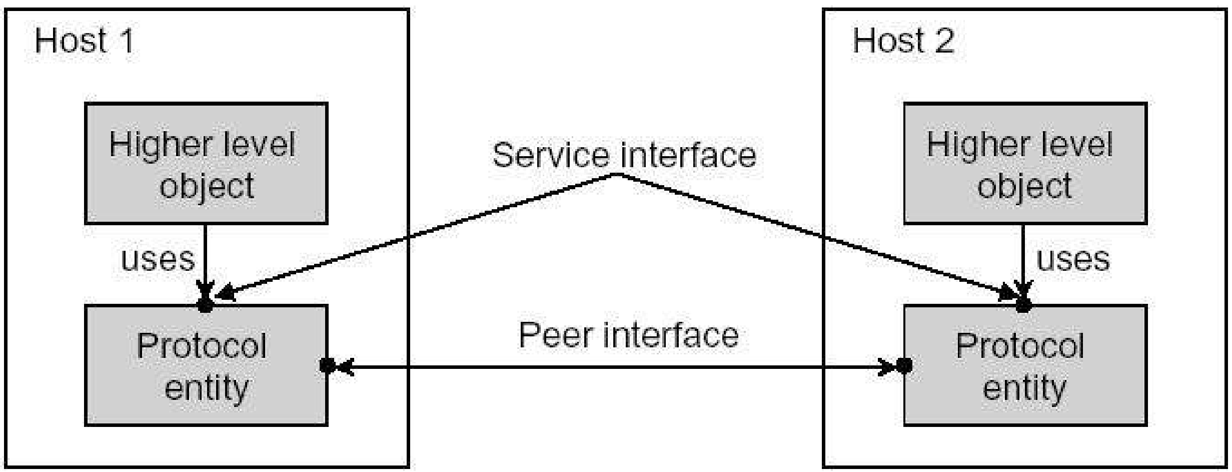
Layered architectures

- A computer network must provide for a large number of hosts
  - **cost effective, fair, robust and high performance connectivity**, and
  - it must be easily able to **accommodate new network technologies**
- *“well defined”* Network architecture
  - to **guide the design** and **implementation** of networks
  - abstractions used to ***hide complexities***
- *(not only)* in networks, abstractions lead to layered designs
  - services offered at ***higher layers are implemented*** in terms of services provided ***by lower layers***
  - often multiple abstractions (services) are provided to serve the varying requirements of above layers (**multiplexing of upper layer protocols**)
- Benefits of layering
  - decomposes the *implementation problem* into *manageable components*
  - *modular design* (adding *new functionality may only affect one layer*)



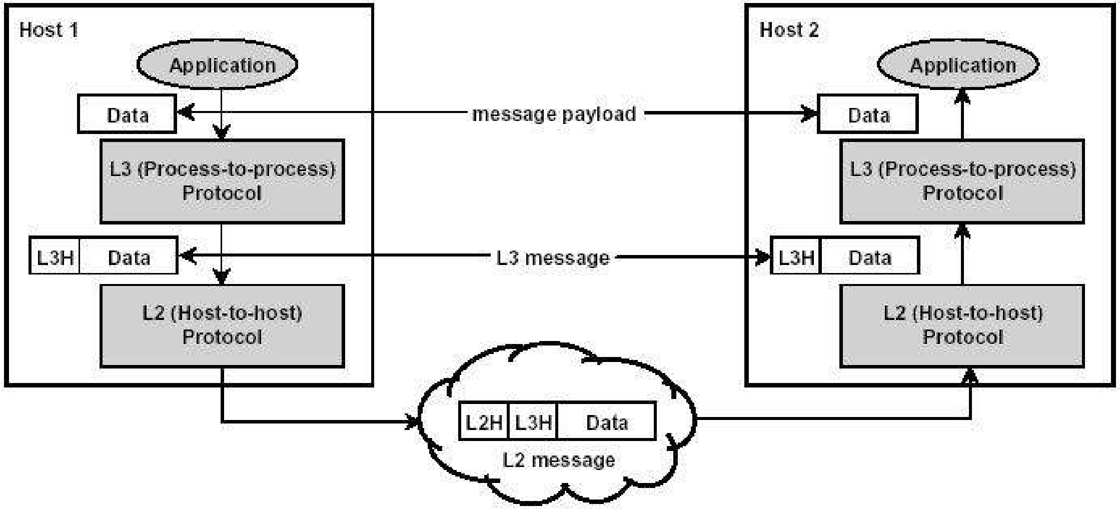
Protocols

- Each layer implemented by a **protocol**
  - protocols offer communication services to higher level objects
- A protocol offers two interfaces:
  - **Service interface**: offered to *higher level objects on the same host*
  - **Peer interface**: offered to *peer protocol objects* existing *on other hosts*



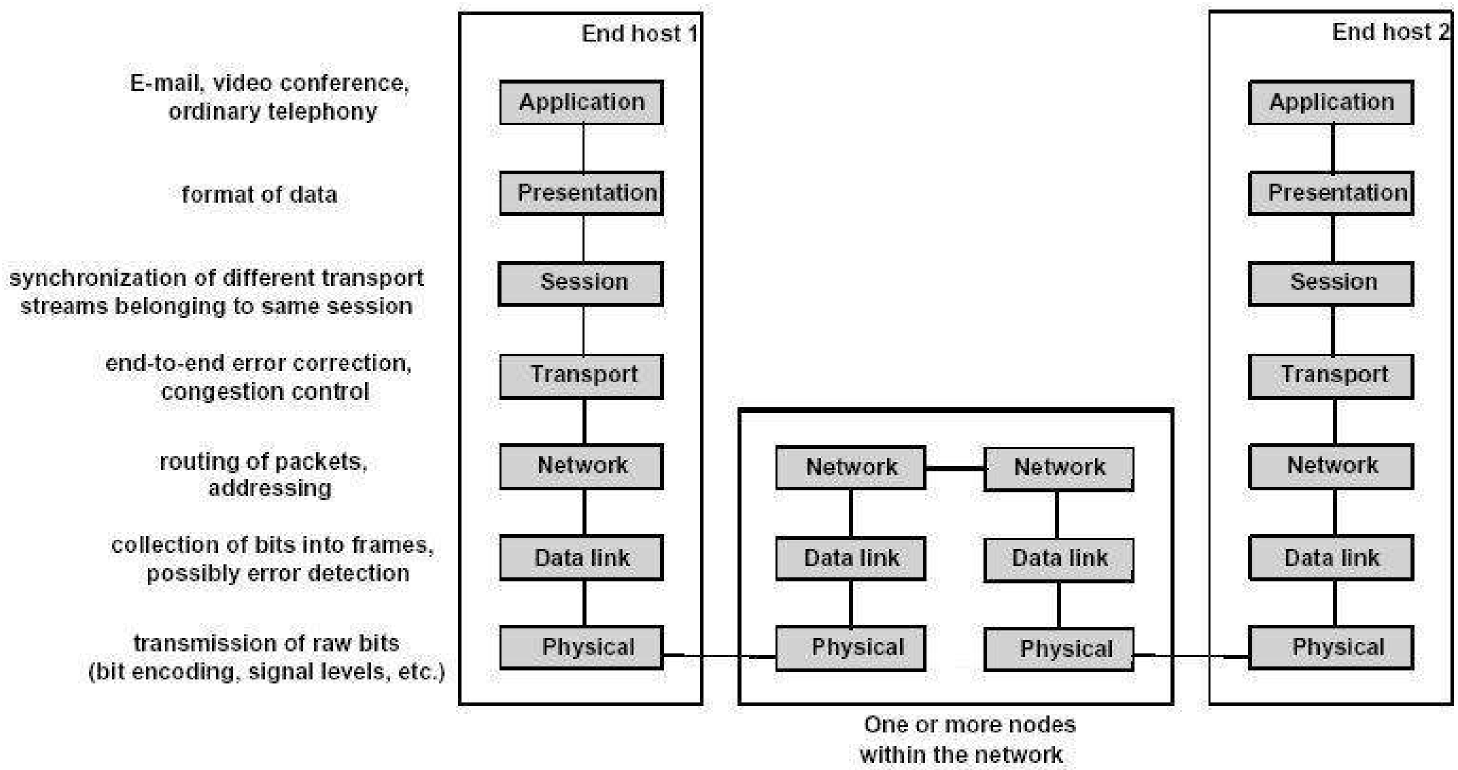
Encapsulation

- At the ***sender side***, each lower layer protocol *adds a header* (L3H, L2H) thus encapsulating the upper layer packet
  - *simple transformations* (compression, encryption) of the packets are possible
- At the ***receiver side***, each layer *removes* the corresponding *header* and forwards the packet to the higher layer protocol entity



OSI (Open Systems Interconnect) architecture

- The “classic” 7-layer reference model (late 70’s)
  - protocols following the model defined in conjunction with ISO and ITU-T



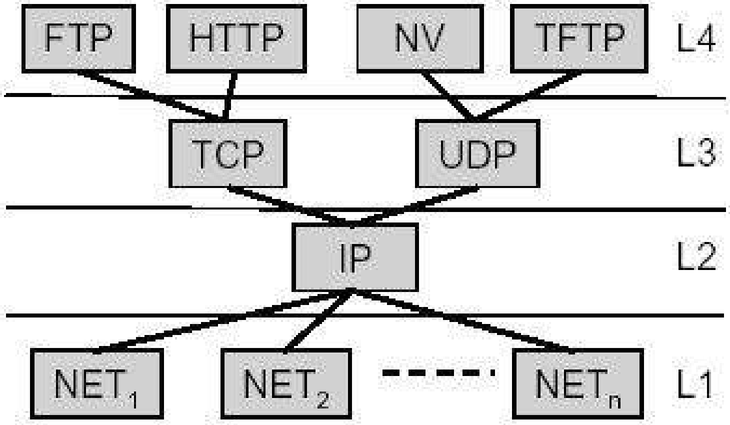
Internet architecture

- Internet architecture is sometimes called TCP/IP architecture (2 main protocols)
- Evolved out of experiences with early packet-switched ARPANET
- Internet & ARPANET were founded by “Advanced Research Project Agency” (one of R&D funding agencies of the DoD)
- … were around before OSI (major influence on OSI reference model)
- does not imply strict layering (*it is free to bypass*)
- Hour glass shape (*reflects the centralized philosophy*)
- Standards are adopted by IETF (*specification and implementations are required*)

Internet architecture

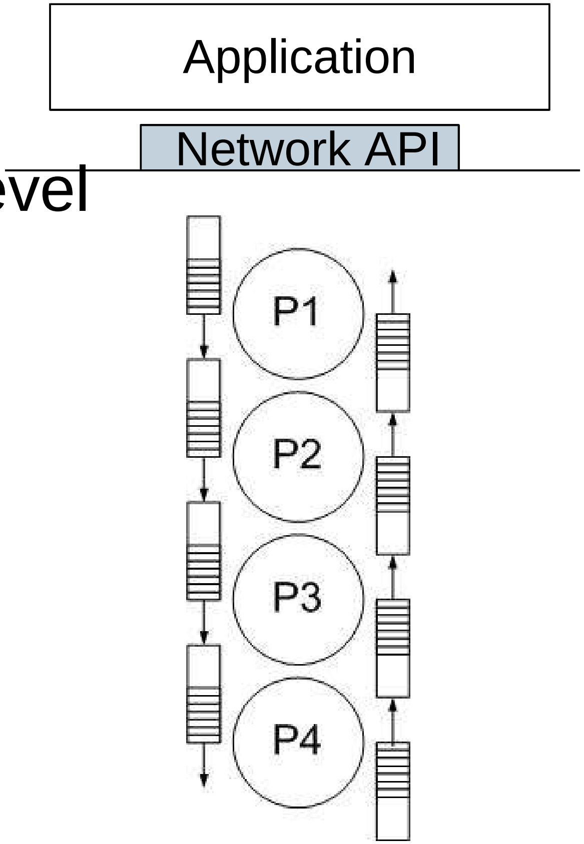
- Internet architecture has only 4 layers
  - L4: range of application protocols (**FTP**, …)
  - L3: **TCP** (*reliable byte transfer*) and **UDP** (*unreliable datagram delivery*) *provide logical channels* to applications
  - L2: **IP** protocol interconnects multiple networks into a single logical network
  - L1: wide variety of network protocols

"hour glass" shape



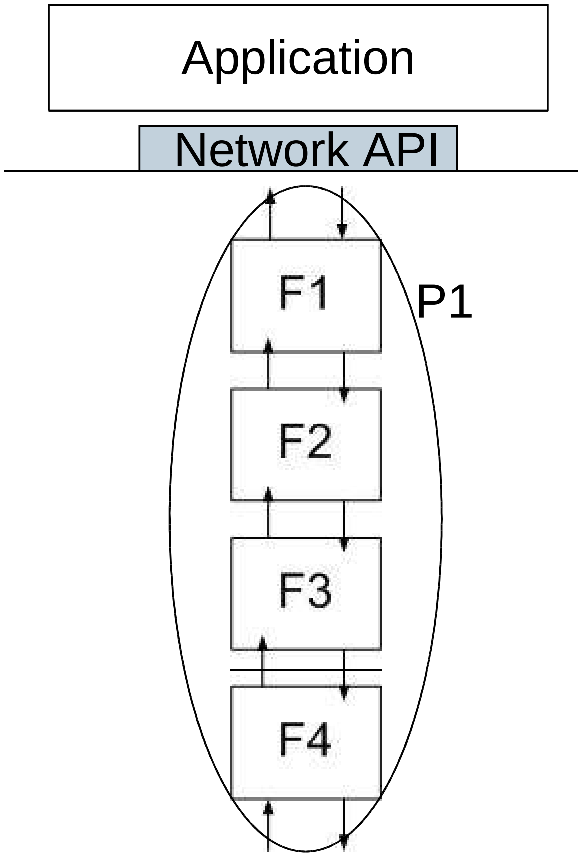
? Implementing network software

- protocol implementation issues
  - High level protocols interact with low level protocols (TCP-IP service interface)
- **Process model**: (*or "thread"*)
  - *separated address space*, CPU cycles (scheduler), … -> perfect model to describe concurrency
  - *process-per-protocol*
    - OS specific *inter-process communication*
    - … typically there are simple mechanisms to *queue messages with processes*
    - *Context switch* is required between each layer



Implementing network software

- **Process-per-message**
  - Each *protocol* is a *piece of code*
  - … a procedure is invoked
  - *Inbound-messages:* OS dispatches a process that is responsible for the message
  - *Outbound-messages:* application's process invokes a procedure call
  - *Network stack* is traversed in a sequence of procedure-calls.



protocol-to-protocol interface

Implementing network software

- process-per-protocol
  - …*easier* to think about
  - …, since protocols can be *implemented independently*
  - Service interfaces are required
- process-per-message
  - … generally more efficient
    - Procedure call is *more efficient* than a context switch
    - *Context switch at each level* vs. procedure call per level

Direct Link Networks

Chapter 2

(Part 1)

Outline

Hardware Building Block (Nodes, Links)
Encoding
Framing (SONET)
Error Detection
Reliable Transmission
Ethernet (802.3) vs. DIX Ethernet
Token Ring (802.5)
Wireless (802.11)
Network Adapter

Nodes

- Remember the recursive definition of the Internet (starting with a single physical link, nodes)
- This section gives an overview what “node” and “link” does mean – in doing so: … we will *define the underlying technology*.

- Nodes:=

1. **General purpose computer** (workstation-class machine)
   - Maybe a switch – forwarding messages (if2if)
   - Maybe a router – forwarding IP packets (nw2nw)
2. **Special-purpose HW**: usually done for:
   - *performance* and *cost reasons*

Nodes

- Limitation of nodes:
  - Finite memory (*packets must be buffered*)
  - Link bandwidth of the network adapter
  - Network adaptor: system’s I/O bus, device driver manages this adaptor
  - Computers are running at memory speed:
    - memory latency improvement (7% per year)
    - Processor speed is doubling every 18 month
  - <u>Network software has to be carefully about this</u>

Links

- Links are implemented on different physical media
- Anyway, the signals are electromagnetic waves, traveling at the speed of light (medium dependent)
- … provide the foundation for **transmitting all sorts of information** – *encoded signals*
- … divided into **two layers**:
  - Modulation (varying *freq., ampl., phase*, …)
  - Multiplexing (how many links per connection- half duplex vs. full duplex)

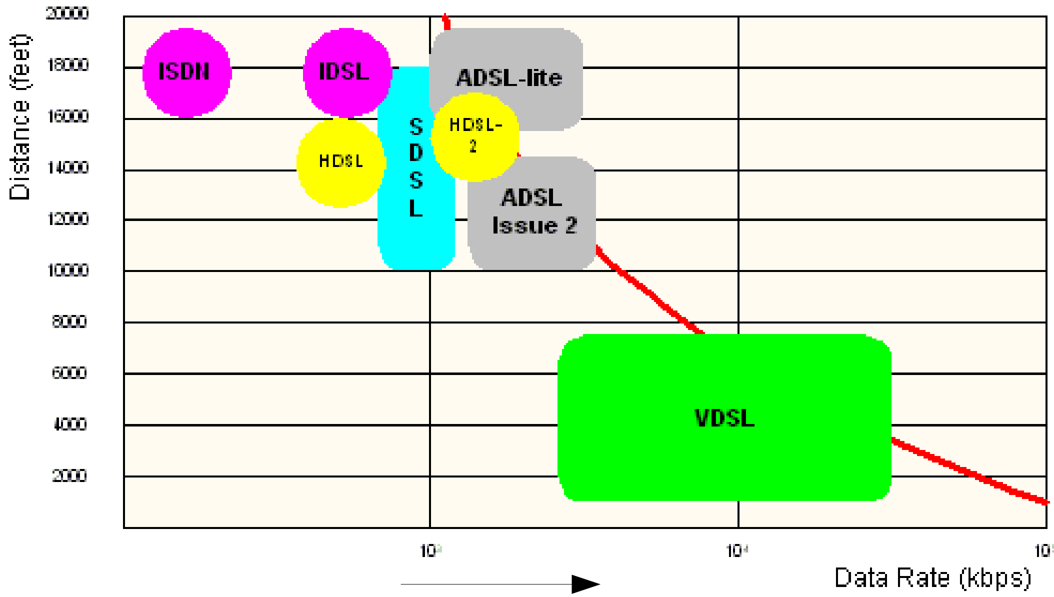
Links

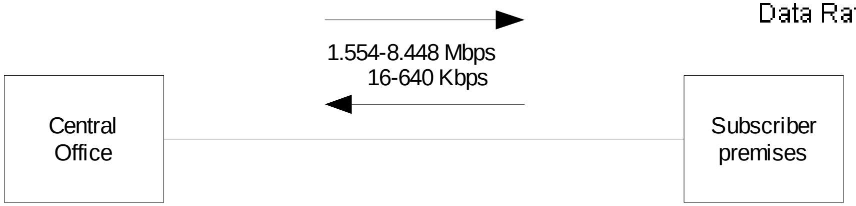
- **Cables**
  - CAT-5 (*twisted pair*) within-building norm, capable to run Gigabit Ethernet
  - Fiber is used to connect buildings
- **Leased Lines**
  - Leased dedicated link of the telephone company
  - STS-N (*synchronous transfer signal*)

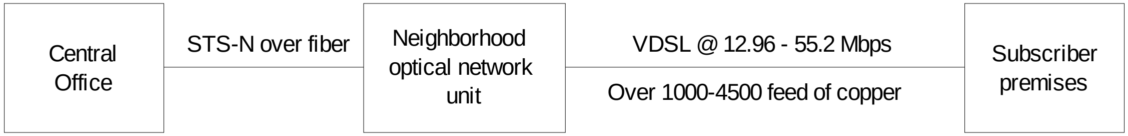
Service	Bandwith		
DS1 (T1)	1.544 Mbps	24 x 64 kbps	*Copper based*
DS3 (T3)	44.736 Mbps	30 x DS1	*Copper based*
STS-1 (OC-1)	51.840 Mbps	Basis STS *linkspeed*	*Optical fiber*
STS-3 (OC-3)	155.250 Mbps	3 x STS-1	*Optical fiber*
STS-12 (OC-12)	622.080 Mbps	12 x STS-1	*Optical fiber*
STS-24 (OC-24)	1.244160 Mbps	24 x STS-1	*Optical fiber*
STS-48 (OC-48)	2.488320 Mbps	48 x STS-1	*Optical fiber*

Links

- Last Mile
  - subscriber loops
    - 56 kbps POTS
    - 128 kbps ISDN (2 x 64kbps, 30 x 64kbps)
    - 2 Mbps ADSL,
    - 52 Mbps VDSL,
    - 40 Mbps, …







Links

- Shannons Theorem meets your modem
  - Voice-grade phone 300-3300Hz
  - $C = B.\log_2(1 + S/N)$ $[C] = Hz$
  - $B = 3300 – 300 = 3000$ Hz
  - S…signal power; N…noise power
  - $dB = 10.\log_{10}(S/N)$ $(S/N) = 1000$
  - $3000.\log_2(1001) \sim$ 30Kbps (28.8Kbps)
  - but it is possible to buy 56Kbps modems ????
    - improved line quality (S/N > 30dB)
    - elimination of analog lines (TAPs, …)

Links 01.03

- Wireless Links
  - PCS (personal communication services - US   1900MHz)
  - GMS (global mobile service – rest of the world 900/1800 MHz)
  - Medium- and low-orbit satellite constellations (ICE, Globalstar, Iridium, Teledesic)
  - IR (FIR) 850-950nm, 1Mbps, up to 10m
  - Radio
    - 5.2Gbps/17Gbps HIPERLAN (high performance European radio LAN) HiperLAN/1 and HiperLAN/2 (developed by ETSI BRAN)
      - HiperLAN/1 up to 20 Mbps @ 5-GHz
      - HiperLAN/2 up to 54 Mbps - is compatible with 3G-WLAN systems (worldwide in conjunction with similar systems in the 5-GHz RF band)
    - IEEE 802.11 (2.4GHz / 5GHz)
      - 802.11(PSK),
      - 802.11a (wireless ATM @ 5 or 6GHz - orthogonal frequency-division multiplexing (OFDM)), 54Mbps
      - 802.11b Wi-Fi (*complementary code keying* CCK @ 11Mbps) , and
      - 802.11g (up to 54Mbps @ 2.4 GHz)
    - Bluetooth 2.54GHz @ 1Mbps (up to 10m) *Piconets*

http://www.surveyor.in-berlin.de/perls/cshg-suchen.cgi?suchen=IEEE

Links 01.03

- Wireless Links
  - WLAN WG LETTER BALLOTS
    - IEEE Std
      - 802.11, published, WLAN Standard
      - 802.11a, published, 54Mbps@5GHz
      - 802.11b, published, 11Mbps@2,4GHz
      - 802.11b-Cor1, published corrective actions on 802.11b
      - 802.11d, published, 2,4 GHz ISM Band
      - 802.11e, planned, QoS (for VoIP)
      - 802.11f, published, access points handover
      - 802.11g, published, 54Mbps@2,4 GHz
      - 802.11h, published, European 802.11a
      - 802.11i, planned, secure WEP extensions
      - 802.11j, planned, Japanese 5GHZ band
      - 802.11k, planned, local (based) signal control
      - 802.11m, planned, some corrections to 802.11
      - 802.11n planned, 108Mbps-320Mbps

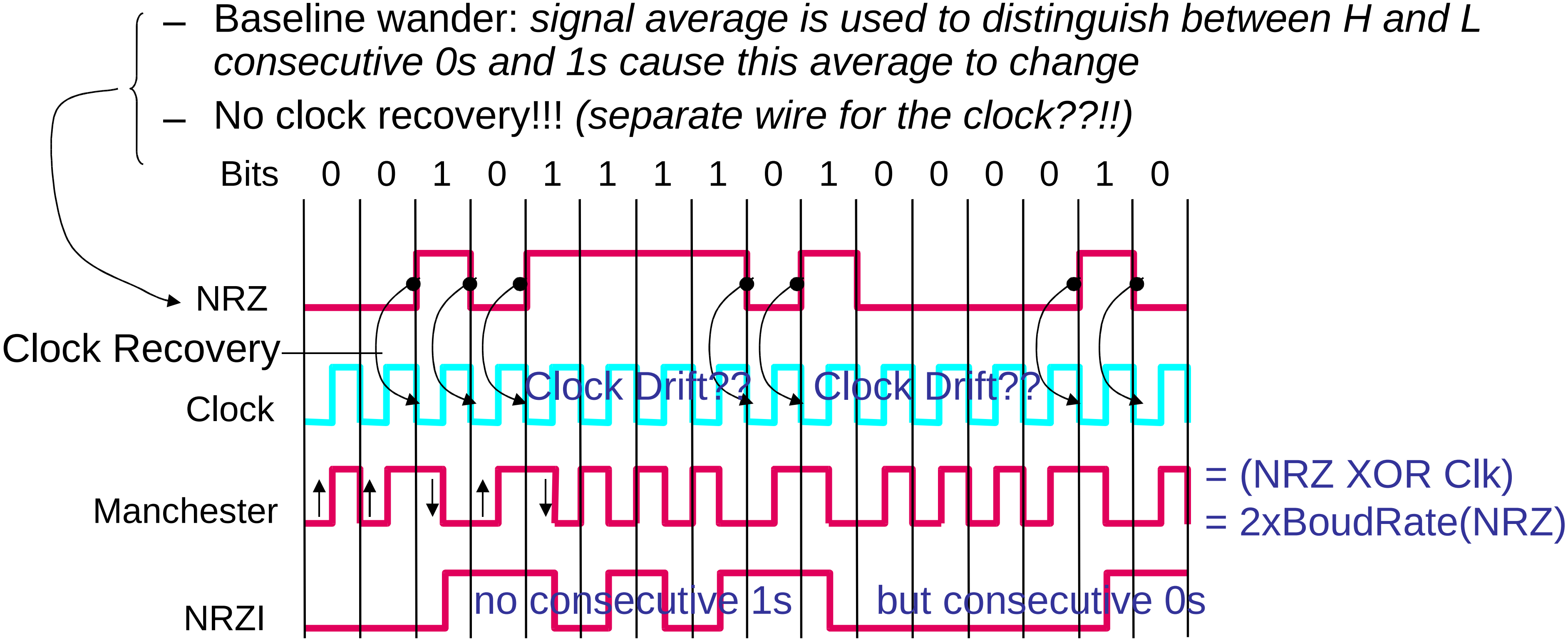
http://www.surveyor.in-berlin.de/perls/cshg-suchen.cgi?suchen=IEEE

Encoding

We ignore the details of modulation

Encoding requirements

- Baseline wander: *signal average is used to distinguish between H and L consecutive 0s and 1s cause this average to change*
- No clock recovery!!! *(separate wire for the clock??!!)*



transition if “1” stay if “0”

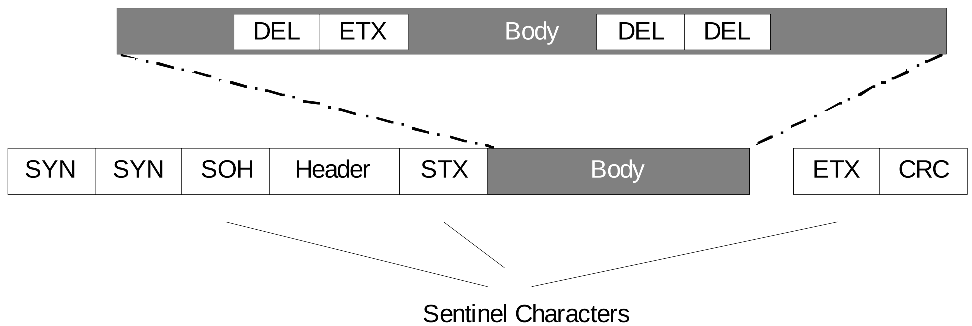
Encoding

- 4B/5B - a simple solution for the inefficiency of "Manchester Coding"
  - Insertion of extra bits
  - 4 bit are coded into 5 bits
    - Code has no more than one leading 0
    - No more than two trailing 0s
      - No code-pair has more than 3 consecutive 0s
    - 80% efficiency
    - 11111 is used when the line is idle
    - 00000 is used when the line is dead
    - FDDI is using this scheme (Fiber optics Distributed/Digital Data Interface)

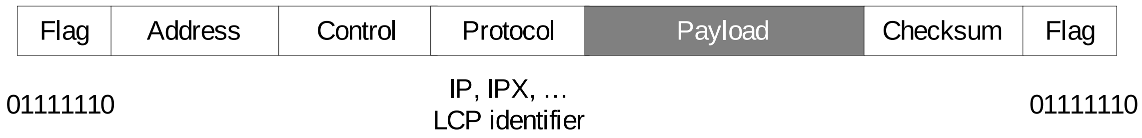
Framing

- Frames: Blocks of data (no bit streams any more)
- Byte oriented Protocols (BISYNC, PPP, ...)
  - Sentinel Approach (*guard*)
    - BISYNC (binary synchronous communication)

DLE data link escape
SYN syncronise
SOH start of header
STX start of text
ETX end of text



    - PPP



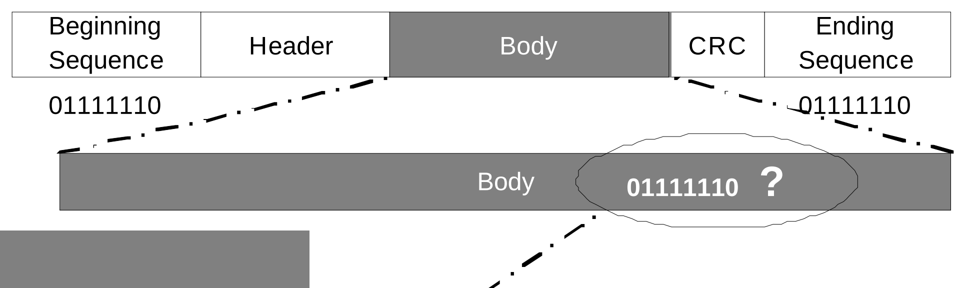
Framing

– Byte Counting Approach (DDCMP)

SYN	SYN	Class	Count	Header	Body	CRC

- Problem of transmission errors -> *framing error (wait until next SYN to start collecting data again)*

- Bit oriented Protocols (HDLC)



Bit Stuffing:
011111 01 ↙ 0111111
011111 00 ↙ 0111110
0111111 ?
01111110 ↙ Ending Sequence
01111111 ↙ Error

- LAP-B *(ISO 4335)*
- LAP-D *(ITU Q.921) Link-Protokoll für ISDN-Signalisierung*
- LLC *(Logical Link Control, IEEE 802.2) für LANs*

?

Framing:
SONET/SDH

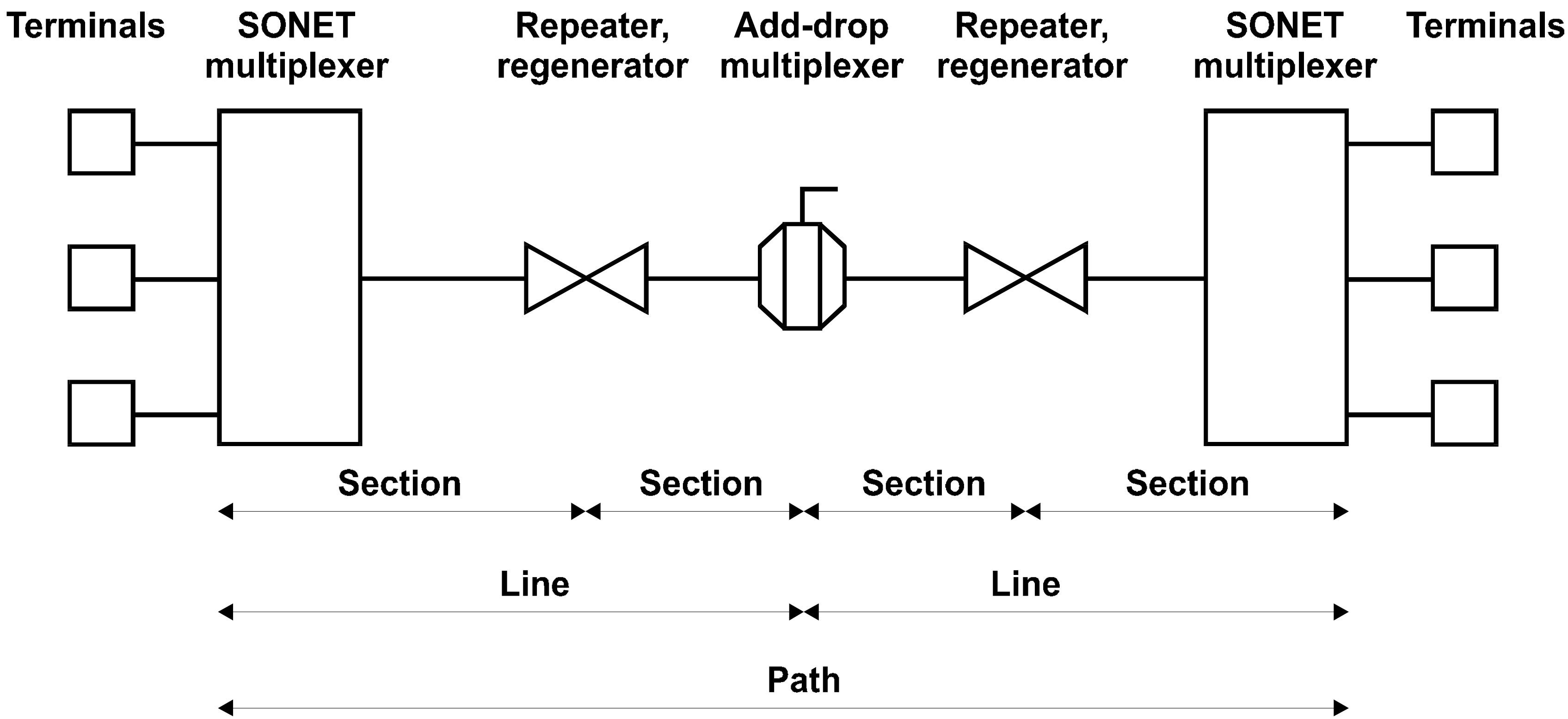
- "Clock based framing" (SONET/SDH)
  - Proposed by *"Bell Communications Research"* (**Bellcore**)
  - … then developed under ANSI of optical fibers
  - … adopted by the ITU-T *(SDH is an ITU standard)*
    - *Synchronous optical network* (USA)
    - *Synchronous digital hierarchy* (CCITT, ITU-T)
  - SONET := set of standards and concepts that provide:
    - Standard interfaces (compatibility between different suppliers equipment)
    - Appropriate payload access
    - Sufficient overhead capacity

Framing:
SONET/SDH

- Sources are synchronized by a master clock
- Standard Multiplexing Format with **51.84 Mbit/s** as a Basis-*frame*
- SONET **addresses framing-** and **encoding problems**
- Multiplexing low speed links on a single high speed link
- Integration of Organization, administration & maintenance (**OAM**) capabilities

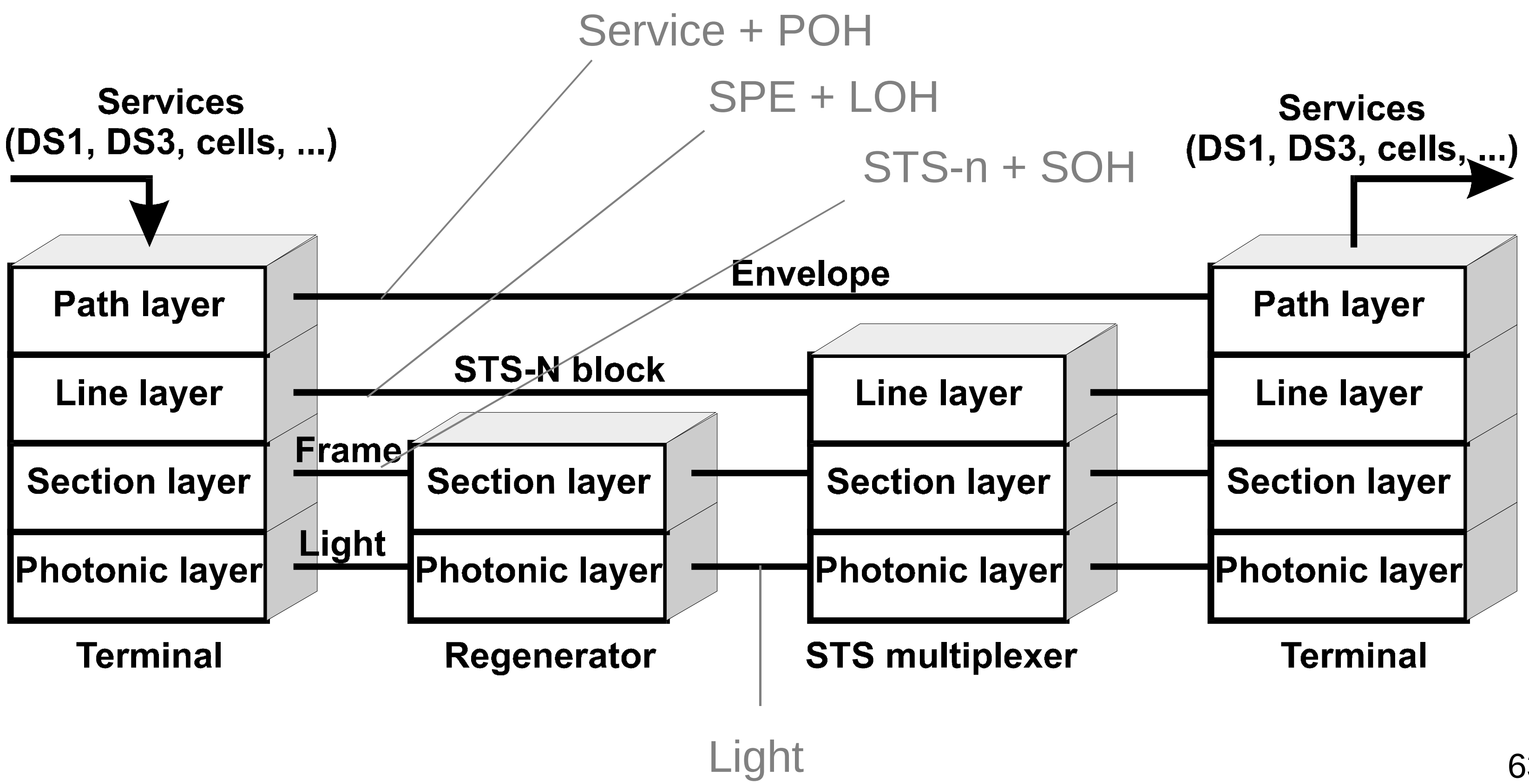
SONET Tutorial
http://www.iaik.tu-graz.ac.at/teaching/03_rechnernetze/download/index.php
Username: rn2004
Passwd: 4711123

Framing: SONET/SDH physical hierarchies



Framing: SONET/SDH
Logical hierarchies I

- Model based on four layers



Framing: SONET/SDH Logical hierarchies II

- *Path layer*
  - End-to-end – data transfer (end-to-end error detection)
- *Line layer*
  - *Synchronisation, multiplexing on SONET-frames , with frame and frequency alignment*
- *Section layer*
  - Forming a basic SONET-*frame*
- *Photonic layer*
  - transforming electr. (STS-n) to optical (OC-n) signals
  - Fiber definition, power of the Laser, etc.

Framing: SONET/SDH Signalhierarchie

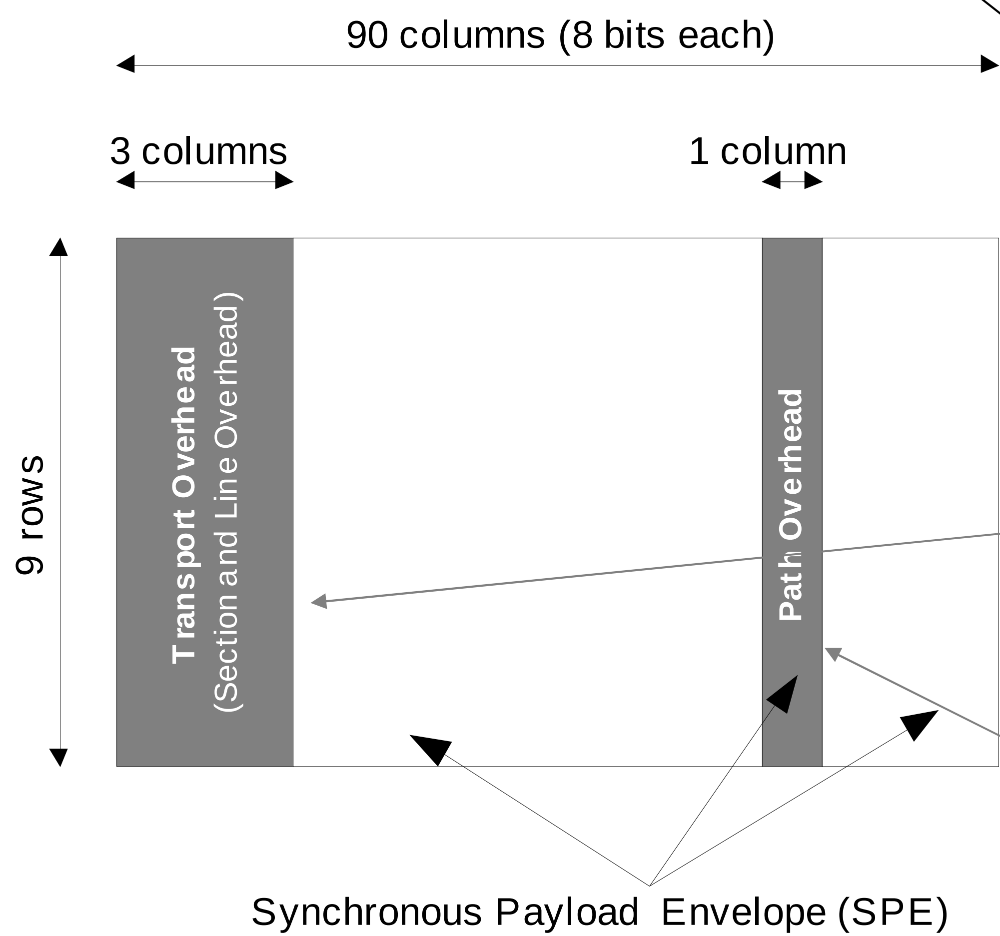
SONET	**SDH (ITU-T) name**	**Datenrate [Mbps]**	**Payload-Rate [Mbps]**
STS-1	*N/A*	**51.84**	50.112
STS-3	STM-1	155.52	150.336
STS-9	STM-3	466.56	451.008
STS-12	STM-4	622.02	601.334
STS-18	STM-6	933.12	902.016
STS-24	STM-8	1244.16	1202.688
STS-36	STM-12	1866.24	1804.032
STS-48	STM-16	2488.32	2405.376

Synchronous Transport Module Level x

Synchronous Transport Signal Level x

Framing: SONET STS-1 Frame

90 columns * 9 rows* 8 Bit / $125*10^{-6}$ = **51.84 Mb/s**



4000Hz voice signal

(8000 samples/s) = 64Kbps

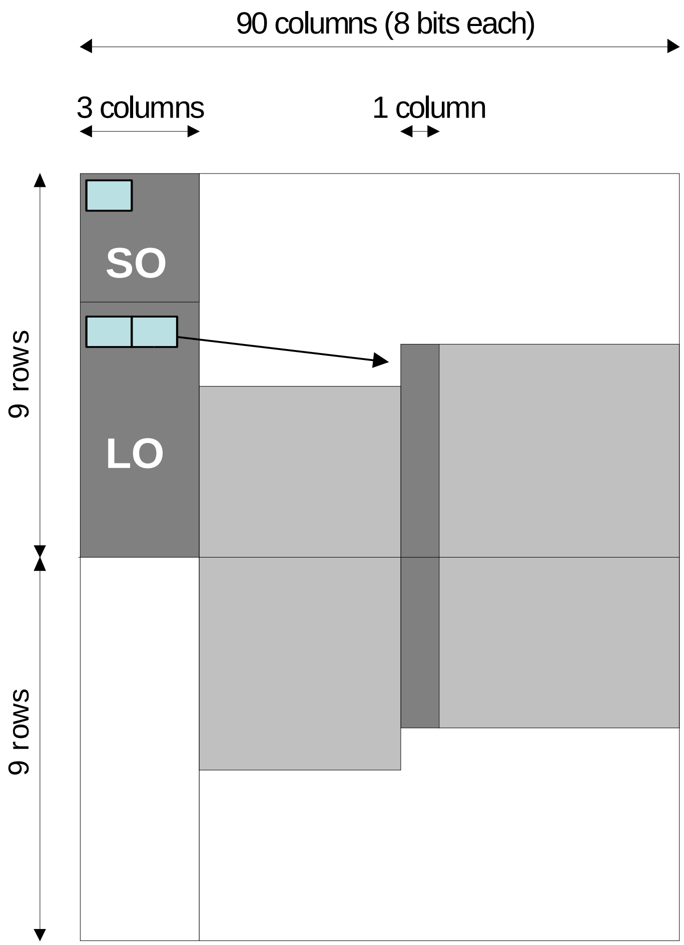
resulting timeframe = 0.125ms

(async. DS3 signal rate = 44.736Mbps)

- Multiplexing
- Switching
- Signal regeneration

- end-2-end error control

Framing: SONET STS-1 Frame



SPE does not need to be aligned to a single STS-1 frame …

… may occupy parts of two consecutive frames

Two bytes in in LOH are allocated are indicating the offset between the pointer and the first byte of the SPE

SPE is “floating”

Framing: Transport overhead

Section overhead (*unscrambled*)

- **Framing:** **begin of a STS-1 *frame***
- **STS-1 ID:** **STS-1 *channel ID (byte interleaving in STS-n signals)***
- **BIP-8:** ***8-bit parity (error monitoring) calculated over all bits of the previous STS-n***
- **Orderwire:** ***network maintenance staff***
- **User:** ***operator applications***
- **Data com:** ***maintenance, provisioning info.***

Line overhead

- **Pointer:** **begin of the STS SPE**
- **Pointer Action:** **frequency justification**
- **APS:** ***automatic protection switching***
- **Data com:** ***maintenance, provisioning info.***
- **Growth: for later usage**

Section overhead	Framing A1	Framing A2	STS-1 ID C1
	BIP-8 B1	Orderwire E1	User F1
	Data Com D1	Data Com D2	Data Com D3
Line overhead	Pointer H1	Pointer H2	Pointer Action H3
	BIP-8 B2	APS K1	APS K2
	Data Com D4	Data Com D5	Data Com D6
	Data Com D7	Data Com D8	Data Com D9
	Data Com D10	Data Com D11	Data Com D12
	Growth Z1	Growth Z2	Orderwire E2

Framing: Transport overhead
Example: OC-3c / STM-1

<u>Section overhead</u> (*unscrambled*)

A1, A2
: Frame alignment. These octets contain the value of 0xF628. The receiver searches for these values in the incoming bit stream.

C1
: STS-1 identification. Since OC-3c and STM-1 contain three STS-1 streams, the three C1 bytes contain 0x01, 0x02 and 0x03, respectively.

B1
: Section error monitoring. Contains BIP-8 of all bits in the previous frame using even parity, before scrambling.

Section overhead	Framing A1	Framing A2	STS-1 ID C1
	BIP-8 B1	Orderwire E1	User F1
	Data Com D1	Data Com D2	Data Com D3
Line overhead	Pointer H1	Pointer H2	Pointer Action H3
	BIP-8 B2	APS K1	APS K2
	Data Com D4	Data Com D5	Data Com D6
	Data Com D7	Data Com D8	Data Com D9
	Data Com D10	Data Com D11	Data Com D12
	Growth Z1	Growth Z2	Orderwire E2

Framing: Transport overhead Example: OC-3c / STM-1

<u>Line overhead</u> (*unscrambled*)

B2

> Line error monitoring. Contains BIP-24 calculated over all bits of the line overhead of the previous frame with even parity.

H1 (bits 1-4)

> New data flag (specifies when the pointer has changed), path AIS.

H1 and H2 (bits 7-16)

> Pointer value, path AIS. These bytes specify the offset between the pointer and the first payload byte.

H3

> Pointer action (used for frequency justification), path AIS.

Section overhead	Framing A1	Framing A2	STS-1 ID C1
	BIP-8 B1	Orderwire E1	User F1
	Data Com D1	Data Com D2	Data Com D3
Line overhead	Pointer H1	Pointer H2	Pointer Action H3
	BIP-8 B2	APS K1	APS K2
	Data Com D4	Data Com D5	Data Com D6
	Data Com D7	Data Com D8	Data Com D9
	Data Com D10	Data Com D11	Data Com D12
	Growth Z1	Growth Z2	Orderwire E2

Framing: Transport overhead Pointer Action

… even though all sources are synchronized by the same clock, deviations do occur:

When the rate of the source is higher than the local STS-1 rate H3 is used to add an extra byte to the SPE

When the source is slower, a byte is deleted from the SPE

Suppose that the payload speed is higher than the frame speed – an **extra byte is added** to he payload – this is done bay decrementing the pointer in frame$_{n+1}$ by 1

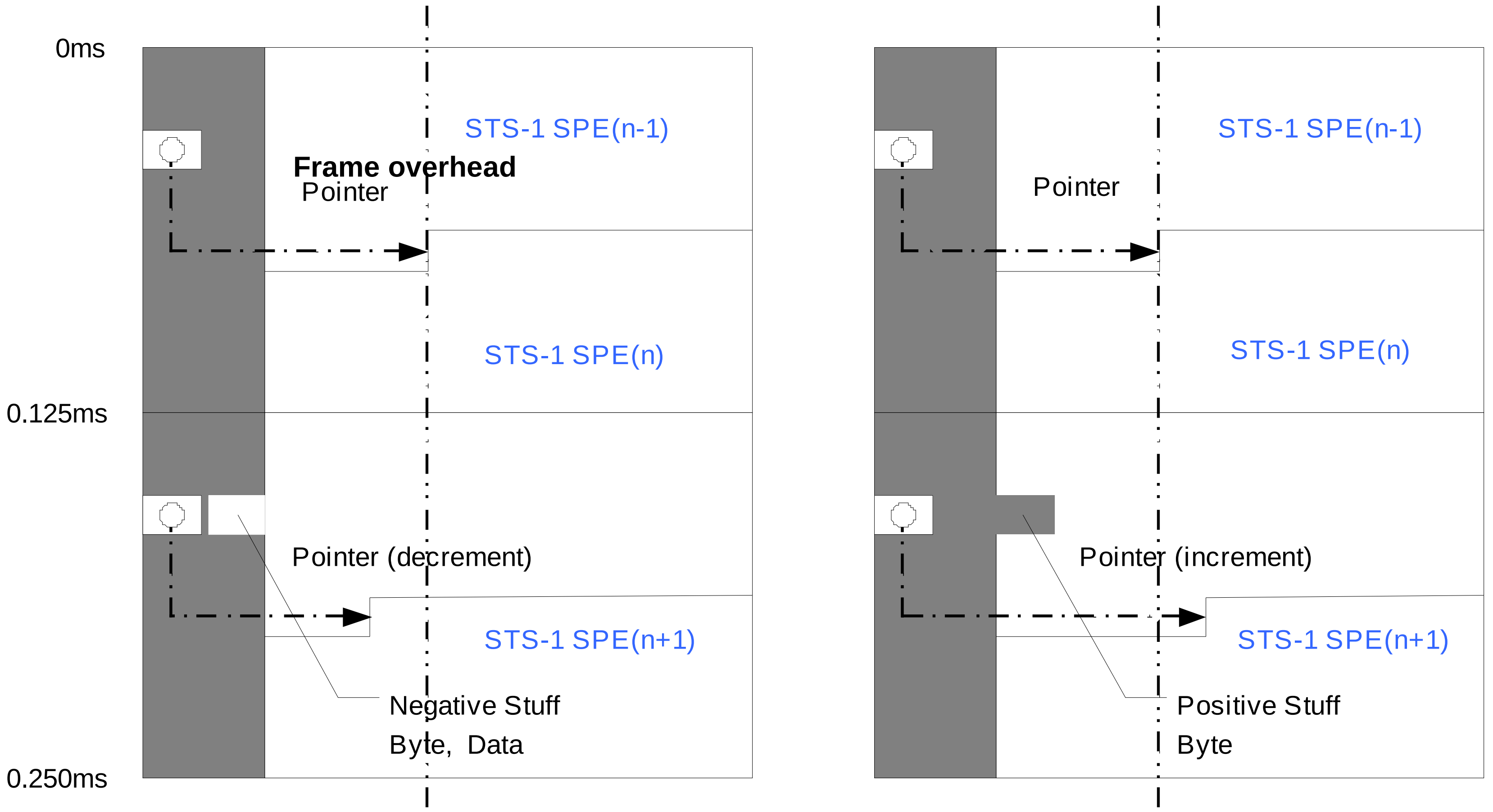
… so SPE$_{n+2}$ starts sooner – the extra byte is place in H3

Suppose that the payload bit rate is slower than the frame rate – then, **one byte is not made available** to the payload – this is done by incrementing the pointer in frame$_{n+1}$ by one

… so SPE$_{n+2}$ starts later – the byte next to H3 remains empty (“stuffed”)

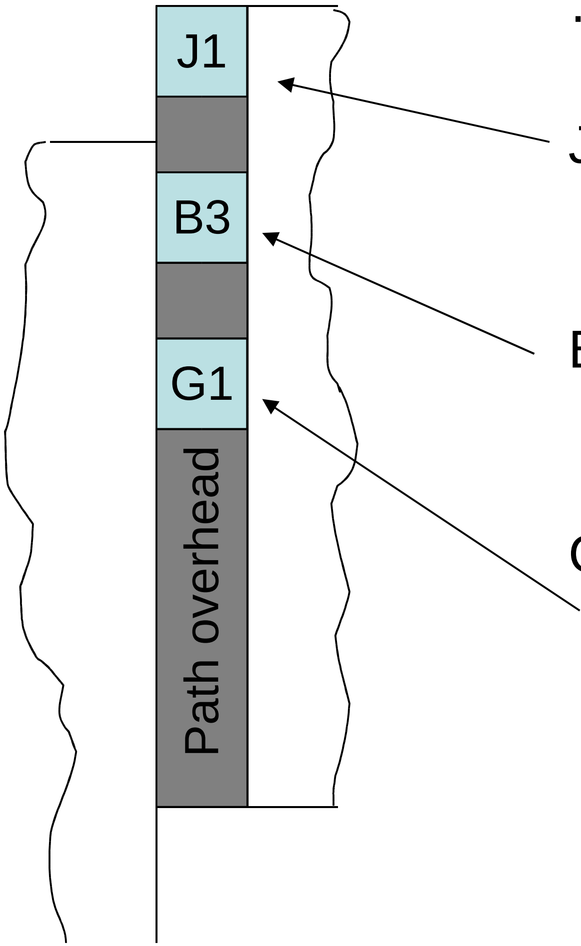
Framing: SONET Synchronization

- Clock-differences: *pointer action* field



Framing: SONET Synchronization

- Path Overhead



… is assigned and remains with the payload until the payload is de-multiplexed

J1 is used to repeat a (user programmable) 64-byte fixed length string – so path terminals can verify its continued connection

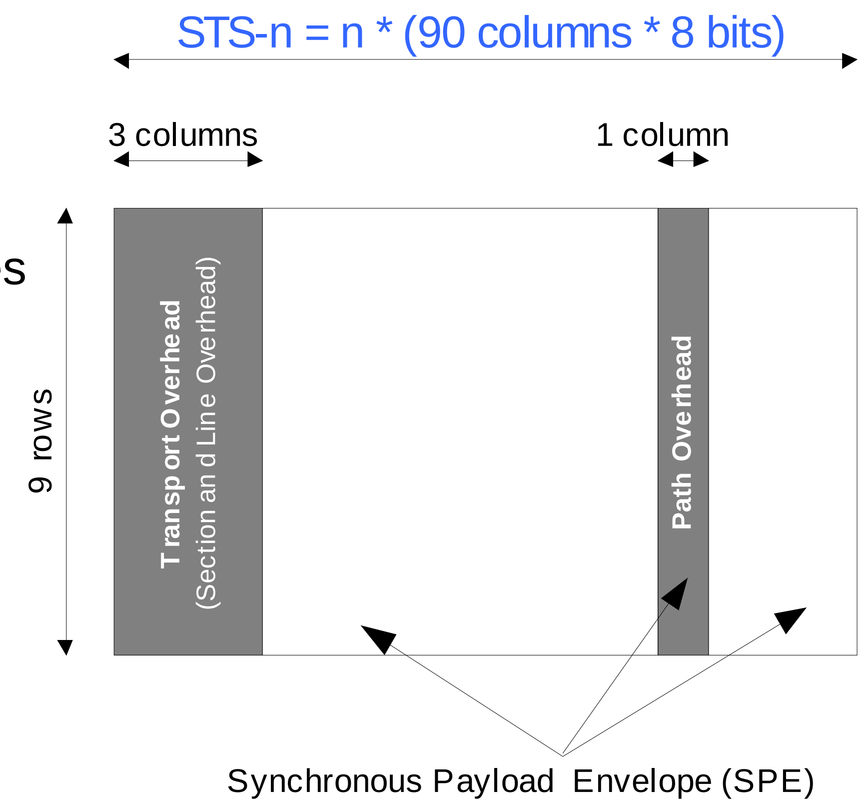
B3 Path error monitoring. Path BIP-8 over all bits of the payload of the previous frame, using even parity before scrambling.

G1 Path status (monitoring) Allows monitoring of complete full-duplex path at any point along a complex path.

Framing: High-rate multiplexing

- concatination of STS-1 *frames*

  => STS-n (OC-N) aus
  *n * STS-1* (*n * OC-1*)
- OC-N vs. OC-Nc
  - c … *concatenated*
  - Access to a single SPE at multiple service rates
  - Example: OC-3c, i.e. 155.52 Mbit/s at *concatenated frames* (9 + 261) * 9 Byte

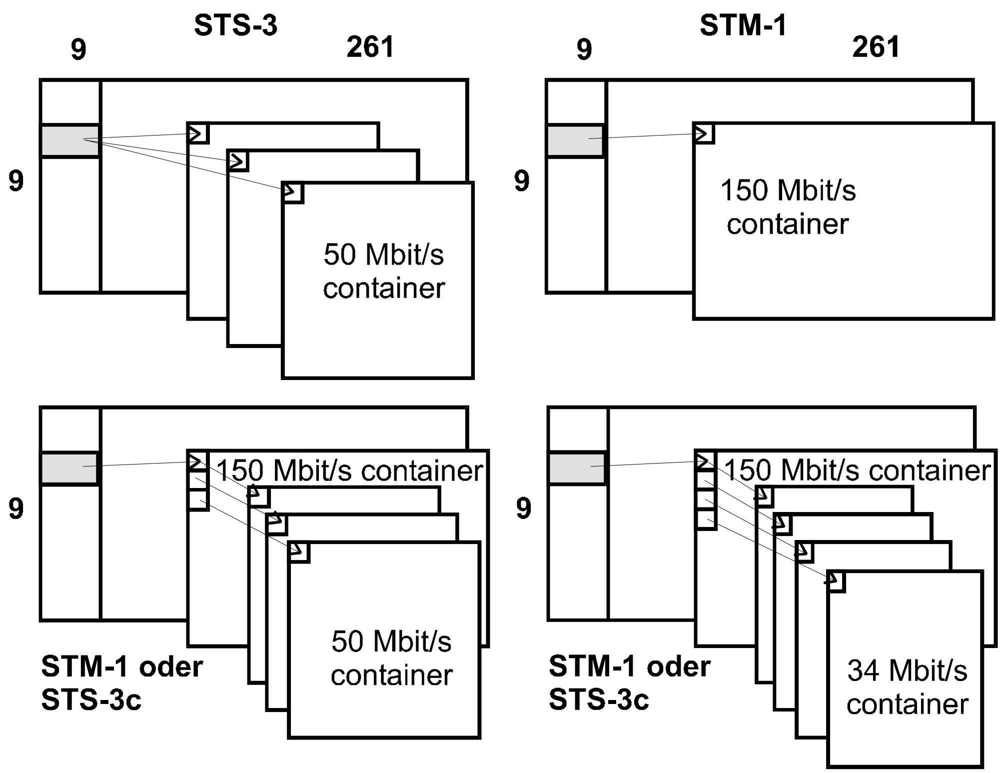


Framing: High-rate multiplexing

- STS-N
  - Formed by byte-interleaving N*STS-1 signals
  - 3*N columns of Transport Overhead
    - Frames aligned
    - Redundant fields are not used (APS, Datacomm)
  - <u>N distinct payloads (87*N bytes)</u>
    - Not frame aligned
    - **N columns of Path Overhead – all used**
- STS-Nc
  - 3*N columns of Transport Overhead
    - Frames aligned
    - Redundant fields are not used (APS, Datacomm)
  - <u>single payload</u>
    - **1 column of Path overhead**

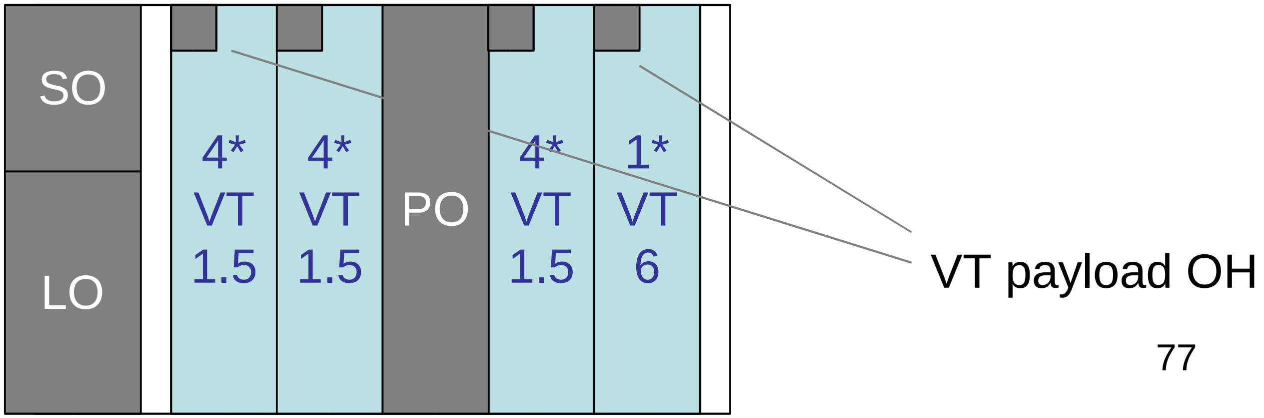
Framing: SONET container

- Access to different service rates within a single SPE as *administrative unit* (AU) or *nested signal*



Framing: SONET Virtual tributaries

- access to sub-DS-3 signals
- STS-1 SPE contains 7 VT-*groups* @ 6912 Mb/s
  - VT1.5 T1-Signal *(max. 4 per VT group)*
  - VT2 E1-Signal *(max. 3 per VT group)*
  - VT3 DS1C (3.152 Mb/s) *(max. 2 per VT group)*
  - VT6 DS2-Signal
- VT's (virtual tributaries) are based on 9-Byte columns – realized within a single STS-1 SPE (1 column *path overhead*)



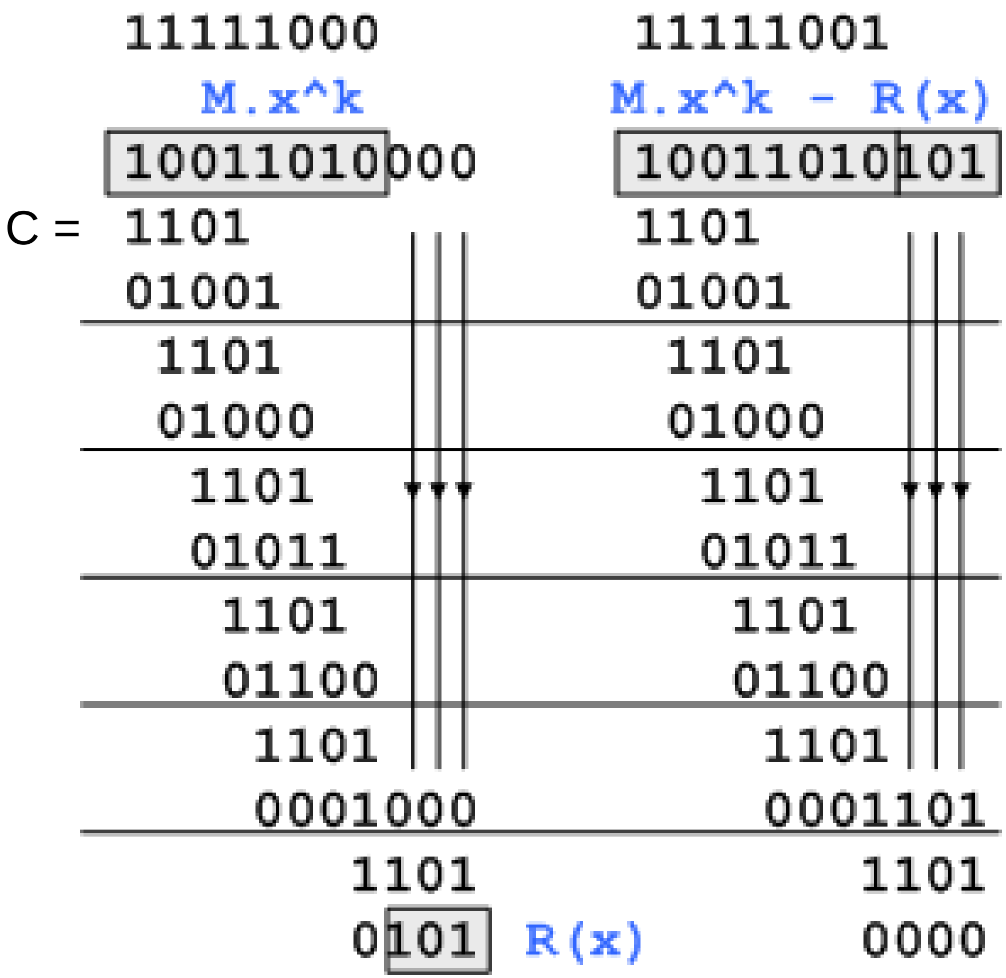
Framing: Why SONET ?

- standardized, flexible Multiplexing for different rates
- **simple *payload access***
  - *Add-drop multiplexer* – unwrapping of the whole frame is not required.
- OAM functionality on
  - *section layer (F1)*
  - *line layer (F2)*
  - *path layer (F3)*

Error detection/correction

- Error detection
  - Partity-calculation
  - „Internet checksum“ sum of all 1-compliments
  - CRC: cyclic redundancy check (LFSR)
    - Used in nearly all link-level protocols
    - Ethernet is using a well known polynomial of degree 32 (giving strong protection against common bit errors in messages thousands of bytes long)
    - (n+1) bit messages – represented by an *n* degree polynomial M (x) = $x^7+x^4+x^3+x$ $n=7$
    - C(x) divisor polynomial C(x) = x^3+x^2+1 $k=3$
    - Idea: T(x) = M(x).x^k … add *k* zeros at the end
    - transmit: M(x) .x^k - R(x) | R(x)=T(x)/C(x)

Error detection/correction



- Calculation rules:
  - B(x) can be divided by C(x) if B(x) is of higher degree than C(x)
  - B(x) can be divided once by C(x) if B(x) and C(x) are of the same degree
  - Based on modulo-2 binary division:
    - » R(x) = B(x)/C(x) = B (x) – C(x)
    - » B(x) – C(x) = B (x) EXOR C(x)

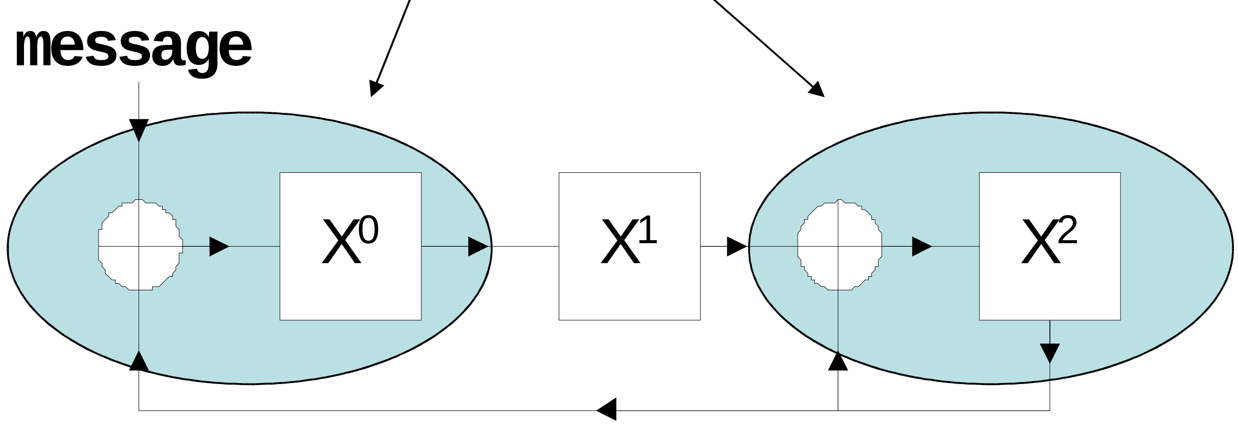
http://www.cs.nmsu.edu/~jcook/Classes/DE-CS484/DataLink-3.html

Error detection/correction

- $C(x)$ = ???
- $M(x)$ … message $M(x) + E(x)$ $E(x)$ …error
- … can $E(x)$ be divided by $C(x)$???
- Single bit error: $E(x) = x^i$
  - … can be detected if first and last term of $C(x)$ is non zero, thus, $C(x)$ = 2-term polynomial an cannot be divided by $E(x)$
- All single bit errors: x^k and x^0 have nonzero coeff.
- All double bit errors: as long as $C(x)$ has a factor with at least 3 terms
- All odd number errors: $C(x)$ contains $(x + 1)$
- Any burst error: length of the burst is less than *k*

Error detection/correction

- CRC can be implemented in HW using a *k*-bit shift register and XOR gates.
- Number of SR is equal to *k*
- Example: $C(x) = 1+x^2+x^3$ $k=3$

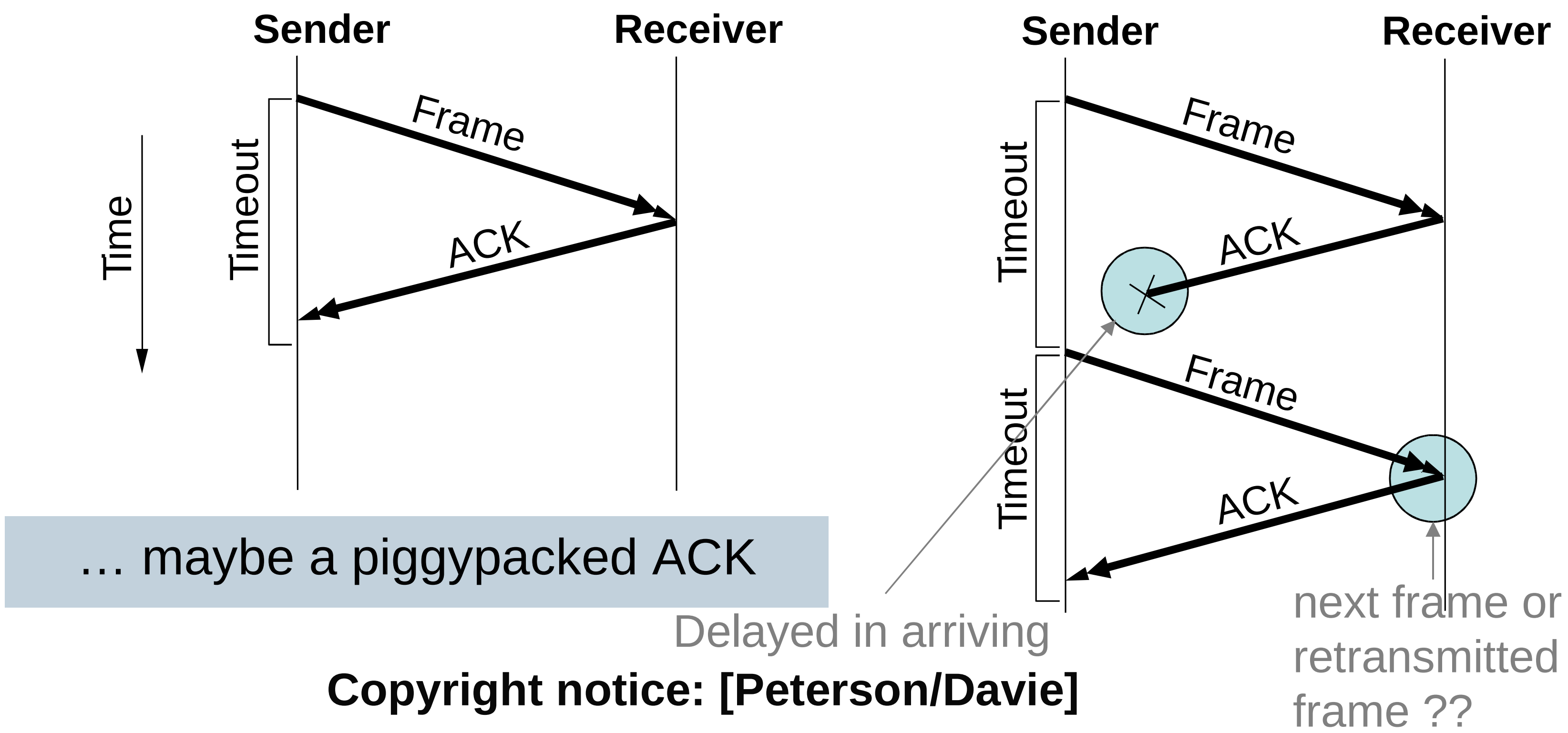


Error detection/correction

- Error detection
  - Ethernet and 802.5 use CRC-32
  - HDLC uses CRC-CCITT
  - ATM uses CRC-8, CRC-10, and CRC-32
- Error correction
  - FEC: Forward error correction
    - ECC: error correcting codes
  - ARQ: Automatic repeat request
    - Stop & Wait vs. window-system

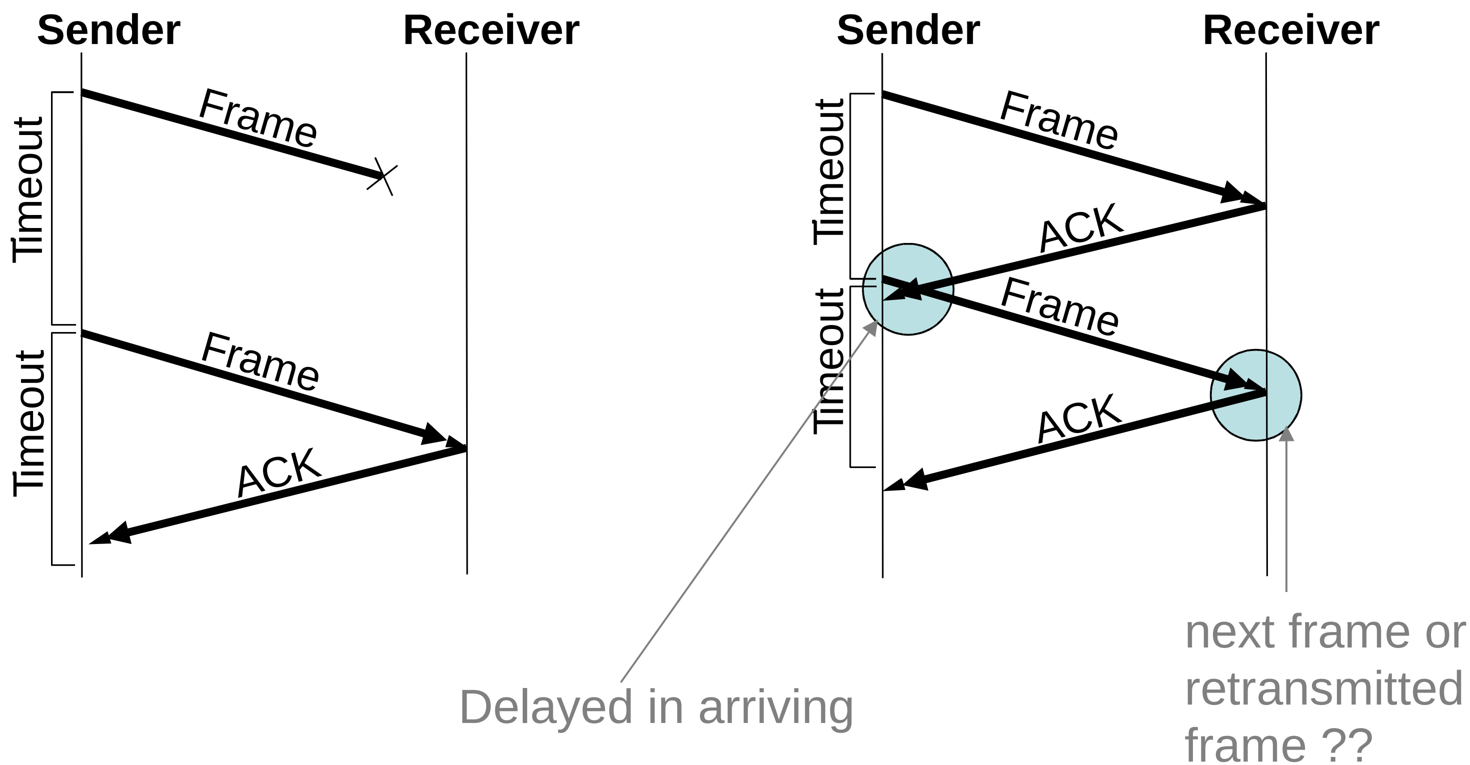
Stop and wait

- Sender is waiting for an „ACK“ (on each frame)
- Based on *ACK’s* and *timeouts* (known as ARQ)
  - … wait before transmitting the next frame
  - If the ACK does not arrive -> start retransmission



Stop and wait (cont.)

-



Copyright notice: [Peterson/Davie]

Stop and wait (cont.)

1. Problem: retransmission of acknowledged frames due to timeout or ACK-lost.
   – use a one bit sequence number to prevent duplicated frames
2. Problem: just one outstanding frame allowed (*bandwidth x delay* pipeline not filled)

Example: 1.5 Mbps link RTT=45ms

RTT x C = 67.5 Kb (~8KB)

… since just 1 frame per RTT can be sent

assume frame size = 1KB

max. sending rate = 1024*8 / 0.045 = 182Kbps

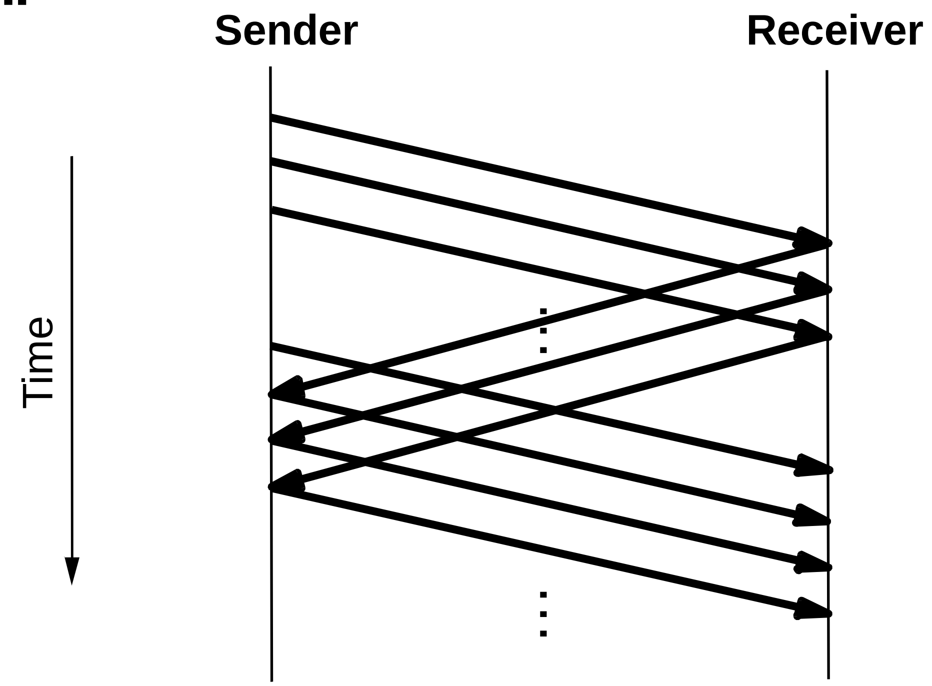
? Sliding Window

- Sending a whole window without „ACK“
- remember:

  RRT x C = 8KB

  1KB frame size

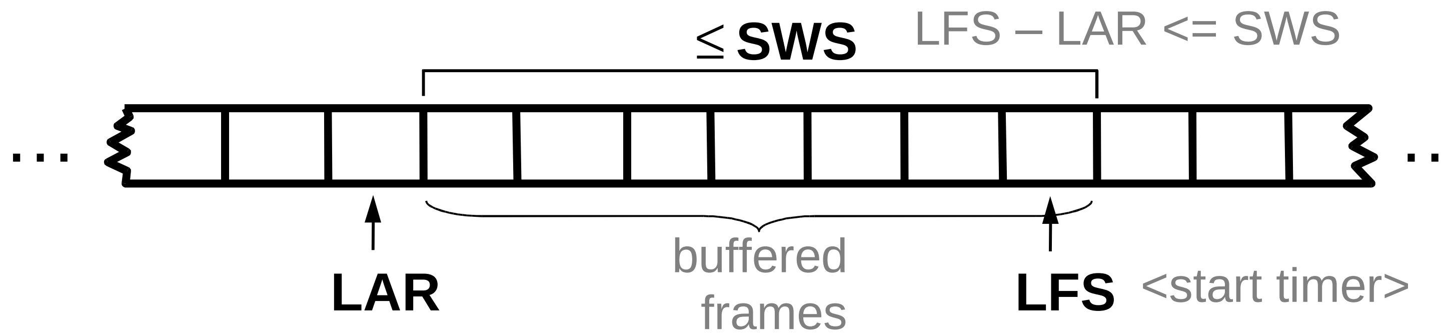
  -> we can send eight frames without ACK



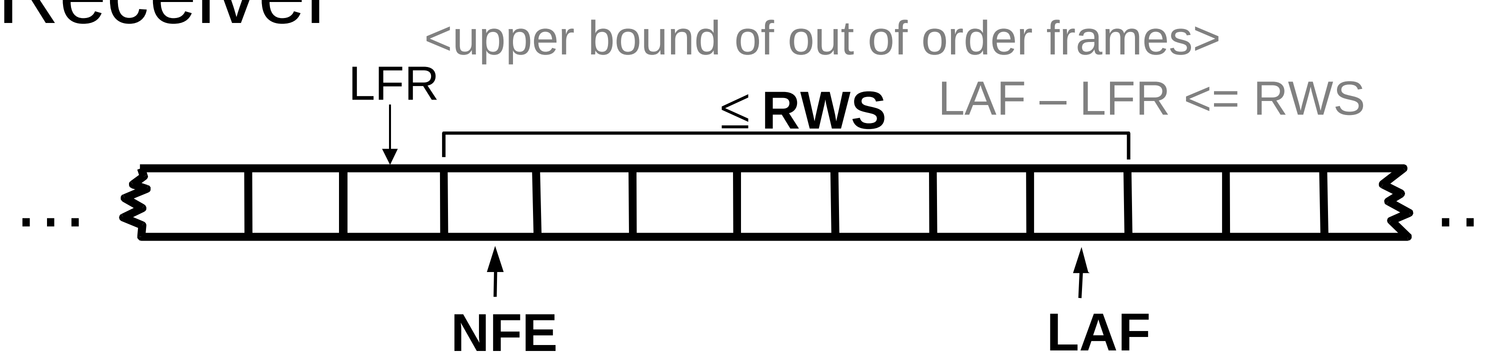
Sliding Window (cont.)

SWS	**send window size**
LAR	**last acknowledgement received**
LFS	**last frame sent**

- Sender



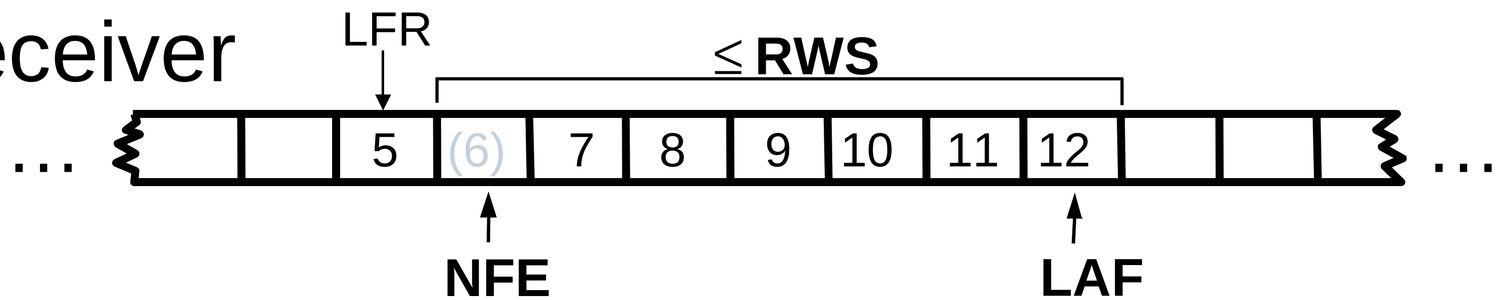
- Receiver



RWS	**receiver window size**
NFE	**next frame expected**
LAF	**largest acceptable frame**

Sliding Window (cont.)

- Receiver




```
if (SeqNum <= LFR || SeqNum > LAF)
      //frame is outside receiver window
      discard_frame (Seq_Num);
else  //(SeqNum > LFR && SeqNum <= LAF)
      accept_frame(SeqNum);
send_frame_ack(SeqNumToACK); //cumulative
LAF = LFR + RWS;
```

RWS	**receiver window size**
NFE	**next frame expected**
LAF	**largest acceptable frame**

Sliding Window (cont.)

1. If a packet loss has occurred, this scheme is no longer keeping the pipe full
2. In this example we can send a NAK for frame 6 as soon as 7 arrives – but this is not necessary since there is the sender timeout mechanism to catch this situation.
3. It is also possible to send additional ACK's for 5, when 7 and 8 arrives (as an indicator for lost frames)

2. and 3. can be used to improve performance!!!

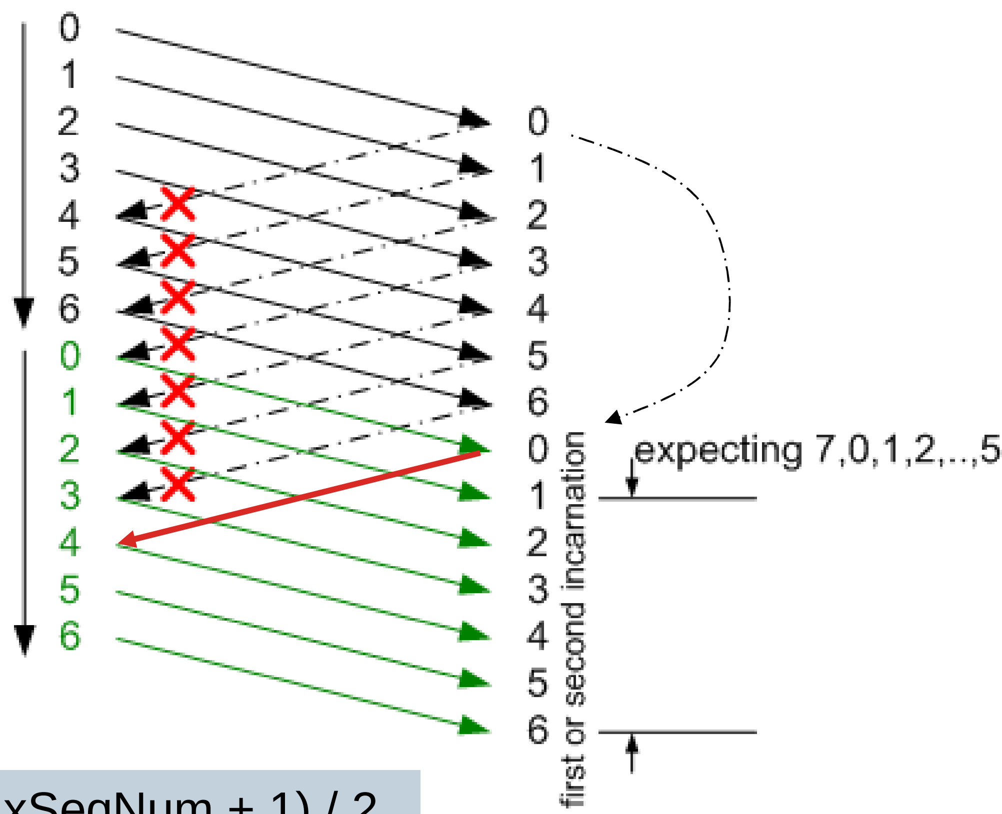
Sliding Window (cont.)

- Selective ACK's: acknowledging exactly received frames (giving more information to the sender)
- SWS is easy to compute for a given RRT x C at the sender
- Receiver can set RWS to whatever it wants
- RWS = 1; // no frame buffering
- RWS = SWS; // receiver can buffer any frames
- RWS > SWS; // not really useful

Sliding Window (cont.)

- Finite Sequence Number
  - Sequence numbers must fit into the header (just a few bits)
  - SWS <= MaxSeqNum – 1 // is this sufficient?
  - If RWS = 1 (ok)
  - If RWS = SWS (not ok), since:

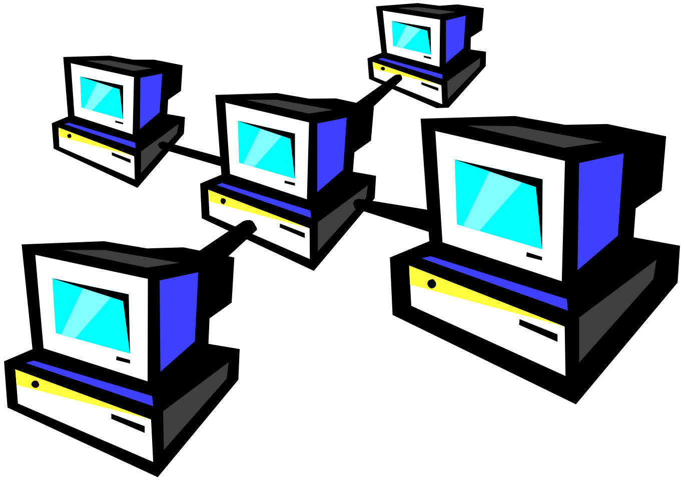
SeqNum = 0 .. 7
maxSeqNum = SWS = RWS = 7



SWS <= (MaxSeqNum + 1) / 2

Local Networks

- **Classification**
- **Example:**
  - Ethernet DIX vs. 802.3



Ethernet (DIX vs. 802.3)

Carrier-sense multiple access collision detection

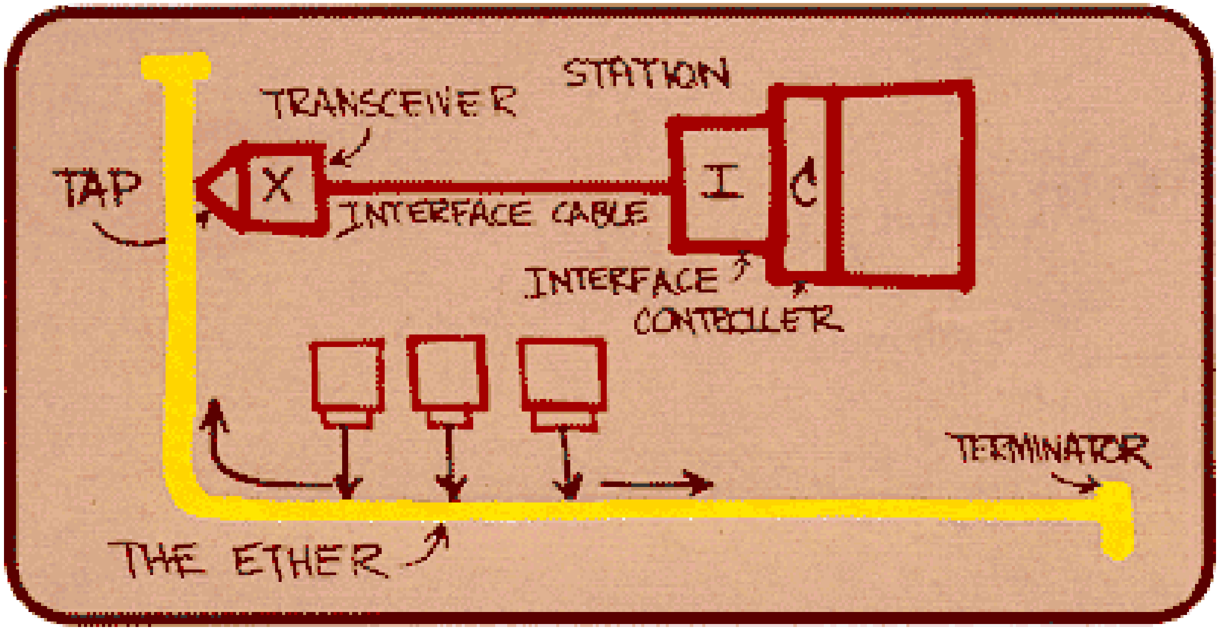
- … most successful LAN
- Researched @ Palo Alto Research Center (PARC)
- Working example of CSMA/CD
- Has its roots in the "Aloha Protocol" (radio network)
- Core idea: find an algorithm to control node transmission in a "collision domain"
- DEC, Intel, Xerox defined 10 Mbps Ethernet in 1978
- DIX Ethernet was forming the base for IEEE 802.3
  - (differences in the number of supported physical media)

Ethernet Physical Properties

- DIX segment: coax cable up to 500m @ 50 Ω
- Taps must be at least 2.5m apart
- Transceivers are attached to the taps
- Multiple Ethernet segments can be joined by repeaters (not more than 5 – total reach of 2500m)
- Signals are broadcasted over all segments
- Ethernet uses Manchester encoding
- 10base5 (thick-net), 10base2 (thin-net) @ 10Mbps
- “5”- not longer than 500m
- “2” - not longer than 200m

Ethernet Physical Properties

- Today: 10BaseT (twisted pair – CAT5 @ 100m)
- 10BaseT is using Hubs (no “daisy-chaining”)



•Robert M. Metcalfe (1976)

Ethernet
Access Protocol

- **MAC** (Media Access Protocol): algorithm to control the access to a shared Ethernet link
- Implemented in HW
- DIX Frame Format:

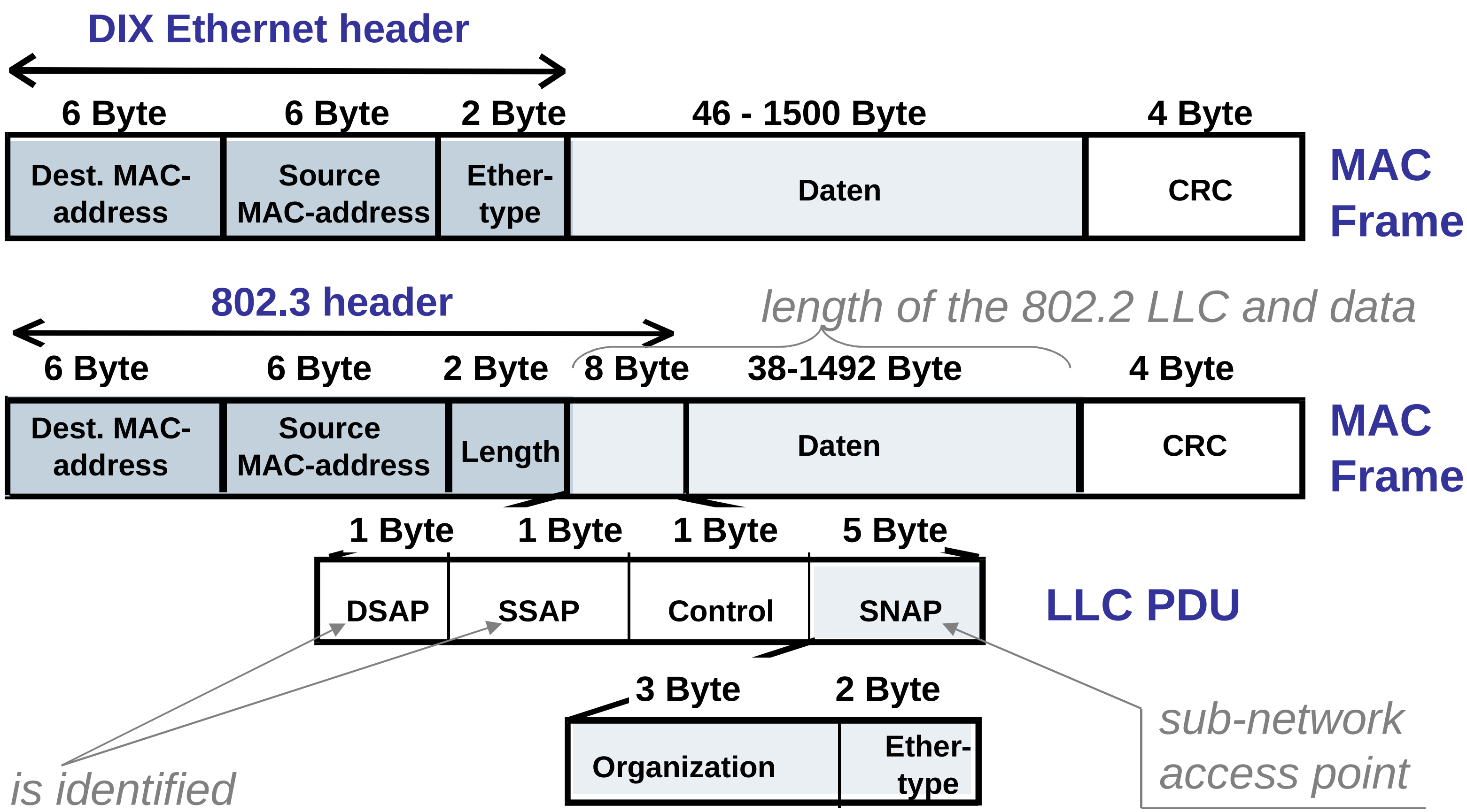
64	48	48	16		32
Preamble	Dest Addr	Src Addr	Type	Body	CRC

  - Preamble: for synchronization (sequence of 0s and 1s)
  - 48 bit source & destination address (MAC address)
  - Type: definition of higher-level protocols
  - Body: up to 1500 bytes of data (minimum of 46 bytes – to detect collisions)
  - CRC: 32 bit CRC (bit oriented framing protocol)

Ethernet
Access Protocol

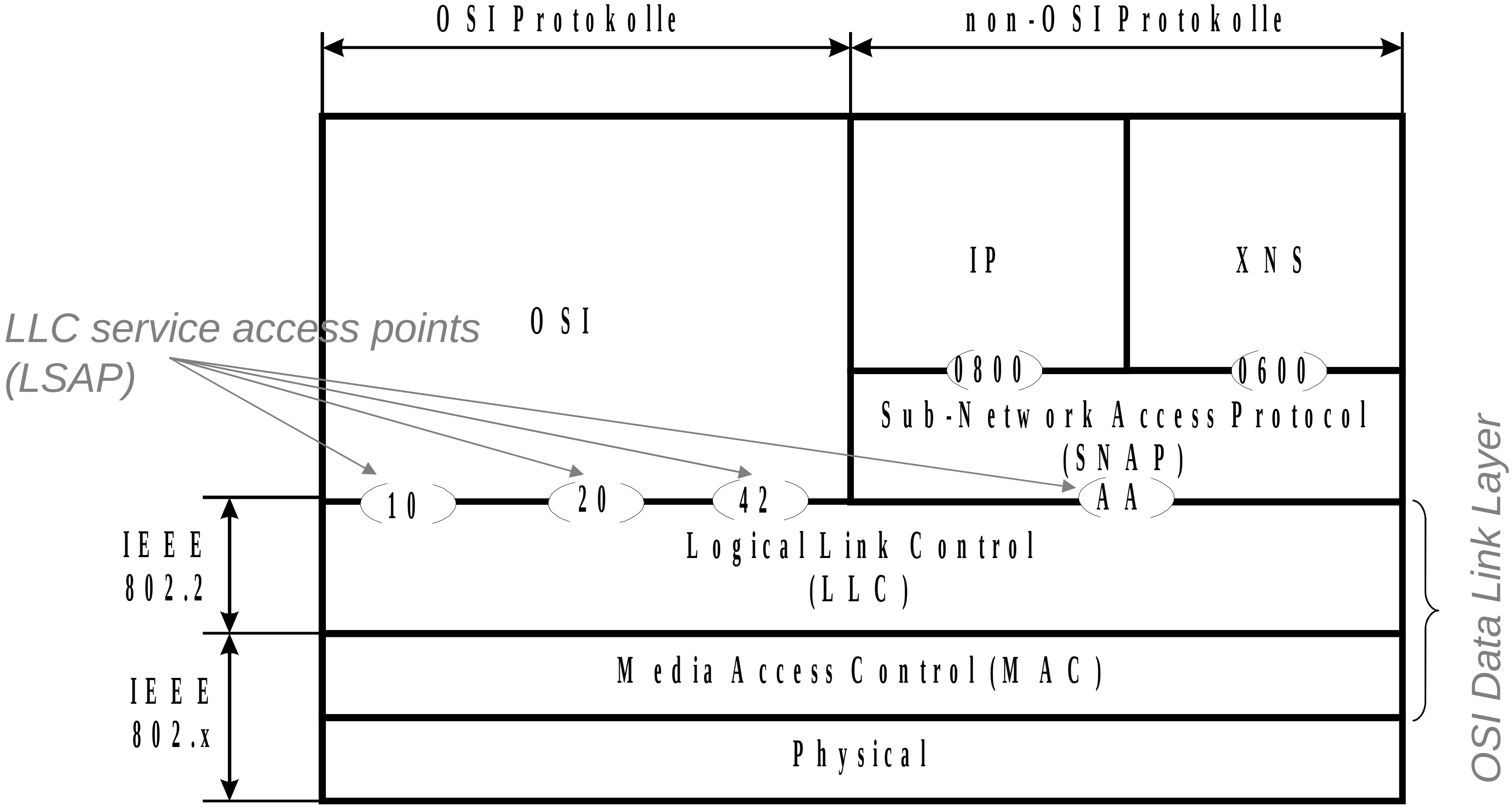
- 802.3 frame is exactly the same but …
- 16 bit length field for 16 bit type field
- Type field is the first thing in the data portion
- <u>Not type value is less than 1500</u> (*maximum length of a 802.3 header*)
  - We can distinguish between 802.3 and DIX
  - We can support both by using a single adapter (this is done in SW)

Ethernet
DIX vs. 802.3



Each user is identified
by a service access point SAP
(1 byte destination/source service access point – just 256 standard values!!)

Ethernet
802.3 LLC, SNAP, SAP



Ethernet DIX vs. 802.3

Ethernet 802.3/802.2 with SNAP

ipx encapsulation sap

- … even IP couldn't get a standard SAP -> use of Subnet Access Protocol SAP (SNAP)
- SNAP:

IPX

DSAP	SSAP	Control	Org. code	Type	Data
0xAA	0xAA	0x03	OUI	0x8137	checksum 0xFFFF, IPX-header

Ethernet 802.3/802.2 without SNAP

ipx encapsulation sap

DSAP	SSAP	Control	Data
0xE0	0xE0	0x03	checksum 0xFFFF, IPX-header

Ethernet MAC Addresses

- Each host (in the world) has a unique address
- Address belongs to the adaptor (not to the host)
- Each manufacturer uses a different prefix
- The adaptor recognizes *unicast addresses* and passes those frames to the adaptor
  - vs. *promiscuous mode:* deliver all received frames to the host.
- The adaptor recognizes *broadcast addresses* and passes those frames to the adaptor
  - all bits set to “1s”
- If no broadcast address, but the first bit is set to “1” an set of adapters can be programmed to accept those *multicast addresses*

Ethernet Transmission Algorithm

- If there is a frame to send, send it immediately
  - No negotiation with other adaptors
- If there is a frame to send, and the line is not idle, wait for the line to go idle and transmit immediately (thus, Ethernet is a *1-persistent protocol*) *(vs. non-, p-persistent)*
- If 2 or more adapters start the transmission at the same time (both found the line to be idle) they will collide on the network
- If collision: adaptor has to send a 32 bit jamming sequence (64 preamble + 32 jamming = 96 bit runt-frame)

? Ethernet Transmission Algorithm

- To make sure that no collision has occurred, we have to send at least 512b (64B = 14B header + 46B data + 4B CRC) but why??

At (t+2.d) "A" knows that the frame is corrupted

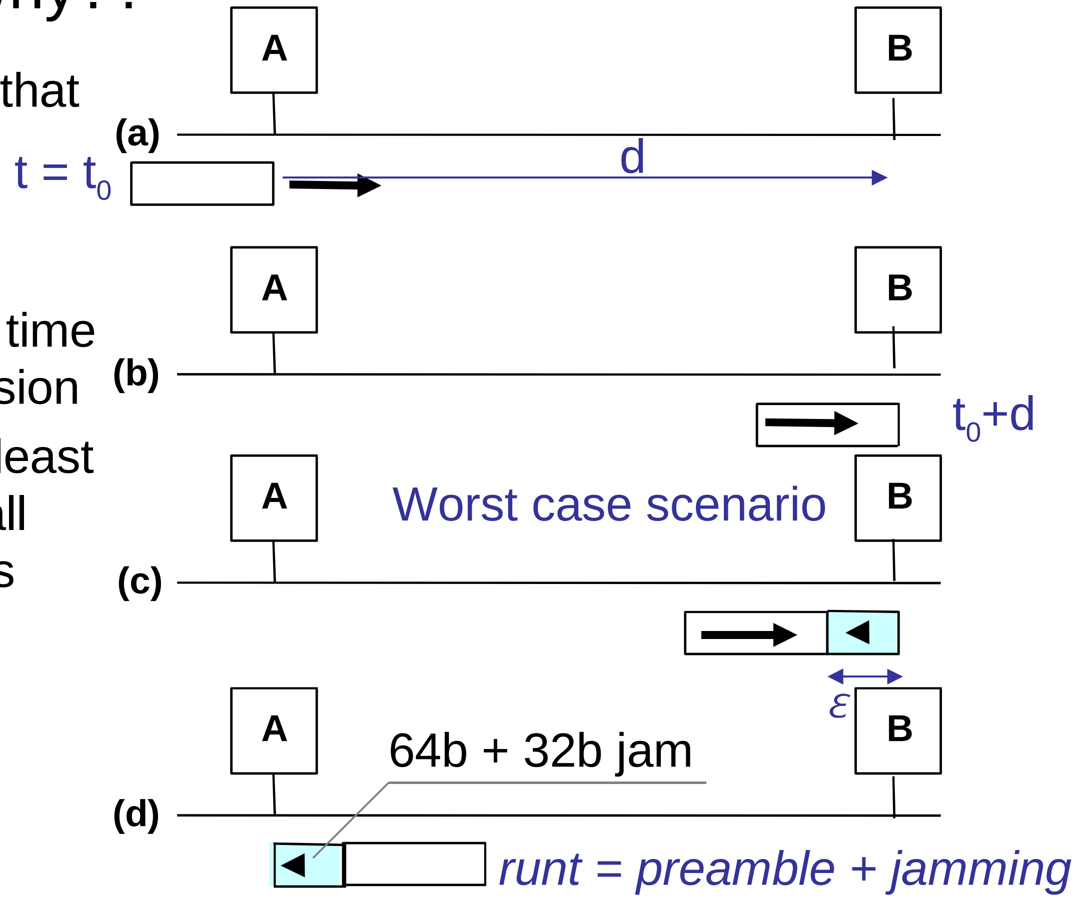
"A" must continue to transmit until this time to detect the collision

"A" must transmit at least for 2.d to detect all possible collisions

Length = 2500m

RTT = 51.2 μ s

512b @ 10 Mbps



Ethernet Transmission Algorithm

- After collision detection:
  - stop transmission
  - wait a certain amount of time
  - try again
  - If retransmission fails, double the amount of time (*exponential backoff*)
    - First delay: wait for 0 or 51.2 μ s
    - If this fails, then wait: 51.2, 102.4, or 153.6 μ s (randomly)
    - After third collision it waits k.51.2 for k = 0..2^3-1 (randomly)
    - In general: select k between 0..2^n-1 and wait randomly k.51.2
      - n = number of collision expected
    - Give up after a given number of tries and report a transmission error (typically after 16 times, n = 10)

Fast Ethernet

- CSMA/CD System, IEEE 802.3
  - 100BASE-TX
  - 100BASE-FX
  - 100BASE-T4
- Variant 100VG-AnyLAN, IEEE 802.12
  - *Demand priority*
- Gigabit Ethernet IEEE 802.3z (1998)

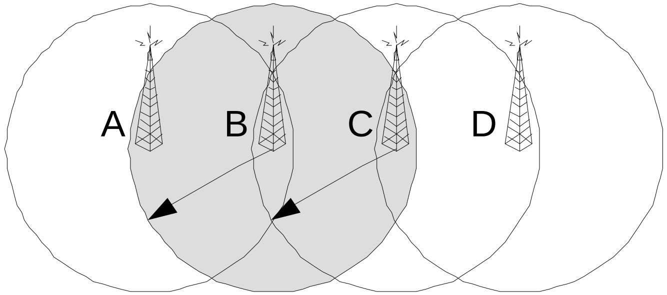
Wireless (802.11)

- Designed for use in limited geographical areas
- Access to a shared communication medium
  - Additional services are required: time-bounded services, power management, security mechanisms.
- Standard based on spread spectrum
  - (FHSS) frequency hopping (random sequence of frequencies)
  - (DSSS) direct sequence (data_bit XOR random_bits, *using a n-bit chipping code*)

… or diffuse IR
  - sender and receiver do not have to be aimed at each other – no clear line of sight)
  - range up to 10m

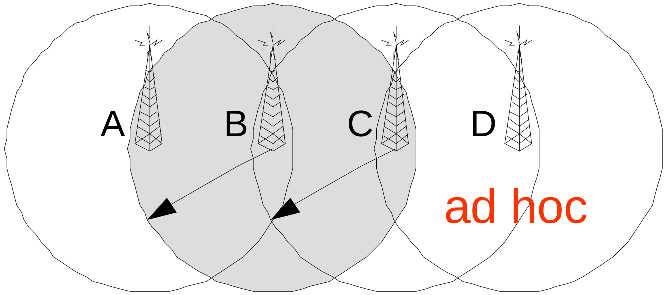
Wireless (802.11)

- Collision avoidance
  - Follow exactly the same algorithm as Ethernet
  - … but not all nodes are always within reach of each other



  - Suppose A and C want to talk to B
    - A and C are unaware of each other
    - Frames collides at B but unlike Ethernet, neither A nor C is aware of this collision (*hidden hosts*)
  - Suppose B sends to A,
    - C is aware of this. C would err, if C concludes that it cannot transmit to D, since it can hear Bs transmission (*exposed node problem*)

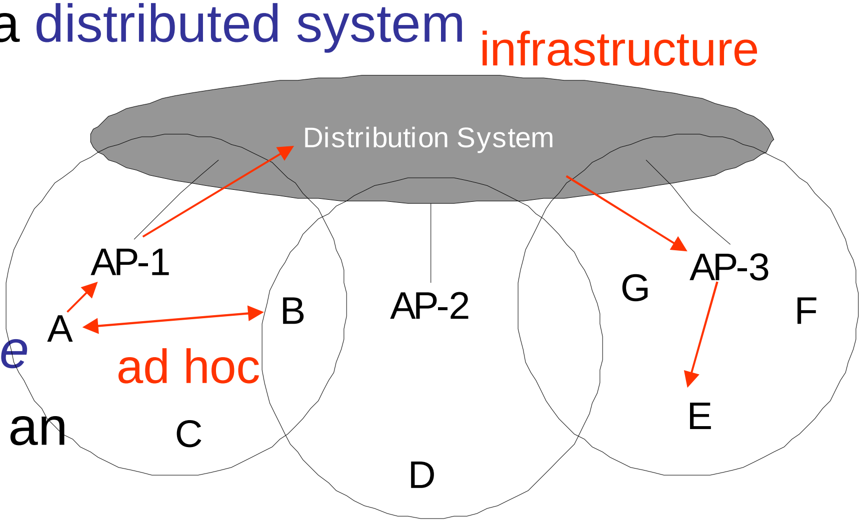
Wireless (802.11)



- Suppose C wants to transmit to D
  - This is not a problem, if B is sending
  - May be a problem, if B receives from A
- 802.11 addresses the hidden host- and exposed node problem with multiple access with collision avoidance (MACA)
  - Exchange control frames before starting the transmission
    - Request to Send frame (RTS) to inform the receiver about the frame length *(how long the sender wants to hold the medium)*
    - Receiver replies with Clear to Send (CTS)
    - Any node that sees the CTS knows that it is close to the receiver and cannot transmit for a certain period of time
    - Any node that sees RTS but not CTS is not close enough to the receiver to interfere with it (free to transmit)

? Wireless (802.11)

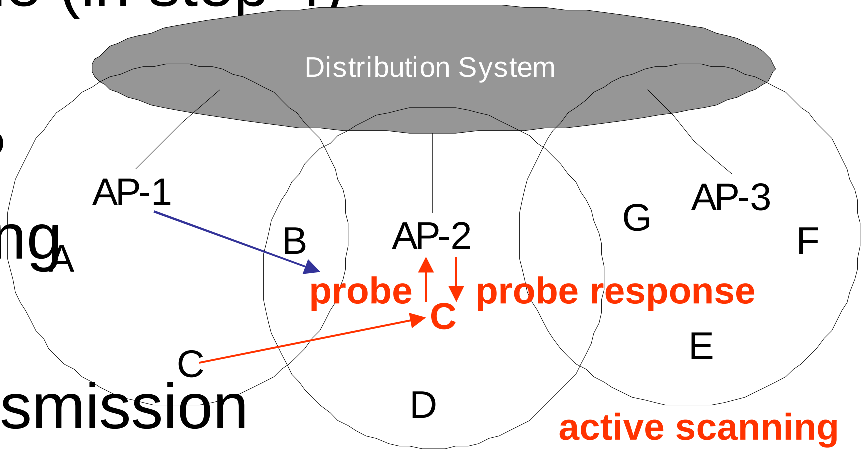
- Distributed System
  - Nodes are free to move around (roaming)
  - Some are connected to the wired network (Access Point)
    - Connected to each other by a distributed system
  - Scanning:



1. Node sends a *Probe Frame*
2. All APs (within reach)
   respond with a *Probe Response*
4. Node selects one and sends an
   Association Request Frame
6. AP replies with *Association Response*

Wireless (802.11)

- This procedure is started by the node: after each *restart*, and when it becomes *unhappy* with the AP.
  - Due to roaming
  - New AP notifies the old one (in step 4)
  - Active scanning: node is actively searching for an AP
  - APs are periodically sending *Beacon frames* that advices the capability of the AP (transmission rate,…). A node can change to this AP by sending an Association Request (*passive scanning*)

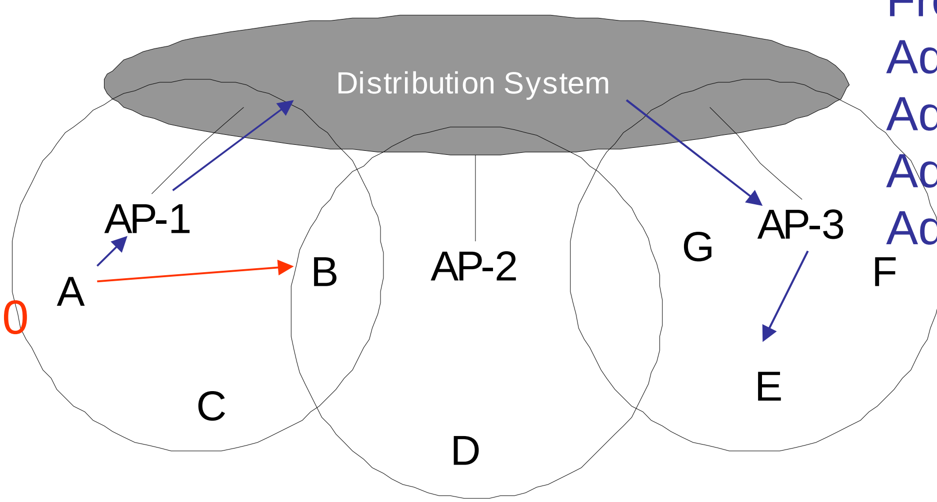


Wireless (802.11)

- Frame Format

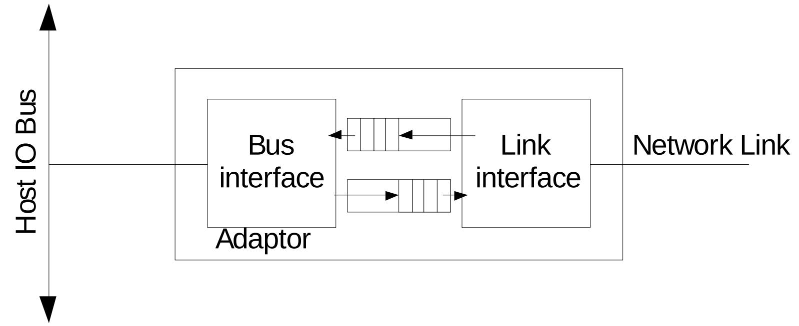
16	16	48	48	48	16	48	0-18,496 b	32
Control	Duration	Addr1	Addr2	Addr3	SeqCtrl	Addr4	Payload	CRC

  – Control field contains 3 subfields:
    - 6b Type field (indicate whether the frame carries data, or is an RTS or CTS frame, or is used for scanning)
    - ToDS 1b
    - FromDS 1b



Network Adapters

- We want to discuss the design of a generic network adaptor and the device driver that control it …

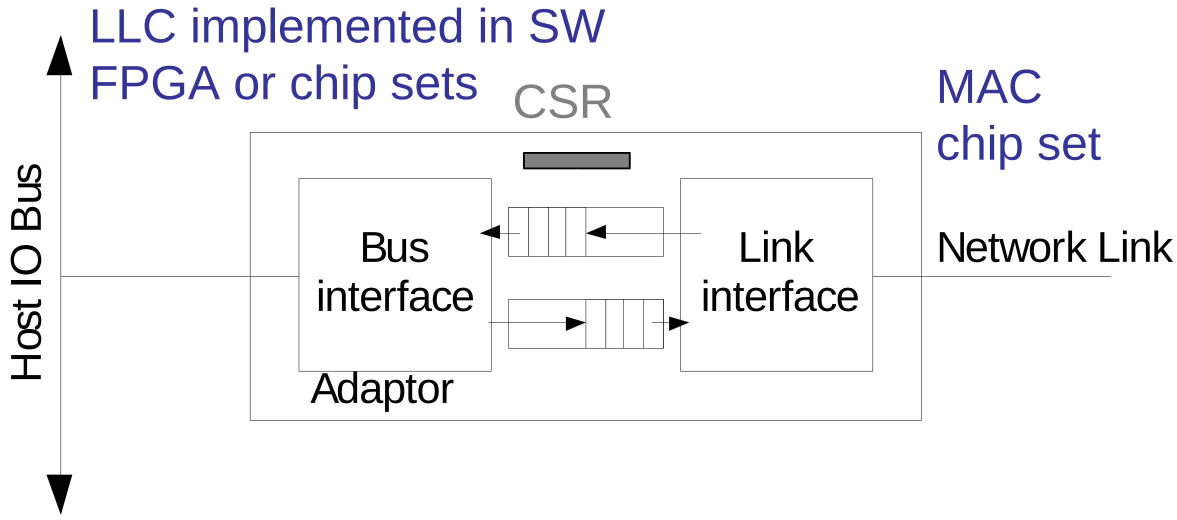


- Components
  - Adaptor is the IF

  between the host and the network
  - Designed for a specific IO bus which is used:
    - by the hosts CPU to program the adaptor
    - by the adaptor to interrupt the hosts CPU
    - by the adaptor to read and write the host memory
  - This IF is limiting the transfer rate …

Network Adapters

- Example:
  - 32 bit bus @ 25 MHz (bus cycle time is 40 ns)
  - Giving a peak transfer rate of 32 x 25 E^6 = 800 Mbps
  - … enough for a 622 Mbps STS-12 link (unidirectional)
  - … tells us nothing about the average rate



Network Adapters

- View from the host
  - Control status register
    - Network device driver is implemented in software
    - CSR -> set by the CPU to transmit or receive frames
  - Interrupts
    - CPU could poll the CSR until something interesting happens
      - Useable for routers but not for end hosts
    - Adaptor can interrupt the host if there are changes in CSR
      - Interrupt handler is called (interrupts are disabled)
      - Interrupt handler is queuing the “defered interrupt service routine”
      - checking the content of CSR – setting actions
      - Dispatch a process (or thread) to perform the protocol stack functions

Network Adapters

- View from the host
  - Direct Memory Access vs. Programmed IO
    - How can we transfer frames between adaptor and host memory
    - DMA directly reads/writes without CPU
    - PIO the CPU is responsible for data transfer
    - DMA:
      - No buffers are required – adaptor reads and writes host memory
      - CPU is responsible to give the adaptor a pair of *buffer descriptor lists* (one to transmit and one to receive)
      - Separate frames are placed in separate buffers, although …
      - … a single frame can be scattered across multiple buffers *scatter-read*
        - » Is not used on an Ethernet, since pre-allocating 1500B is no waste of memory
      - Output works in similar ways: *gathered-write*

Network Adapters

- View from the host
  - Direct Memory Access vs. Programmed IO
    - PIO:
      - Adaptor must contain some buffers (memory)
      - Since OS-scheduler decides the time of execution, we have to prepare the right amount of buffer (but how much??)
      - PIO adaptors usually have some additional memory that can be used
      - Memory is not cheap, since *dual ported* RAM is needed.
      - Typically 64-256kB of adaptor memory

Device Driver

- Routines to initialize the adaptor
- Routines to transmit frames on the links
- Code is sometimes difficult to read (due to device specific details)

-
```
#define csr ((u_int *) 0xffff3579) //init control status reg.
lance-transmit (Msg *msg)
{       discriptor *d;
        semWait(xmit_queue);   //number of transmit buffers (64)
        semWait(mutex);        //mutual exclusion
        disable_interrupts();  //protects from the device
        d = next_xmit_descr();
        prepare_xmit_desc(d, msg);
        csr = LE_TDMD | LE_INEA; //instruct the device to transmit
        enable_interrupts ();
        semSignal (mutex);
}
```

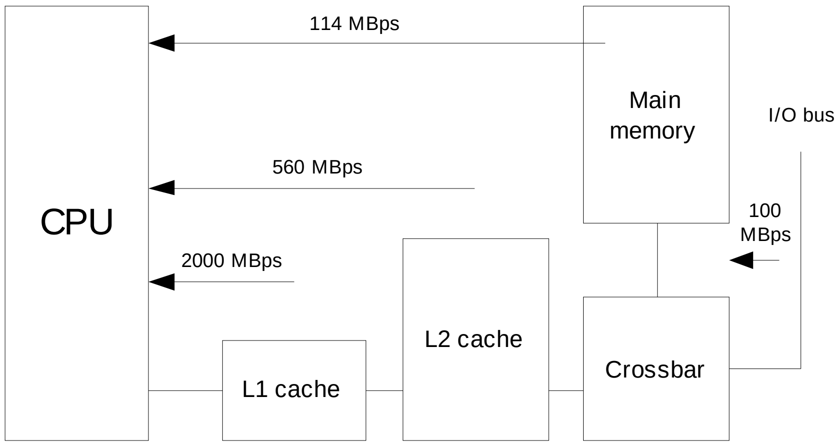
```
LE_TDMD: transmit demand - send it now
LE_INEA: read/write interrupt enable
```

Device Driver

```
lance_interrupt_handler)
{       disable_interrupts();
        if(csr & LE_ERR){               //clear error bits
             csr = LE_BABL | LE_CERR | LE_MISS | LE_MERR | LE_INEA;
             enable_interrupts();
             return;
        }
        if(csr & TINT){                 //transmit interrupt
             csr = LE_TINT | LIEA;      //clear interrupt
             semSignal(xmit_queue);     //signal blocked sender
             enable_interrupts();
             return();
        }
        if(csr & LE_RINT){              //receive interrupt
             csr = LE_RINT | LE_INEA;   //clear interrupt
             lance_receive();           //receive frame
             enable_interrupts();
             return();
        }
}
```

Device Driver

- Memory Bottleneck



1. Overhead in IO bus transfers (8 cycles to acquire the bus, 12 cycles to transfer data for 48 byte ATM cells)
   – *32 bit bus: 4 byte wording during each clock cycles*
   – 32bit I/O bus @ 25 MHz = 800Mbps
   – 12 / (8 + 12) x 800 = 480 Mbps !!!
3. Memory/CPU bandwidth: 114 MBps (956 Mbps)
   – Slightly more than I/O bus (would be OK, but …)
   – Copying form one buffer to another @ 114MBps
   n-times: 114MBps/n = 22MBps | n=5

Packet Switching

- Switching and forwarding
- Frame Relay
- ATM
- ATM switches

Slides vs. [Peterson/Davie]

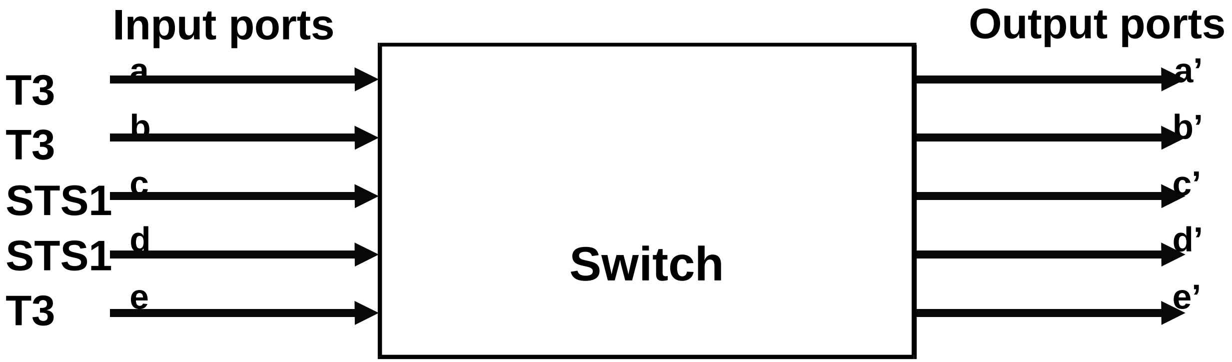
- … making a comparison:
  - Slides (Switching/Forwarding)
    - [Peterson/Davie] Section 3.1
  - Slides (Frame Relay)
    - [Peterson/Davie] Section 3.1.1
    - [Peterson/Davie] Section 3.2 (Bridges/LAN Switches)
  - Slides (ATM) corresponds to
    - [Peterson/Davie] Section 3.3
  - Slides (ATM switches) corresponds to
    - [Peterson/Davie] Section 3.4

Switching & Forwarding

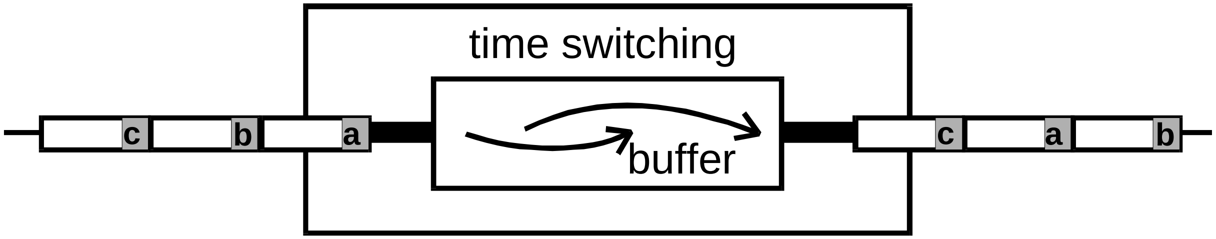
- Basics, Datagram's
- Virtual Circuit Switching
- Source Routing

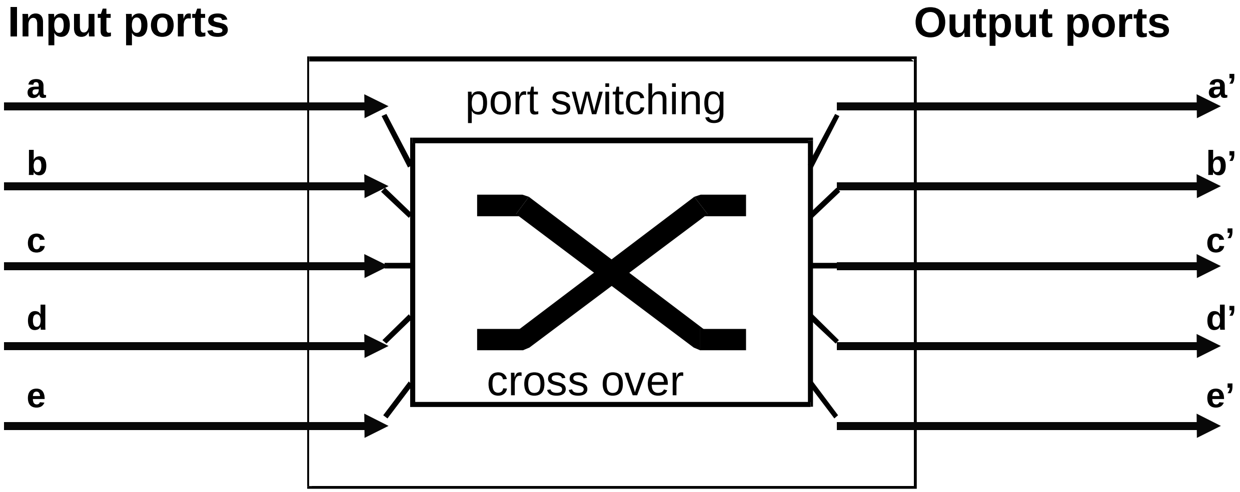
Switching (the basic principle)

- Packet "forwarding"
  - Datagram, Frames, Cells
  - How does the switch decide which output port to place which packet on?
    - Datagram (connectionless service)
    - Virtual circuit (connection oriented)
    - Source routing



Time- vs. Port Switching





A switch is called **“internally blocking”** if cell or packet loss occurs, due to high data rates at the input ports

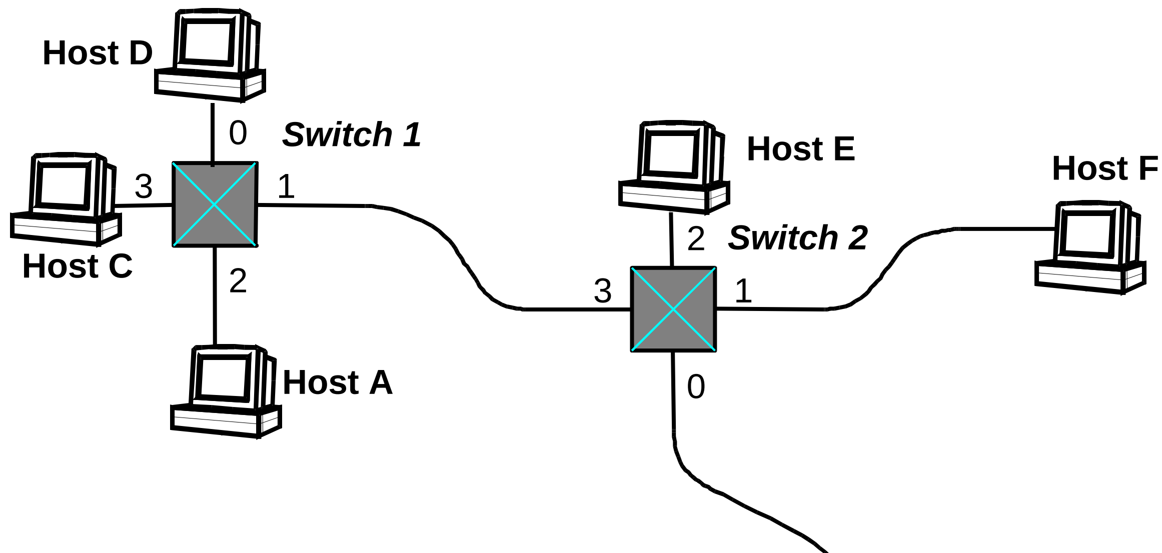
Switching (the basic principle)

- **Datagrams**
  - Every packet contains a complete destination address.
  - Switch contains a forwarding table (routing table)
    - Host can send at any time (switch can immediately forward – assuming a correct populated forwarding table) in contrast to connection-oriented networks
    - … now way to know, if the network is capable to deliver the packet – or if the host is up
    - forwarding - independent of previous packets – two successive packets may follow different paths (due to changes in the forwarding table)
    - switch or link failure may not have serious effects (routing protocols – important goal of the ARPANET)

Forwarding-Tables

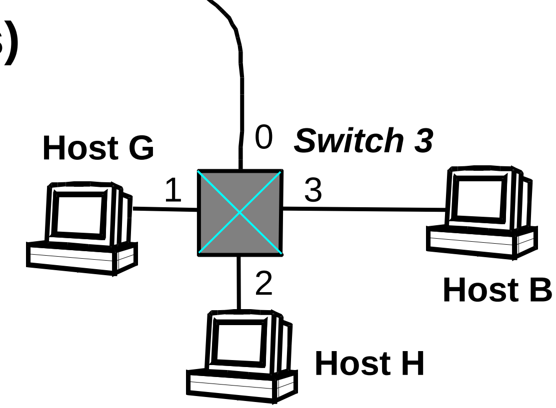
Switch 1

Destination	port
Host A	**2**
Host B	**1**
Host C	**3**
Host D	**0**
Host E	**1**
Host F	**1**
Host G	**1**
Host H	**1**



Comment.: **Switches (bridges) can learn forwarding tables**

- **e.g. Spanning tree (Ethernet)**
- **e.g. source routing (Token Ring)**



? Switching (the basic principle)

- **Virtual Circuit Switching**
  - Based on the connection oriented model
  - Virtual connection has to be established first
  - Two stage process: (1)connection setup, (2)data transfer
    - Permanent virtual circuits (PVC) - established by administrators
    - Signalling to establish a switched virtual circuit (SVC)
      - Host may setup and delete such circuits
      - … would rather call it “Signalled” VC (not switching)

Switching (the basic principle)

- Connection state is an entry in a VC table (for each connection) – such entries contains:
  - Incoming interface
  - Virtual circuit identifier (VCI) – will be carried in the arriving packet
  - Outgoing interface
  - A VCI that will be used for outgoing packets
- incoming IF and incoming VCI uniquely identifies the virtual connection
- For each new connection, we need a new VCI
- Incoming an outgoing VCIs are not the same
- VCI is not a globally significant identifier for the VC

Permanent Virtual circuits

Switch 1

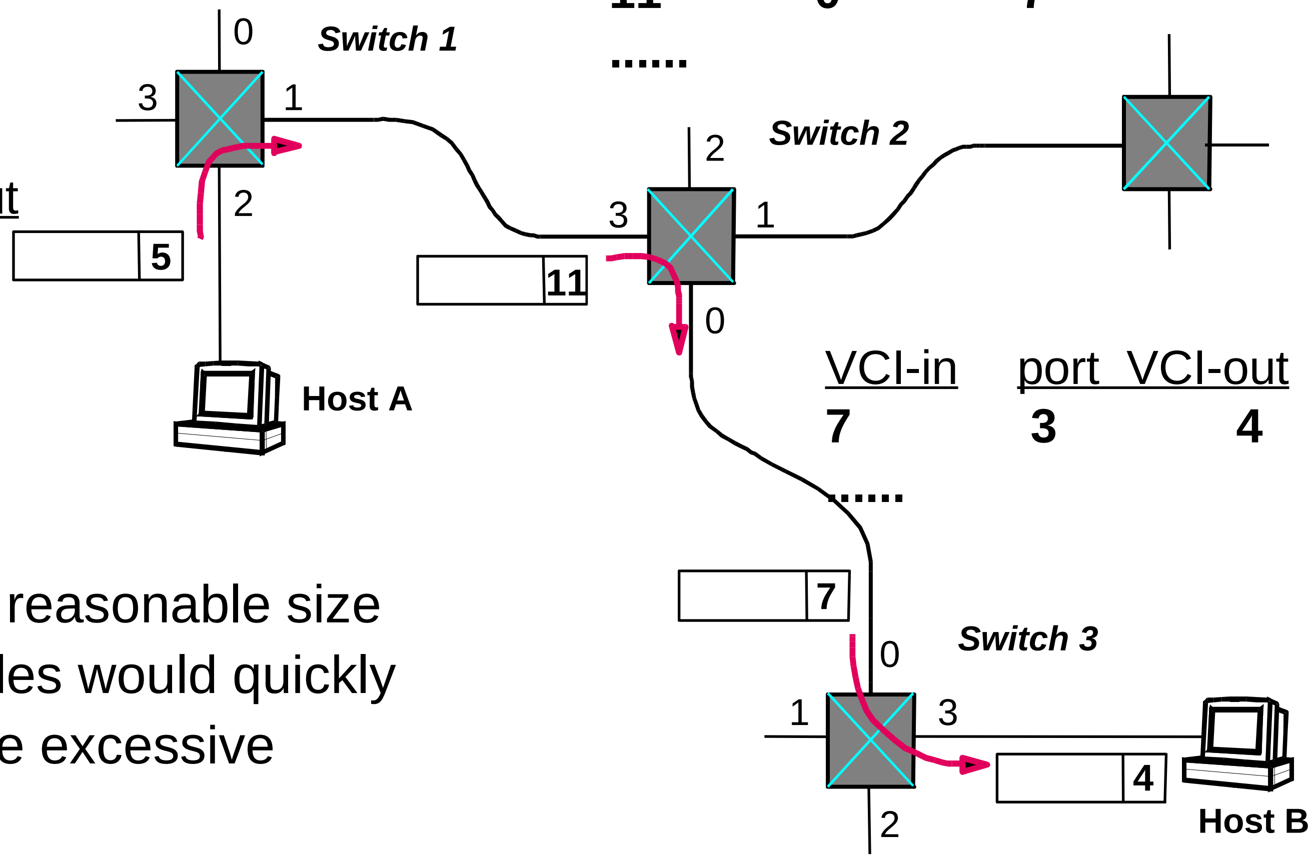
VCI-in	port	VCI-out
5	1	11
12	0	7
......		

Switch 2

VCI-in	port	VCI-out
11	0	7
......		

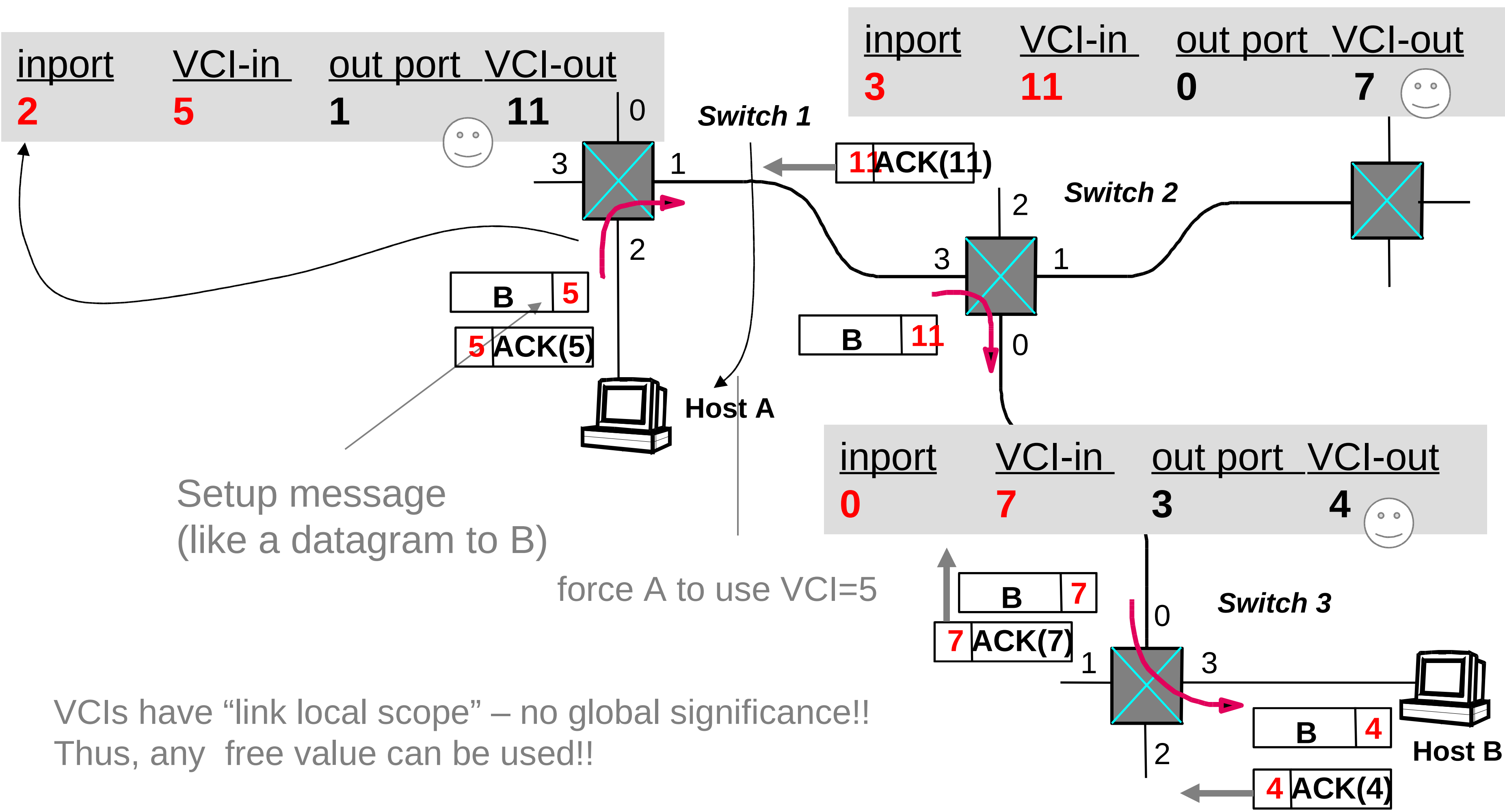
Switch 3

VCI-in	port	VCI-out
7	3	4
......		



In real networks of reasonable size configuring VC tables would quickly become reasonable excessive

Switched Virtual circuits



VCIs have "link local scope" – no global significance!!
Thus, any free value can be used!!

Switched Virtual circuits

- When A no longer wants to send data, A sends a “teardown message” to B
- … each switch removes the corresponding entries from its table
- <u>Several things to note about VC switching:</u>
  - A has to wait for 1 RTT before A can send its 1 packet (1RTT delay – not really true)
  - Connection request contains a full B-address (quite larger than a single VCI)
  - If a switch or link fails, a new one will be needed
  - … the old one will have to be torn down to free table entries
  - *… the issue of how a switch decides, which link to forward the connection request will be discussed later*

Switched Virtual circuits

- It is also possible to allocate resources to the VC during connection-establishment
- For Example X.25 uses the connection oriented approach employs a 3-stage strategy:
  - Allocate buffer to each VC during initialization
  - Sliding window is running between each pair of nodes (along the VC) – perform flow control to keep the receiving node from buffer overrun
  - Reject circuit if receiver is out of buffer (during connection request)

This is called *hop-by-hop flow control*

Switched Virtual circuits

- *Comparison with datagram networks:*
  - Datagram networks:
    - … do not require connection establishment
    - Each switch processes packets independently
    - Arriving packets competes with all other packets for buffer space
    - If there is no free space – discard packet
  - VC model:
    - Provide each VC with a different *quality of service* QoS
    - Examples: X.25, Frame Relay, ATM

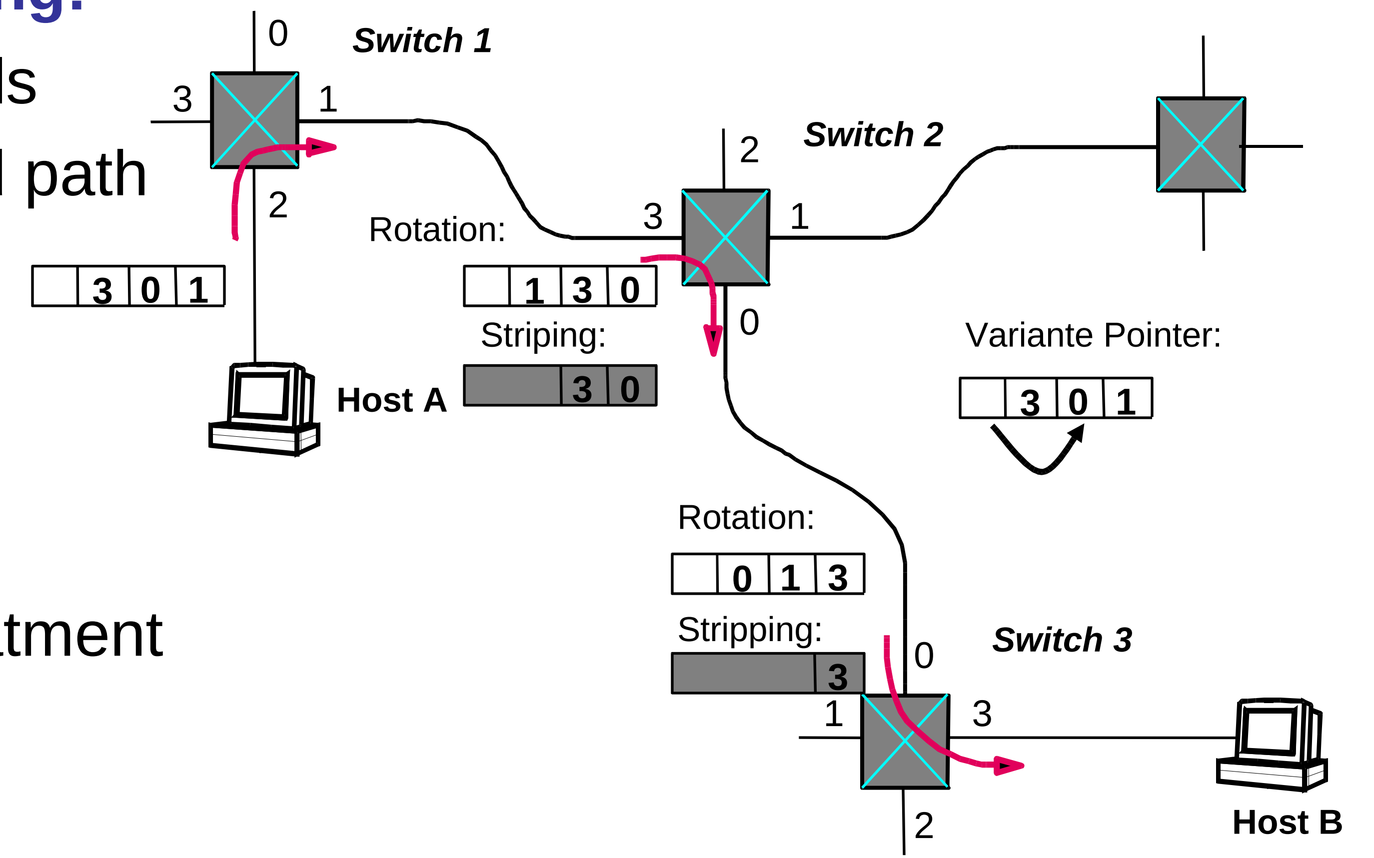
? Switching (the basic principle)

- **Source Routing**
  - … uses neither VCs nor datagrams
  - All routing information is provided by the source host
  - Assign the number of output ports in the header of the packet
  - Put an ordered list of output ports and **rotate the list** – the next switch is at the front of the list

Switching (the basic principle)

Source Routing:

Sender controls the end-to-end path by using the header



- Header-treatment
  - Rotation
  - Stripping
  - Pointer

Switching (the basic principle)

- Source routing can be used in datagram and VC networks
- IP (which is a datagram network) includes a source route option – selected packets can source routed (the rest is switched as conventional datagrams)
- Source routing is also used in SVC networks to get the initial setup along the path
- … suffers from scaling problems
  - In large networks, it is hard to get the complete path info

Bridges and LAN Switches

- LAN switches (historically they have been referred to as "bridges")
  - Put a node between two Ethernets (forwarding frames)
  - The node is in promiscuous mode – accepting all frames – forward packets to the other link

This is called a bridge!!

  - A single Ethernet segment can carry 10Mbps
  - An Ethernet LAN switch can carry n.10Mbps
  - n … number of ports (in and out)

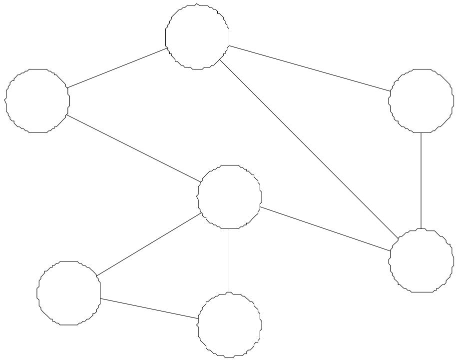
Bridges and LAN Switches

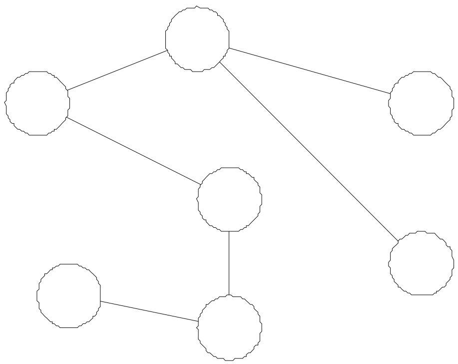
- Learning Bridges:
  - A bridge need not forward all receiving frames!

  How does a bridge can learn on which port the various hosts reside?
  - Human downloaded table (datagram or connection less model is used) – but this is quite a burden
  - But there is a simple trick:
    - Remember the MAC addresses of each senders
    - Build a table and remember the sending addresses
    - Discard entries after a specific period of time
      - Sender has moved

? Bridges and LAN Switches

- Spanning tree algorithm:
  - Preceding strategy works fine – until there is a loop in it (extended LAN with more than 1 bridge)
  - An extended LAN can be represented by a graph
    - … with loops in it (due to redundant links)
  - A spanning tree is a sub-graph - keeping all the vertices – but contains no cycles!!!





Bridges and LAN Switches

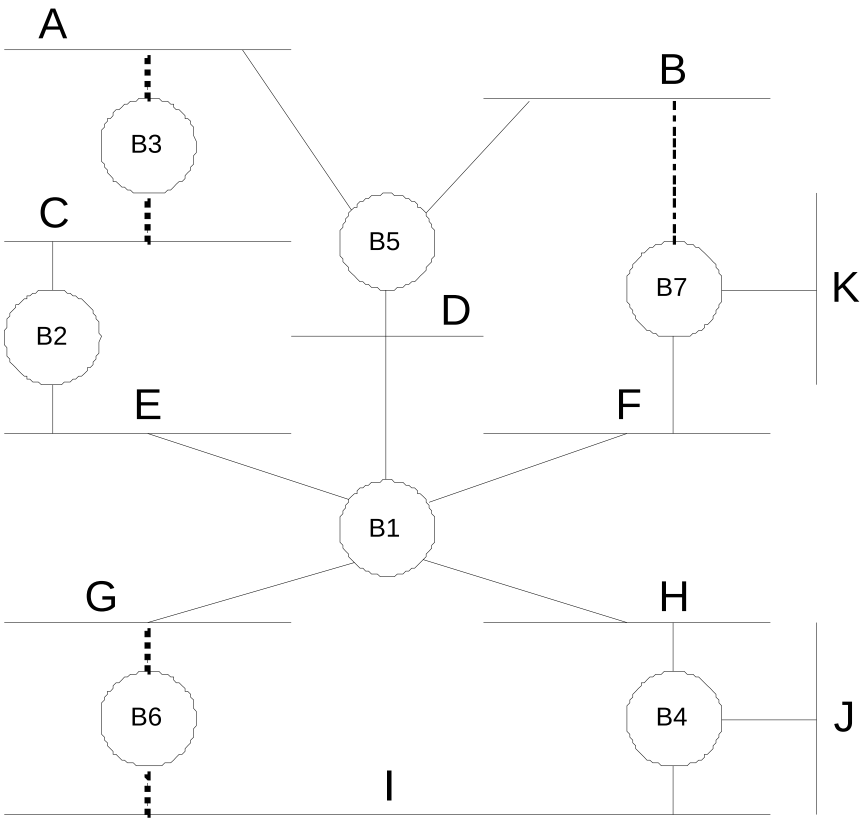
- "Spanning trees" are used by a set of bridges
- IEEE 802.1 specification for LAN bridges is based on this algorithm
- Each bridge decides the ports over which it is and is not willing to forward frames
- Even an entire bridge will not participate …
- … but the algorithm is dynamic – "reconfiguration"
- Algorithm:
  - Each bridge has a unique identifier Bi (i= 1…n)
  - Select the bridge with the smallest ID as the root
  - Each bridge computes the shortest path to the root (and notes which nodes are on this path)

Bridges and LAN Switches

- All the bridges elect a single "designated bridge" that will be responsible to forward packets to the root
- Each designated bridge is the closest bridge to the root
- If more are equal close, the smaller ID wins
- Each bridge is connected to more than 1 LAN
  - Bridge has to elect for each LAN
- This means, that each bridge decides if it is the designated bridge relative to each of its ports
- The bridge forwards frames over those ports for which it is the designates bridge.

Bridges and LAN Switches

- B1 is root
- B3 and B5 are connected to A. B5 is designated bridge, since closer to the root
- B5 and B7 are connected to B. B5 is designated bridge, since lower ID (but equal distance from root)



Bridges and LAN Switches

- Bridges cannot see the network topology, thus …
- Bridges have to exchange configuration messages
- And decide, whether or not they are the root or a designated bridge.
- Configuration-messages contains :
  - ID of the sending bridge
  - ID for what the sending bridge believes to be the root
  - Distance measured in hops from sender to root
- Each bridge records the current "best" configuration message (it has seen) on each of its port
- Initially, each bridge thinks it is the root and sends a configuration message out on each port

Bridges and LAN Switches

- … identifying itself as the root – and giving a distance to root as 0
- Received messages are checked, if they are better than the current best configuration message recorded for that port.
- the new configuration is considered better, if:
  - if it identifies a root with smaller ID or
  - if it identifies a root with an equal ID but with a shorter distance or …
  - the root ID and distance are equal, but the sender bridge has a smaller ID

Bridges and LAN Switches29.03

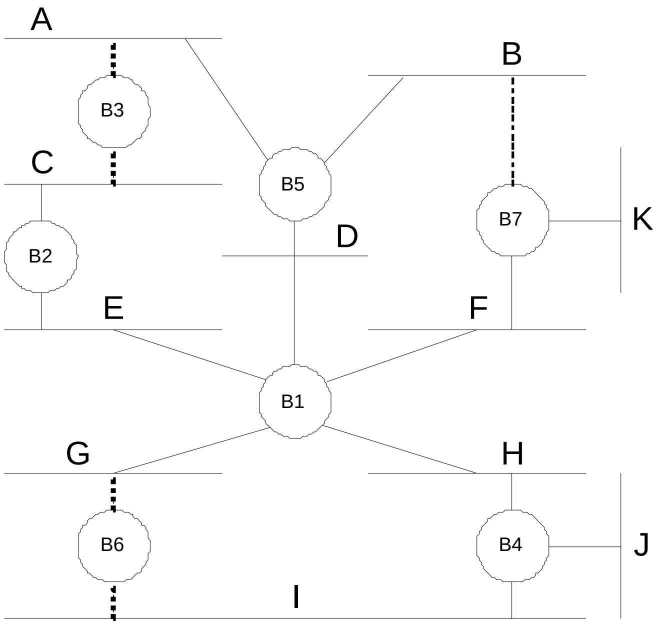
- If one the new message is “better” – discard the old information – save the new
- Add 1 to the ”distance to root” field
- If a bridge receives a configuration message, that it is not the root (message from a bridge with smaller ID) – stop generating configuration messages
- If a bridge receives a configuration message, that it is not a designated bridge for that port (message from a bridge that is closer to the root or equal fare but with a smaller ID) – stop sending configuration messages
- When the system stabilizes: just the root is still generating configuration messages

? Bridges and LAN Switches

– All bridges starts by claiming to be the root

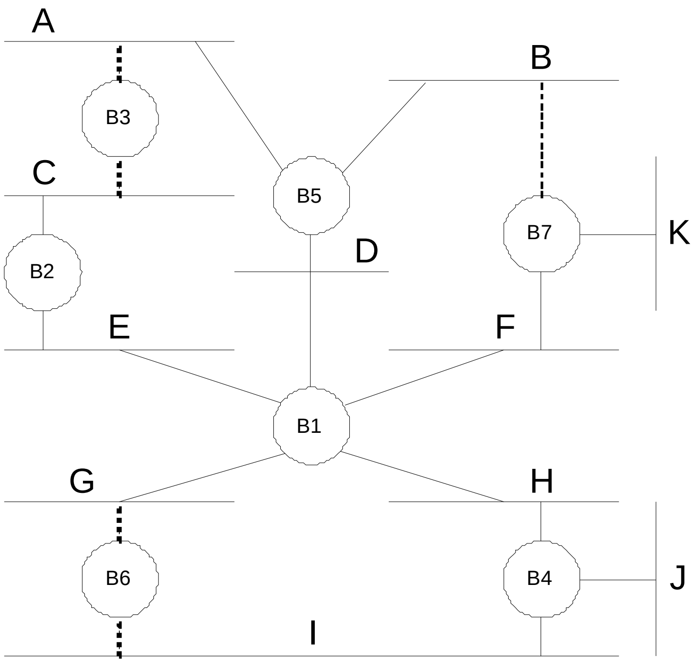
– Configuration message from “X” in which it claims to be distance “d” from root node “Y” (Y, d, X)

3. B3 receives (B2, 0, B2)
4. Since 2<3, B3 excepts B2 as root
5. B3 adds 1 to the distance and sends (B2, 1, B3) to B5
6. Meanwhile, B2 accepts B1 as root (lower ID) and sends (B1, 1, B2) to B3



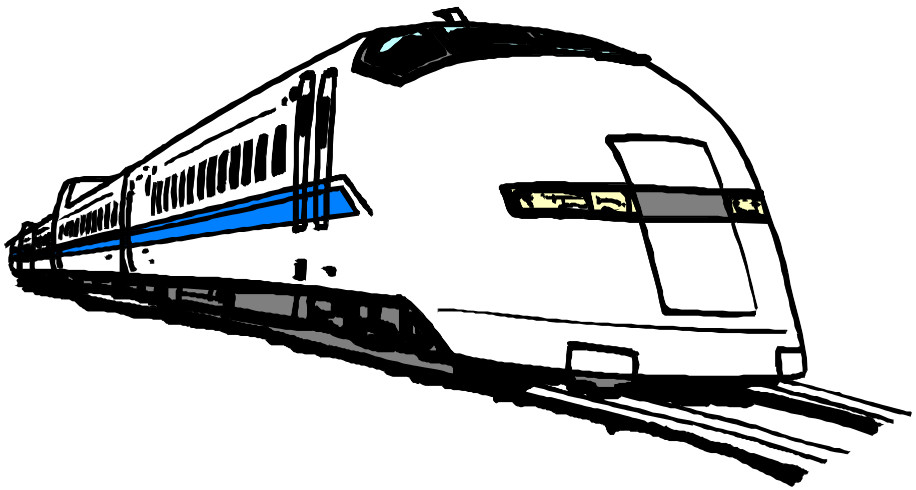
Bridges and LAN Switches

1. B5 accepts B1 as root and sends (B1, 1, B5) to B3
2. B3 accepts B1 as root, and it nodes, that both B2 and B5 are closer to the root than it is. Thus B3 stops forwarding messages on both its interfaces.



Frame Relay

- Broadband Network
  - Frame Relay
  - Asynchronous Transfer Mode (ATM)



Broadband-Motivation

- High throughput
- highly efficient even for “bursty” traffic
- Guarantee *Quality of Services*
  - throughput
    - average rate
    - *peak*-rate
  - delay
    - absolute delay
    - variation of delay
  - loss rate

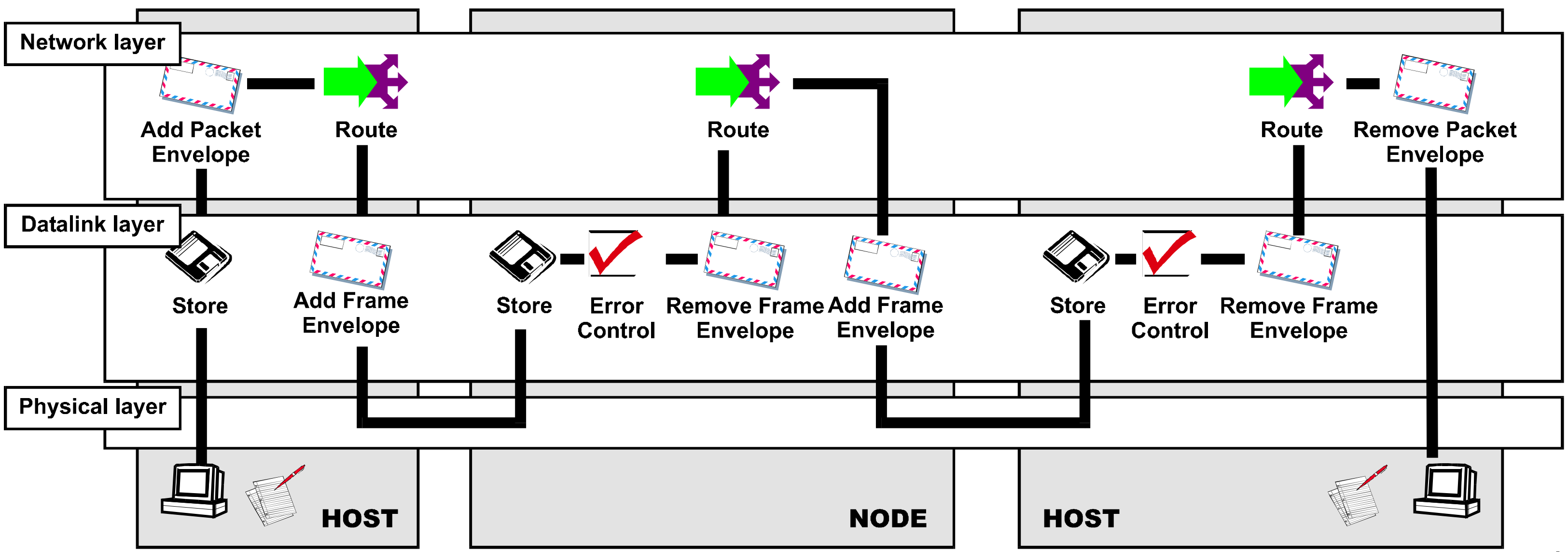
Broadband Approaches

- *Frame relay; Frame mode bearer service*
  - up to E3/T3
- *Distributed queue dual bus* (DQDB)
  - up to E4/T4 (OC-3) {planed for: OC-12}
- *Switched multi-megabit data service* (SMDS)
  - up to E3/T3
- *Asynchronous transfer mode (ATM)*
  - up to multi-Gbit/s

„Classical“ X.25 (historical)

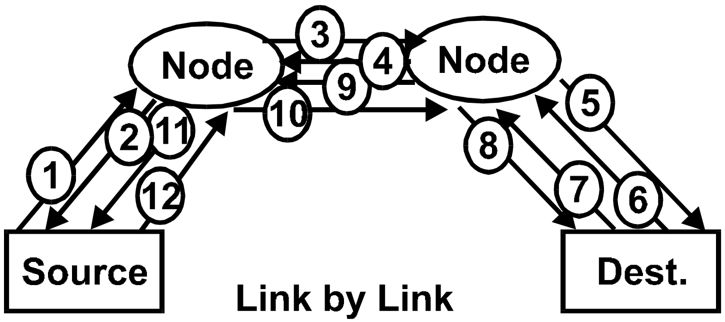
?

- *Inband signaling*
- VC Multiplexing on the *network layer*
- flow-/error control on *Layer-2*

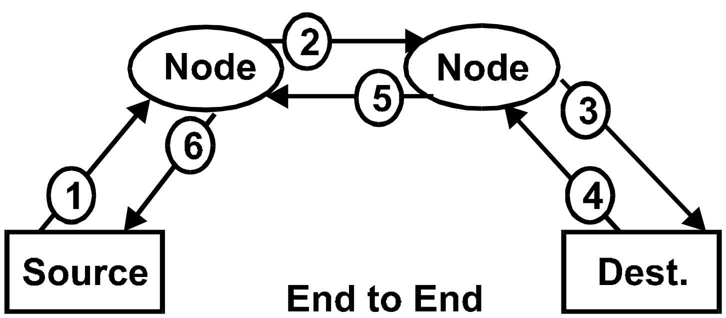


Store & Forward, Error control

- X.25 assumptions: relevant bit error rate
  - link-by-link error control
- Broadband: low error rate (i.e. <10^{-9})
  - typical: End-to-end error control
  - typical: network rejects at congestion/error



Link by Link
error control



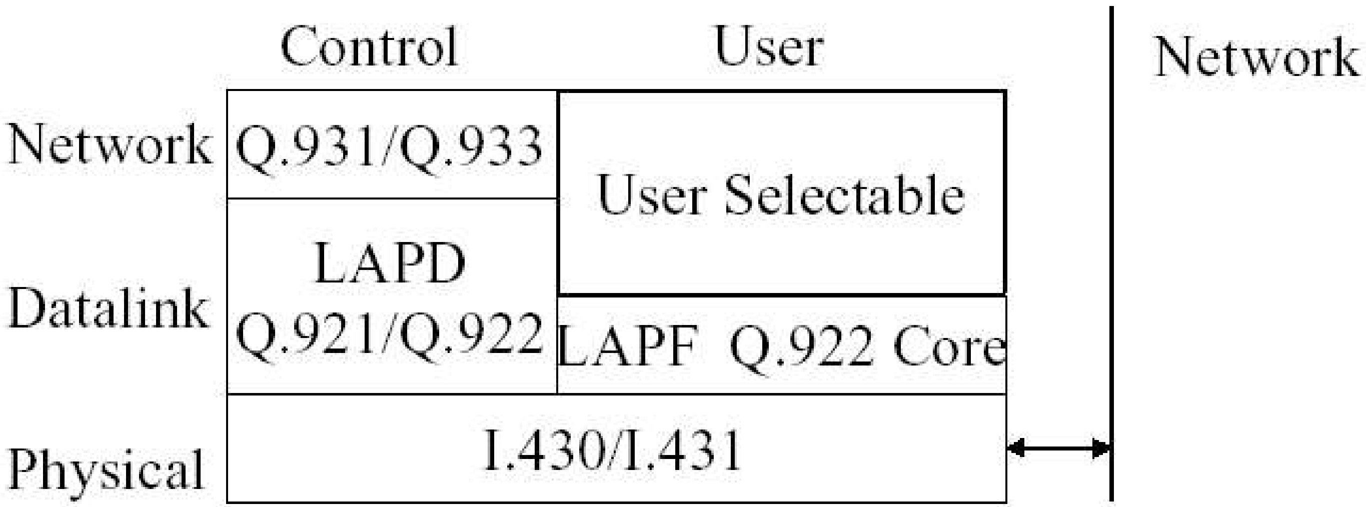
End to End
error control

Relay vs. Switching

- **Switching** (X.25) = Relaying + Ack + Flow control + error recovery + loss recovery
- Switching = X.25
- **Relay** (FR) = Unreliable multiplexing service
- DLCI (*data link connection identifier*): Similar to Logical Channel Numbers in X.25
  - Only local significance
  - Allows multiple logical connections over one circuit
  - DLCI = 0 is used for signalling

User vs. Control Plane

- UNI = User-Network Interface
- LAPF = Link Access Protocol - Frame Mode Services
- LAPD = Link Access Protocol - D Channel



Control Plane

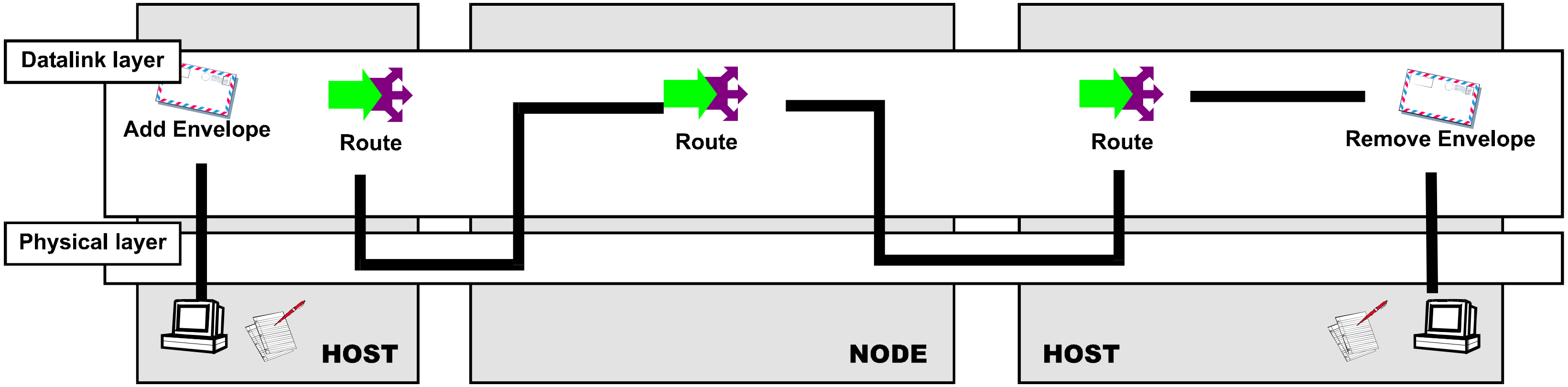
- Signalling over D channel (D = Delta = Signalling)
- Data transfer over B, D, or H (B = Bearer)
- LAPD used for reliable signalling
- ISDN Signalling Q.933 + Q.931 used for signalling messages
  - Service Access Point Identifier (SAPI)
  - SAPI = 0 in LAPD $\rightarrow$ Q.933 + Q.931 Frame relay message

User Plane

- Link Access Procedure for Frame-Mode bearer services (LAPF)
- Q.922 = Enhanced LAPD (Q.921) = LAPD + Congestion
- LAPF defined in Q.922
- Core functions defined in Q.922 appendix:
  - *Frame delimiting*, *alignment*, and flag transparency
  - *Virtual circuit multiplexing* and *de-multiplexing*
  - Octet alignment → Integer number of octets before zero-bit insertion
  - Checking *min and max frame sizes*, *error detection*, *sequence-, congestion control*.

? Frame Relay - FMBS

- reduced to the *datalink layer*
- reduction of the X.25 functionality
- ***Frame:*** **Packet of variable lengh**

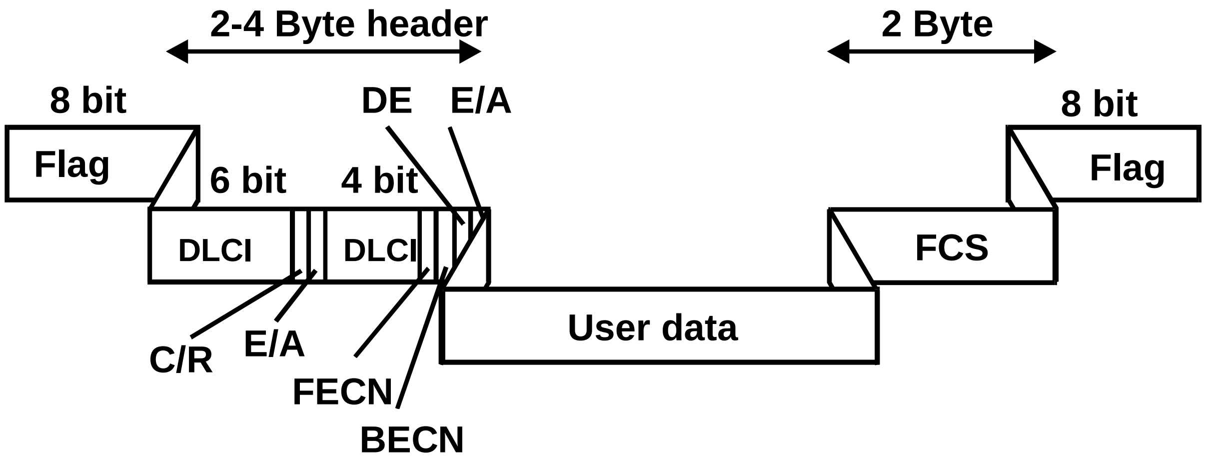


Frame Relay characteristics

- connection oriented
  - SVC: *Switched virtual circuit*, connection establishment!
  - PVC: *Permanent virtual circuit*
- Frame Relay QoS
  - CIR: *Committed information rate*
  - *Burst rate*
- *B-channel, H-channel, D-channel*

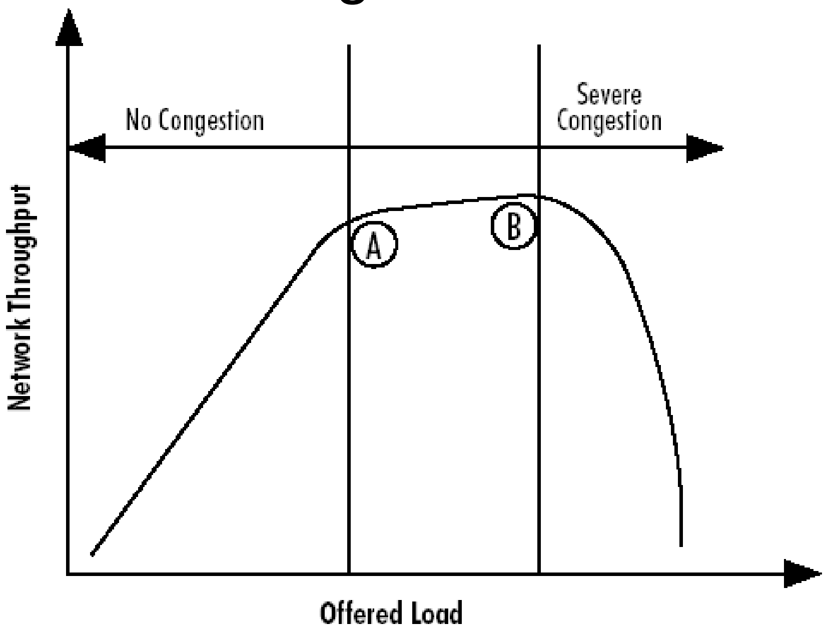
Frame Format

- Flag: frame delimiter
- DLCI: *Data link connection identifier*
- FECN, BECN: *Explicit congestion notification*
- C/R: *Command response;* E/A: *Extended address*
- DE: *Discard eligibility;* FCS: *Frame check sequence*



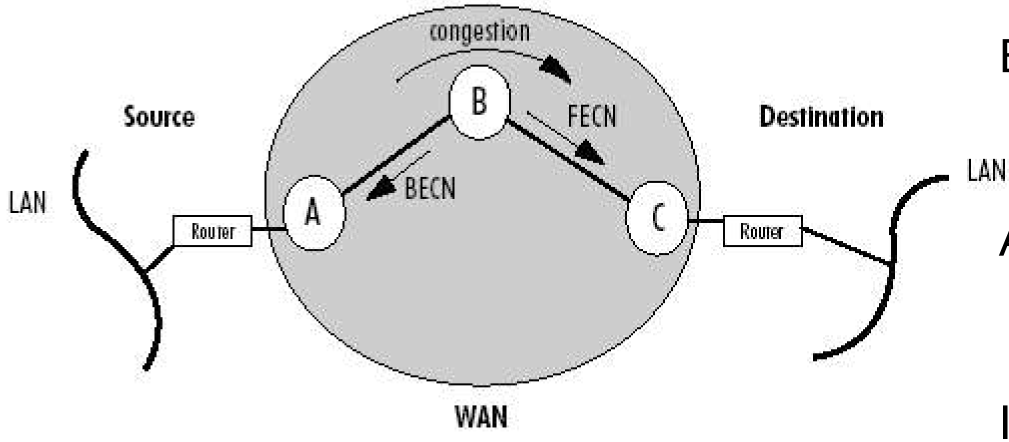
Congestion Notification Mechanisms

- As the offered load increases, the actual network throughput increases linearly.
- … the only way to recover is for the user devices to reduce their traffic.
- … several mechanisms have been developed to notify the user devices that congestion is occurring
- The network should be able to detect when it is approaching congestion (Point A) rather than waiting until Point B is reached
  - Explicit Congestion Notification
  - Discard Eligibility



Congestion Notification Mechanisms

- <u>Explicit Congestion Notification (ECN) Bits</u>



Node B is approaching a congestion (buffer usage or queue length).

B would signal Node C of the congestion within a frames destined for C with FECN = 1

All interim downstream nodes, learn that congestion is occurring on the DLCI (s)

It is sometimes more useful to notify the source of the traffic that there is congestion. This is called Backward Congestion notification BECN = 1.

Congestion Notification Mechanisms

- Discard Eligibility
  - A DE bit is set to one by the CPE*) device or the network switch when the frame is above the CIR **)
  - … this makes the frame eligible for discard in response to situations of congestion.
  - A frame with a DE bit of 1 is discarded in advance of non-discard-eligible data

*) customer premises equipment

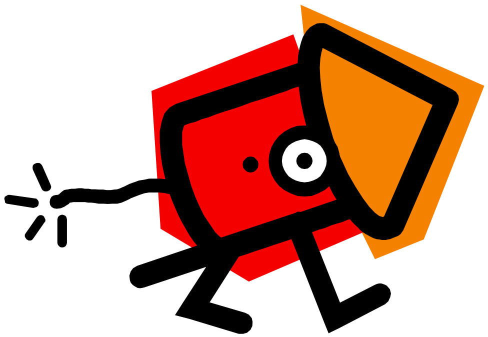
**) committed information rate

Frame Relay Summary

- standardized by ITU-T / ANSI
- based on ISDN, X.25
- using the Minimal-Set of X.25
- connection oriented (SVC, PVC)
- for small (E1) up to (E3) data rates
- efficient for *bursty-LAN* traffic
- moderate usable for real-time application (voice)

?

3.3 Asynchronous Transfer Mode



- Basics
  - semantic/time transparency
- Protocol Model
  - PHY vs. ATM vs. AAL
- PC as a ATM DTE
  - Classical IP, LANE
- TCP/IP throughput via ATM

ATM Characteristics

- cell based
  - cell = packet with fix length
- „tiny“ cell
  - 5 Byte Header, 48 Byte *Payload*
- Connection oriented
  - virtual path; virtual channel
- Limited *header*-functionality
- No error detection/-correction and flow control!

Common System Requirements

- Semantic transparency
  - „correct delivery“
  - keep in mind: **no error correction** in broadband networks (ATM)
- Time transparency
  - „delivery in time“

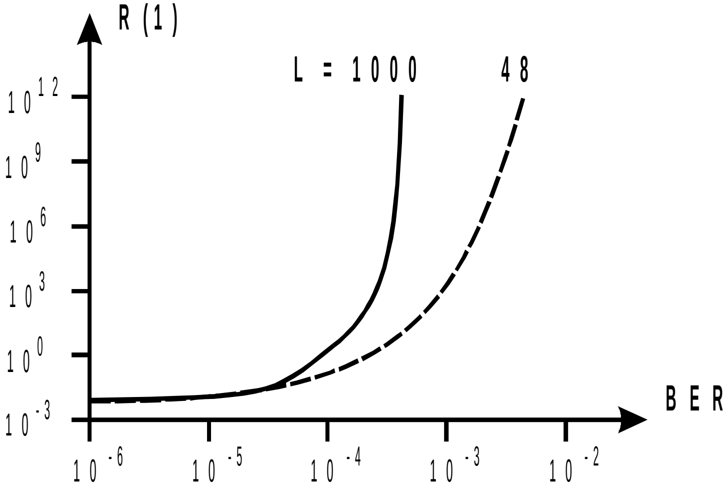
Semantic Transparency Terms:

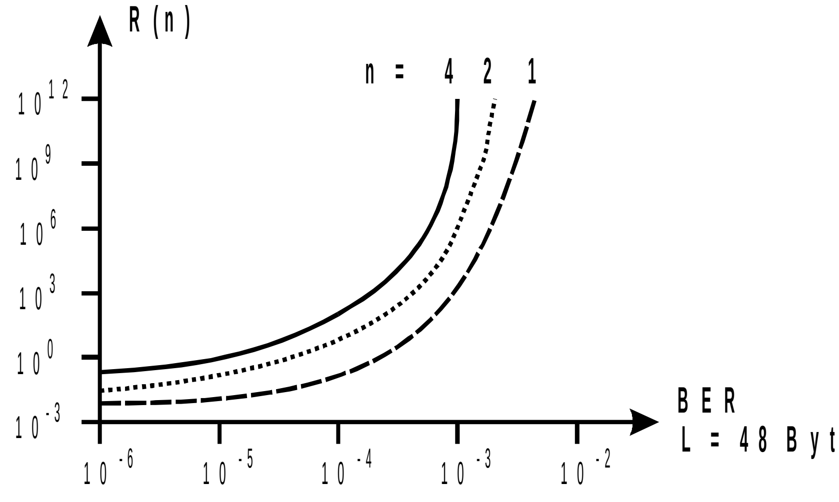
- *Bit error rate* BER
- *Packet error rate* PER
- *Packet loss rate* PLR
- *Packet insertion rate* PIR

- data field error
- misrouting of cells (*header* error)
- buffer overrun

Semantic Transparency
increasing load due to errors

- Example: increasing load **R(n)** due to **BER**
  - **n** nodes at packet length **L** and window size **W**
  - $\mathbf{R(n) = (W/2) * (1 – (1\text{-}BER)^{nL}) / (1\text{-}BER)^{nL}}$





Time Transparency

- ATM service characteristics
  (according to RACE 1022, minimal requirements)

service	*BER*	*PLR*	*PIR*	*delay*
Telephone	10^{-7}	10^{-3}	10^{-3}	25/500 ms
Data transfer	10^{-7}	10^{-6}	10^{-6}	1000/50 ms
Video	10^{-6}	10^{-8}	10^{-8}	1000 ms
HiFi	10^{-5}	10^{-7}	10^{-7}	1000 ms
Remote control	10^{-5}	10^{-3}	10^{-3}	1000ms

Time Transparency

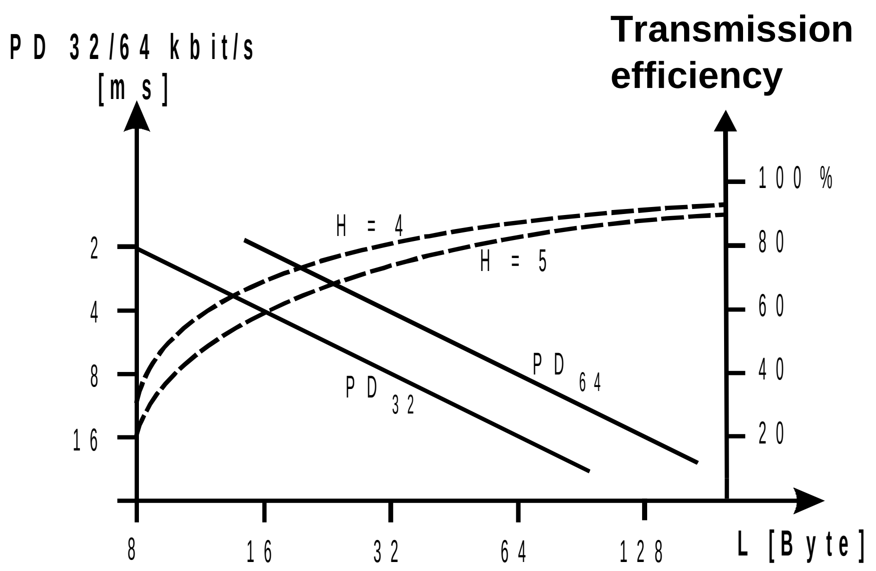
- *Transmission delay* TD
  - Depending on the distance (ATM-independent)
- *Packetization delay* PD
  - e.g. 32 kbit/s (4B/ms) voice, 48 Byte cell payload => **12 ms** (64kbps $\rightarrow$ **6ms**)
- Switching delay
  - *Fixed switching delay* FD
  - *Queuing delay* QD
- *De-packetization delay* DD

Time Transparency and Packet Size

- transmission efficiency $\eta_H = L / (L+H)$

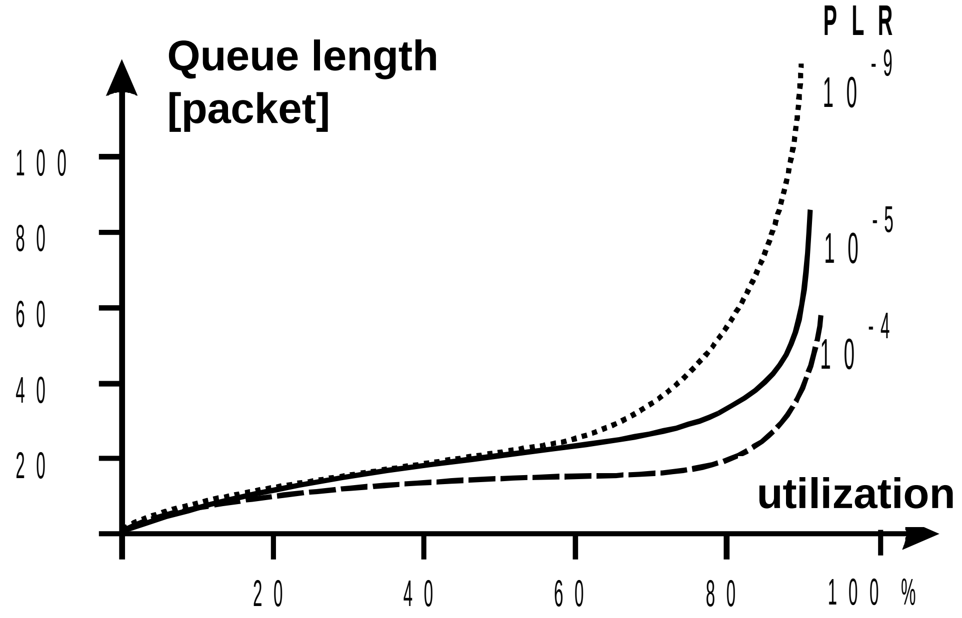
  (L … payload, H… *header* length)

- Example: PD and η_H vs. payload length



Time Transparency, Queuing Delay

– Queue length vs. utilization (50 consecutive queues)



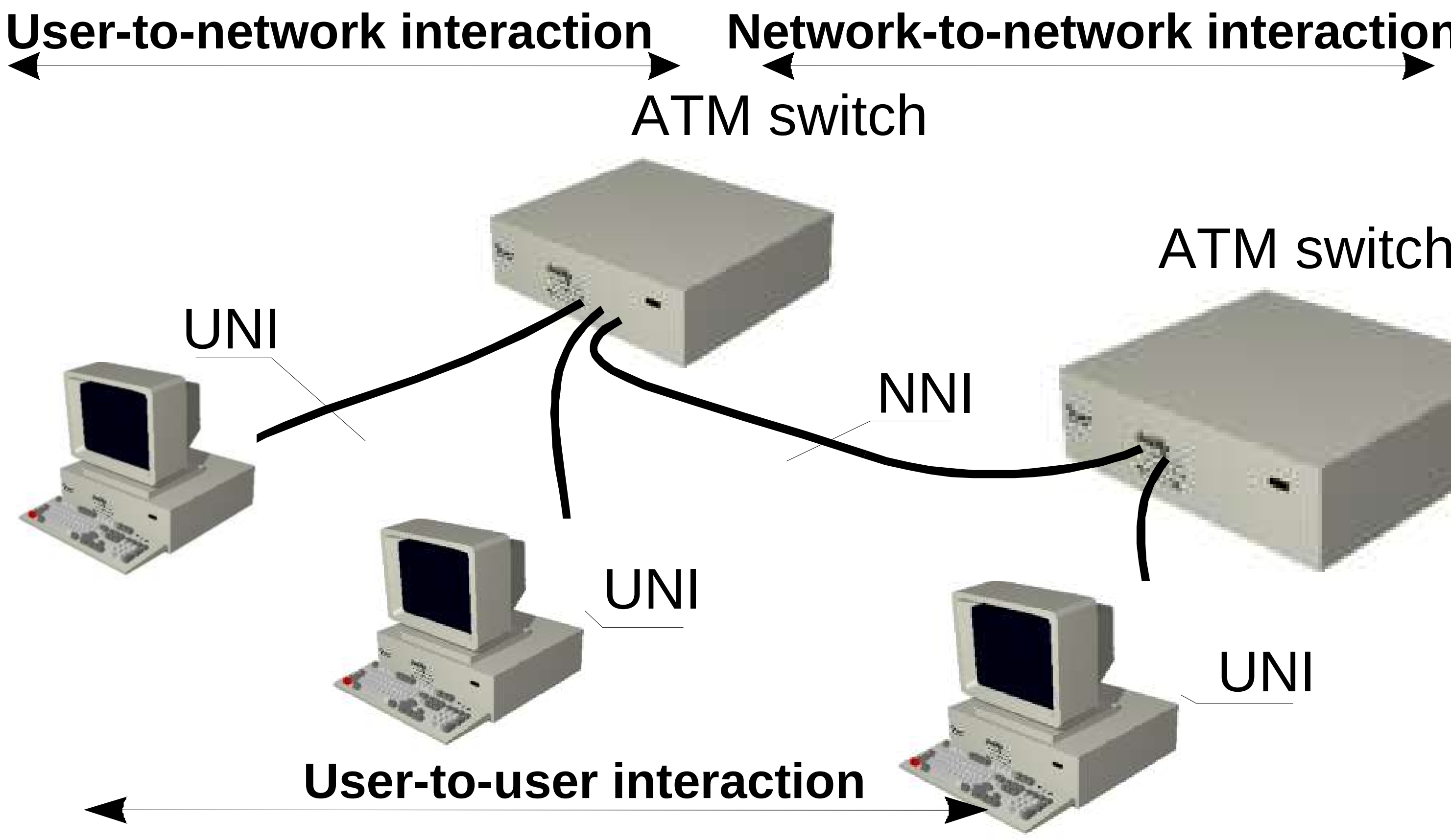
Time Transparency
Example: voice

- Example: 64 kbit/s voice via 155/600 Mbit/s at different packet length, 250 km, [μs]

packet size	*16*	*32*	*64*	*64*
Link rate [Mbit/s]		*155*		*600* .
TD	1000	1000	1000	1000
FD	64	128	256	256
QD/DD	200	400	800	200
PD	2000	4000	8000	8000
total	**3264**	**5528**	**10056**	**9456**

ATM Networks

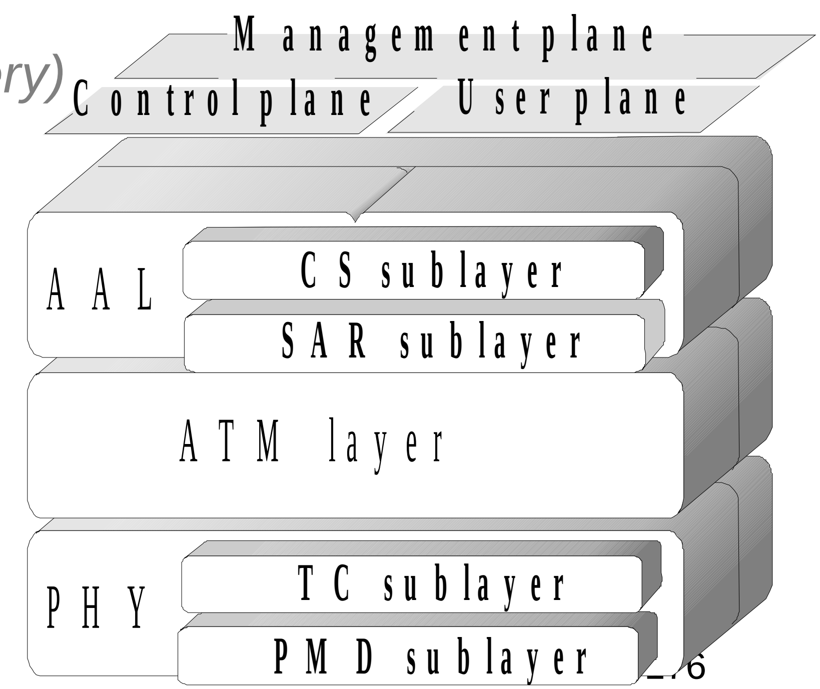
- UNI: *User-network interface;*
  P-NNI: *Private network-network interface*



?

ATM/B-ISDN Reference model

- Breakup into: *control plane, user plane, management plane*
- *ATM adaptation layer* AAL *(independent of physical layer, supports the adaptation of higher layers)*
  - *Convergence Sublayer* CS *(message identification, clock recovery)*
  - *Segmentation & Reassembly* SAR *(segmentation of higher layer data)*
- *ATM layer*
- *Physical layer*
  - *Transmission convergence* (TC)
  - *Physical media dependent* (PMD)



Functionality of ATM Layers

AAL (ATM Adaptation Layer)

- Convergence Sublayer, SAR *(page 56)*

ATM Layer

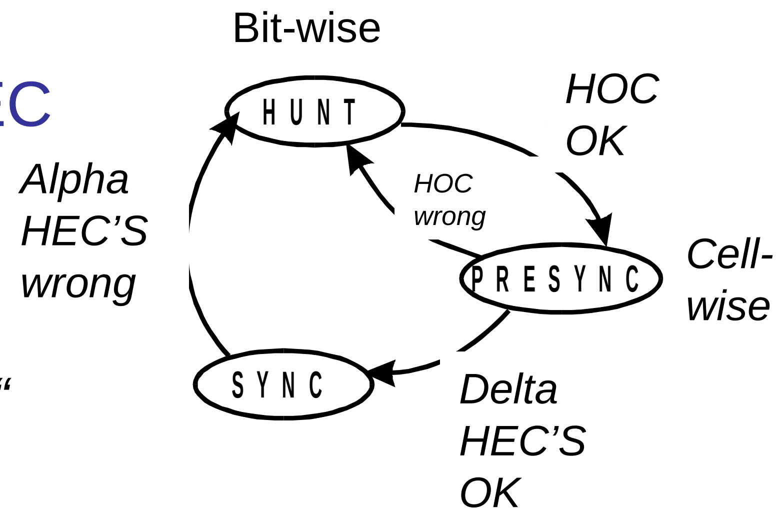
- Common flow control (*generic flow control* GFC)
- Cell switching (based on VPI/VCI)
- Cell multiplexing, de-multiplexing *(different connections on a single cell stream)*
- QoS *(based on CLP bit)*
- OAM *(Operation, Administration, and Maintenance is organized in layers: F5→VC(ATM), F4 →VP(ATM), F3 →Physical Layer)*

Physical layer

- TC: decoupling of cell rates, HEC generation/verification adapting transmission frames, cell limit recognition build transmission frames.
- PMD: Bit-*timing,* Physical media

Physical layer: TC 05.05

- ***Transmission convergence***
  - Cell rate decoupling, by adding idle cells: Sender/receiver are using different cell-cycles (🡯 **„asynchronous“ in ATM)**
  - Generate/verify *header error checksum* HEC
  - Cell limit recognition
    - *Via cell delimiter*
    - *Via HEC-correlation, cells are „close to each other“*



- ***Physical media dependence***
  - Embedding into transmission frame (PMD)
  - Bit timing
  - Physical media (optical, electrical)

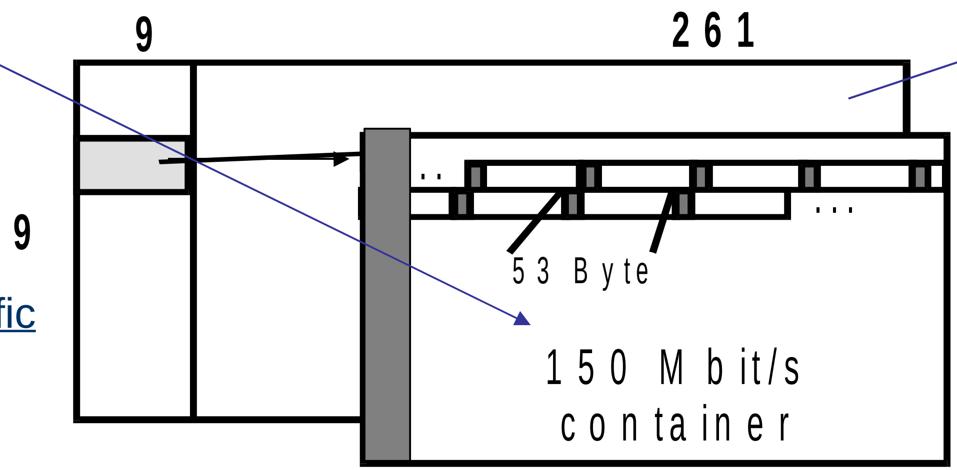
Physical layer (Examples)

(261-1)/(261+9) x 155,52Mbps = 149, 76Mbps

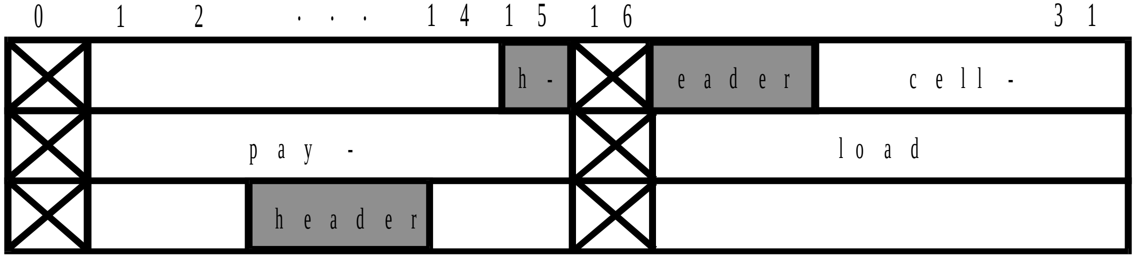
… moreover:
260 x 9 n x 53

… therefore:
CCITT recommendation:
VC-4 frames for ATM trafic

STM-1 resp. OC3c-frame



PDH E1-Rahmen



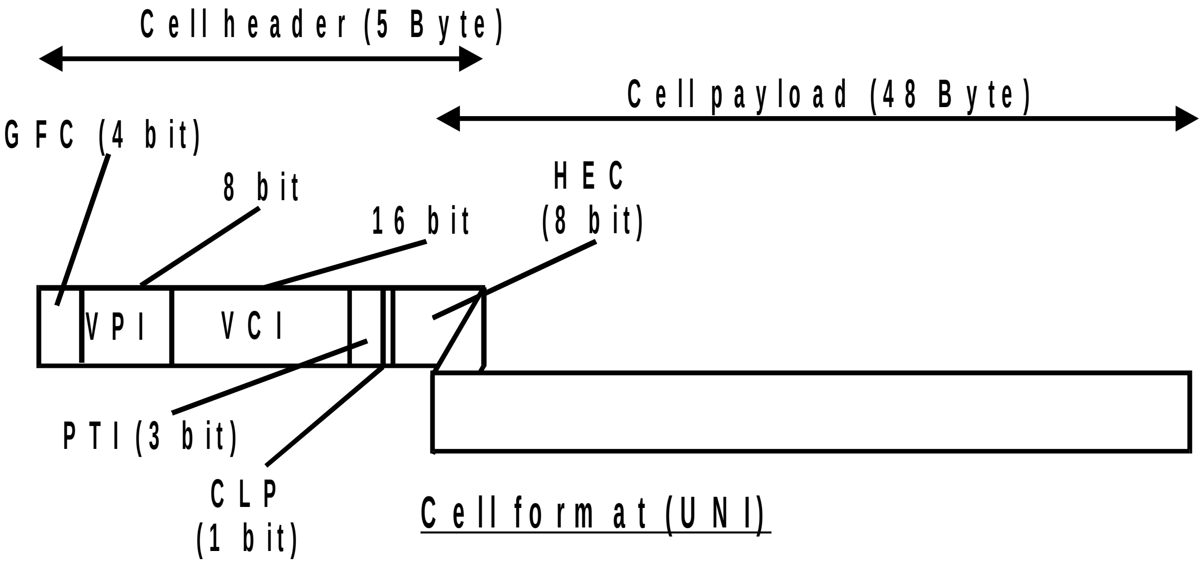
Physical Layer PMD

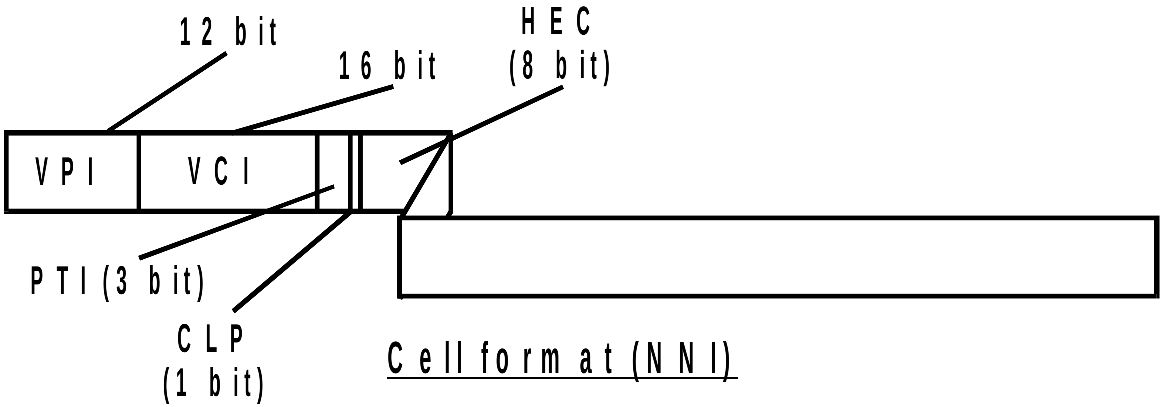
- ***Physical media dependent (Examples)***
  - n*56/n*64 kb/s
  - 1.5 / 2 Mb/s (T1/E1)
  - 6 / 8 Mb/s (T2/E2)
  - 25 Mb/s
  - 45/34 Mb/s (T3/E3)
  - 155 Mb/s (OC3c)
  - 625 Mb/s (OC12)
  - ...

ATM layer: ATM-Cell

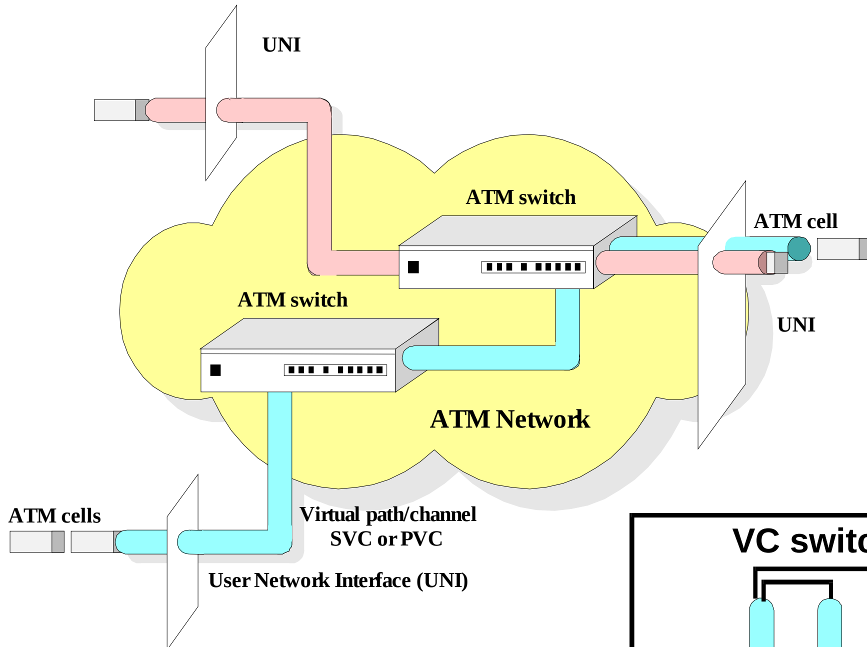
- GFC ***Generic flow control*** **4 bit (just UNI)**
- VPI ***Virtual path identifier,*** **8 bit (UNI), 12 bit (NNI)**
- VCI ***Virtual channel ident.*** **16 bit**
- PTI ***Payload type identifier*** **3 bit**
  - idle cell
  - OAM cell
    - administration of resources
  - user cell
- CLP ***Cell loss priority*** **1 bit**
  - similar to eligibility bit in frame relay
- HEC ***Header error checksum*** **8 bit**
  - verification of the cell header (misrouting)

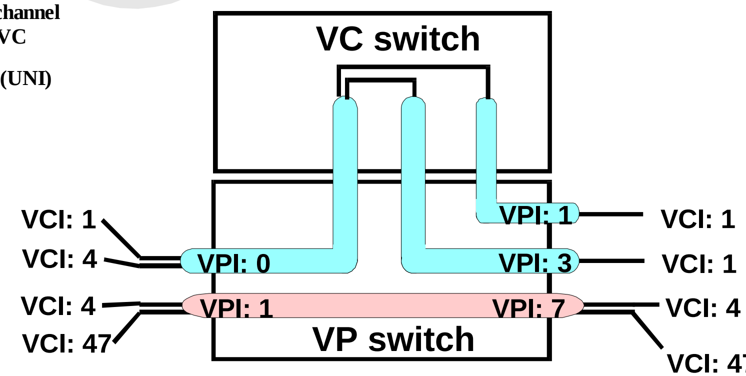
ATM Cell (II)


Cell header (5 Byte)
Cell payload (48 Byte)
GFC (4 bit)
8 bit
16 bit
HEC
(8 bit)
VPI
VCI
PTI (3 bit)
CLP
(1 bit)
Cell format (UNI)


12 bit
16 bit
HEC
(8 bit)
VPI
VCI
PTI (3 bit)
CLP
(1 bit)
Cell format (NNI)

ATM layer: VPC vs. VCC





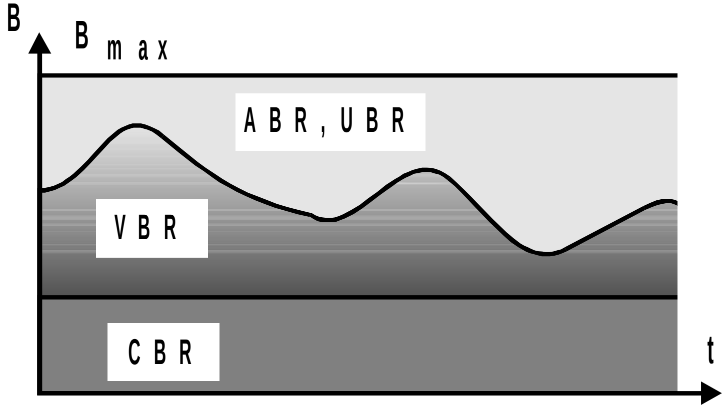
- SVC/PVC: ***Switched virtual circuit permanent virtual circuit***
- *Virtual path:* **„bundling“ of several *virtual channels***

? ATM Adaptation Layer AAL

- Functions and variants (service classes according to ITU-T)
  - Time relation between source and destination *(voice @ 64kbps, real time services)*
  - Bit rate: constant or variable
  - Connection mode: connection oriented / -less

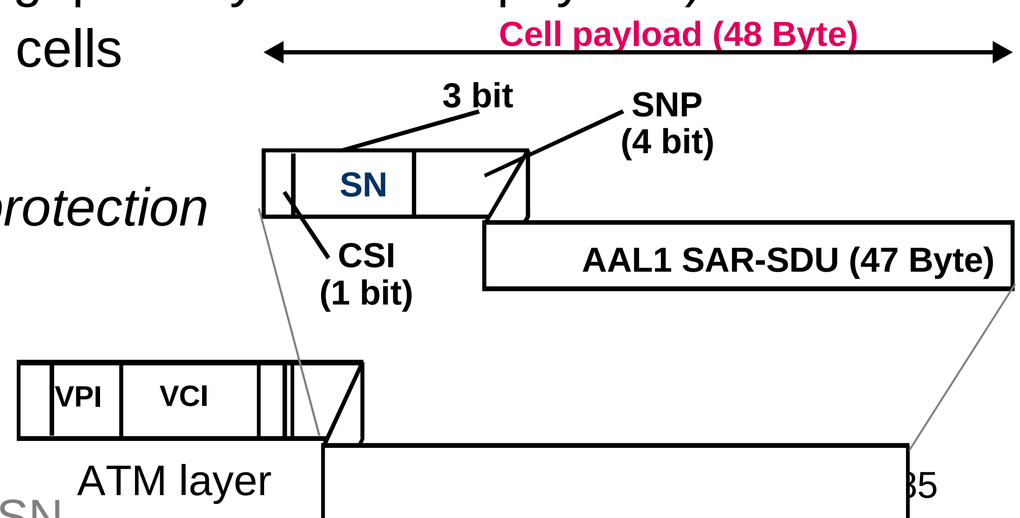
	class A	class B	class C	Class D
Timing between Source and destination	obligatory		without obligation	
Bit rate	constant	variable		
Connection mode	connection oriented			-less

- Service classes according to ATM-Forum
  - CBR *constant bit rate*
  - VBR *variable bit rate*
  - ABR *available bit rate*
  - UBR *unspecified bit rate*



AAL 1

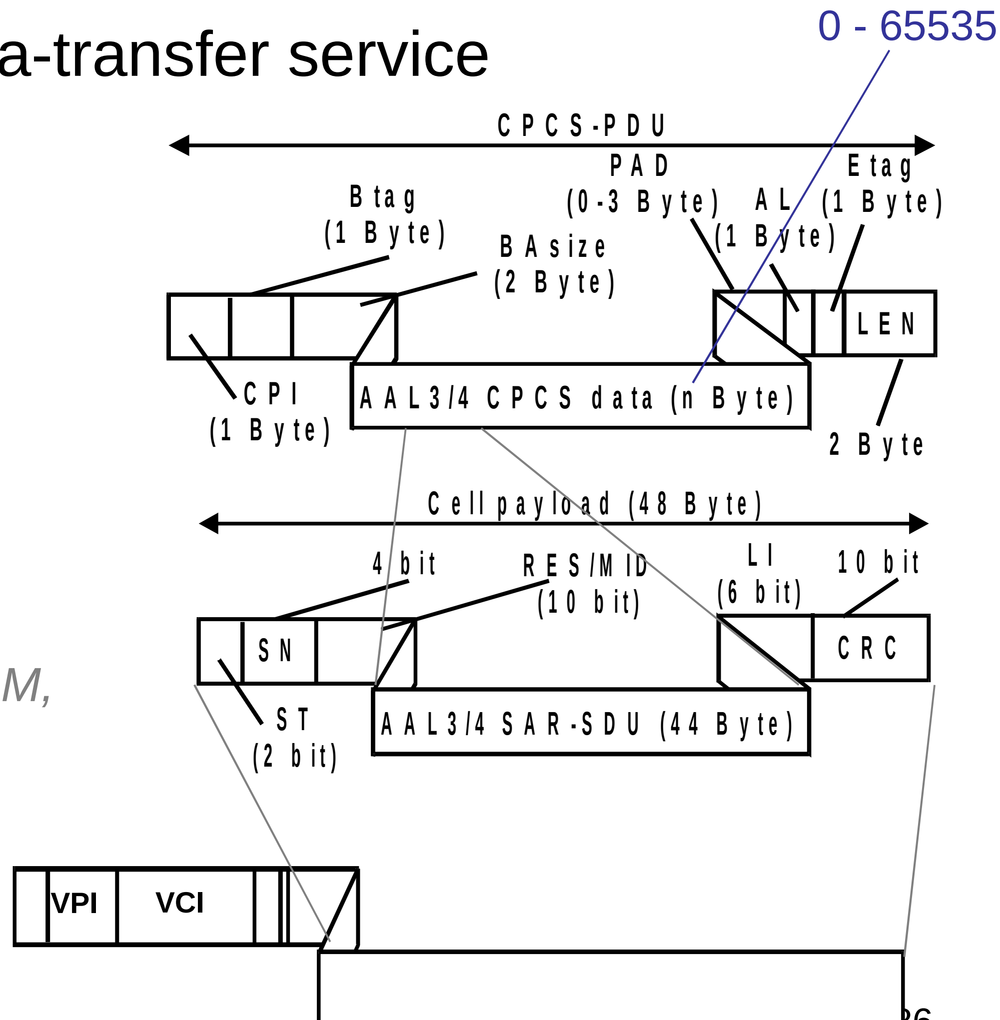
- to accommodate to constant bit rate
  - **Convergence function (CS)**
    - Handle cell delay (*cell delay variation*)
    - Compensation of source clock variation at the receiver
      - Synchronous residual time stamp *(diff. to reference clock)*
      - *Accommodation by using the fill levels of the buffers*
    - Payload reconstruction (e.g. partially-filled last payload)
    - Treating of lost/erroneous cells
  - **SAR**
    - SNP: *sequence number protection*
    - SN: *sequence number*
    - CSI: *CS indicator*



CRC-3: 3 bits; on SN
PC: parity check; 1 bit; even parity on SN

? AAL 3/4

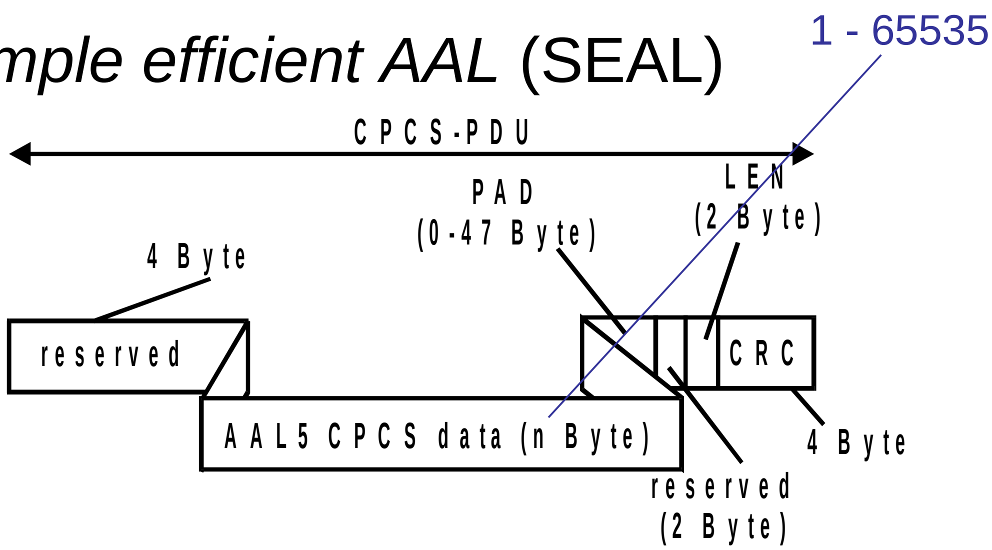
- to accommodate to the data-transfer service
  - **Convergence function**
    - CPI: *common part ID*
    - B/Etag: *begin/end tag*
    - BAsize: buffer allocation size
      - Message mode → LEN
      - Streaming mode > LEN
    - AL: Alignment (4 Byte *trailer*)
      - Alignment of the trailer
  - **SAR**
    - ST: *segment type (BOM, COM, EOM, SSM)*
    - SN: *sequence number*
    - RES/MID: *reserved (AAL3) multiplex ID (AAL4)*
    - LI: *length indicator*



?

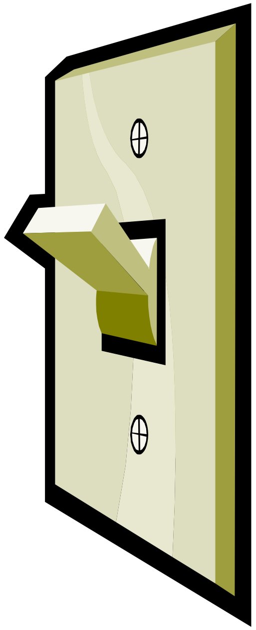
AAL 5

- Data-transfer service, *simple efficient AAL* (SEAL)
  - **Convergence function**
    - **Frame of variable length**



  - **SAR**
    - accepting each SAR-PDU with a length of 48 bytes from CS (complete *cell payload* is used)
    - *user-user* indication bit of PTI is used to indicate the end of CPCS-PDU (set to 1)

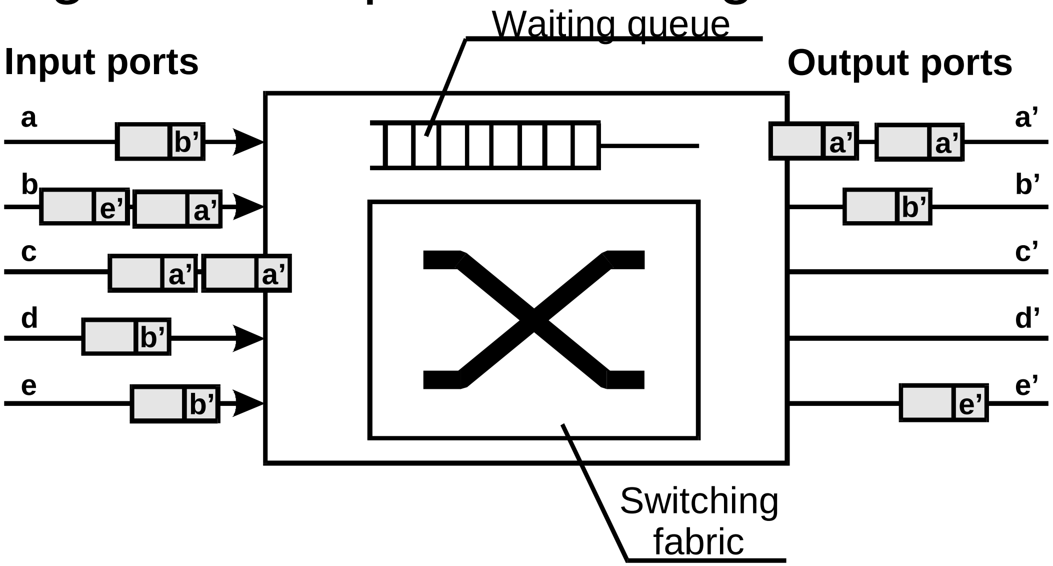
3.4 Switching Hardware



- Fabric vs. buffer
- Switch architectures
  - Batcher-Banyan ,
  - knock-out,
- QoS

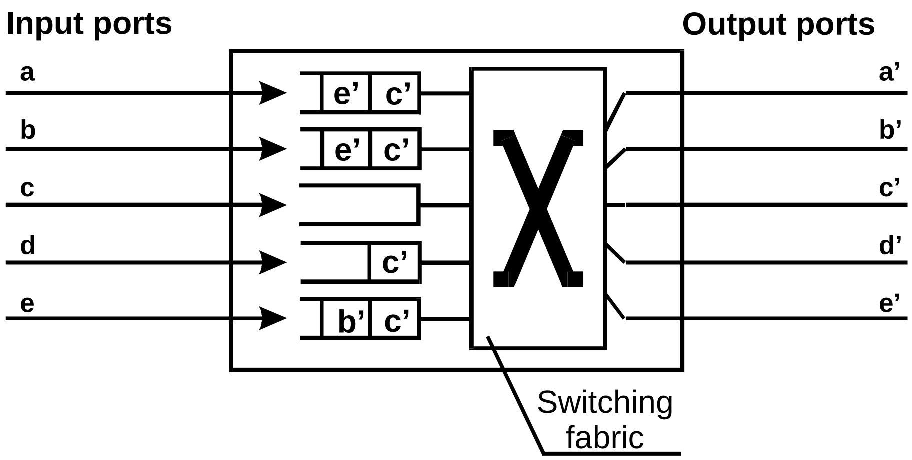
? ATM Switches

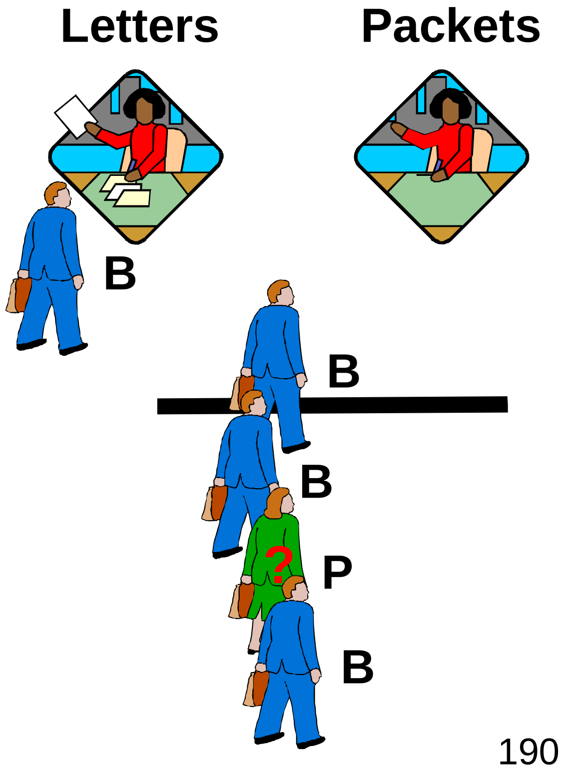
- there is a need for cell buffers on each ATM *switch*
- cell sequence has to be retained unchanged
  - waiting queue = FIFO
- *Switching fabric* $\rightarrow$ port switching



Input Queue

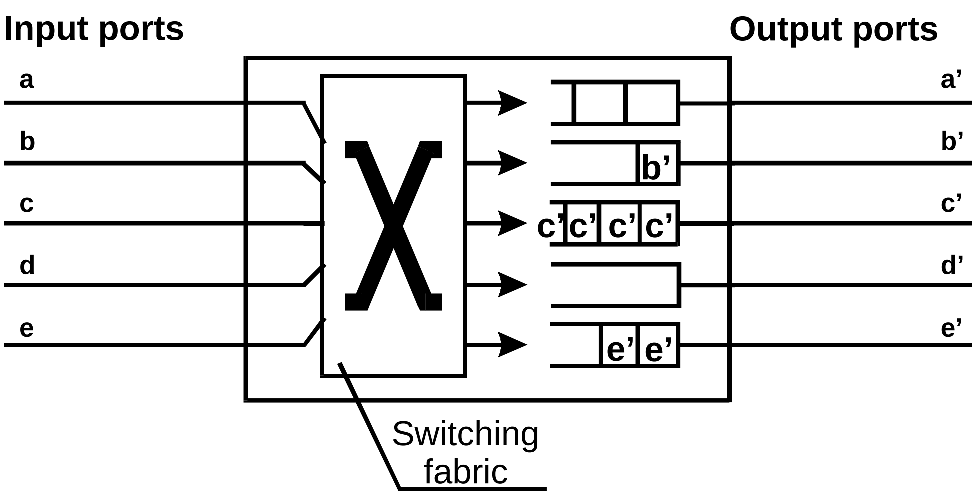
- Buffering cells at input ports
  - Problem:
    head of line blocking

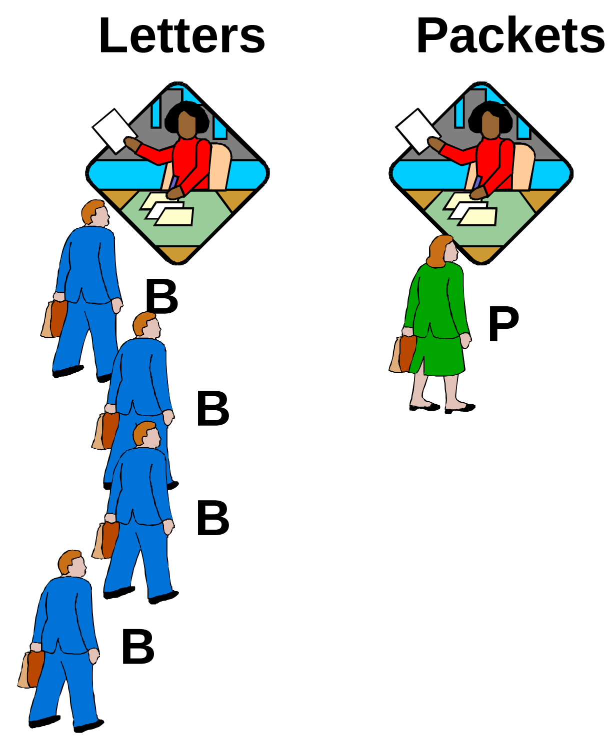




Output Queue

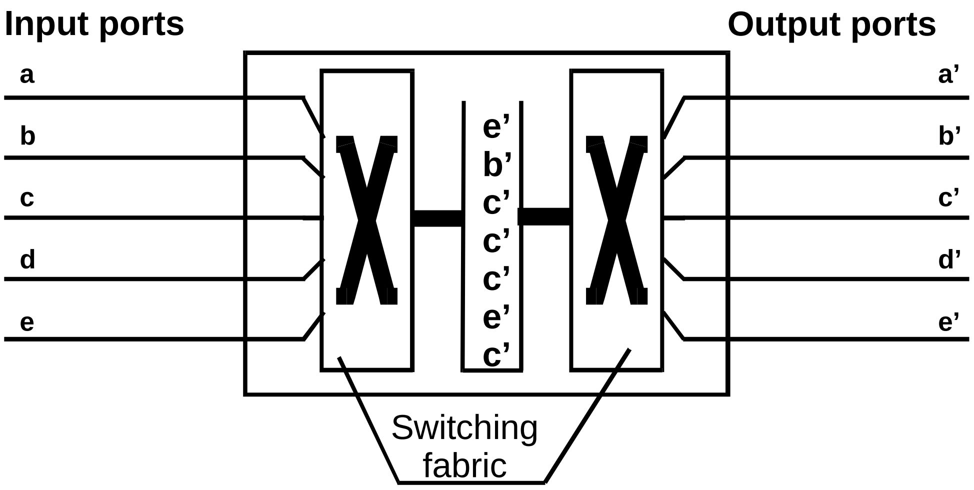
- Buffering cells at output ports
  – No „*head of line blocking*“

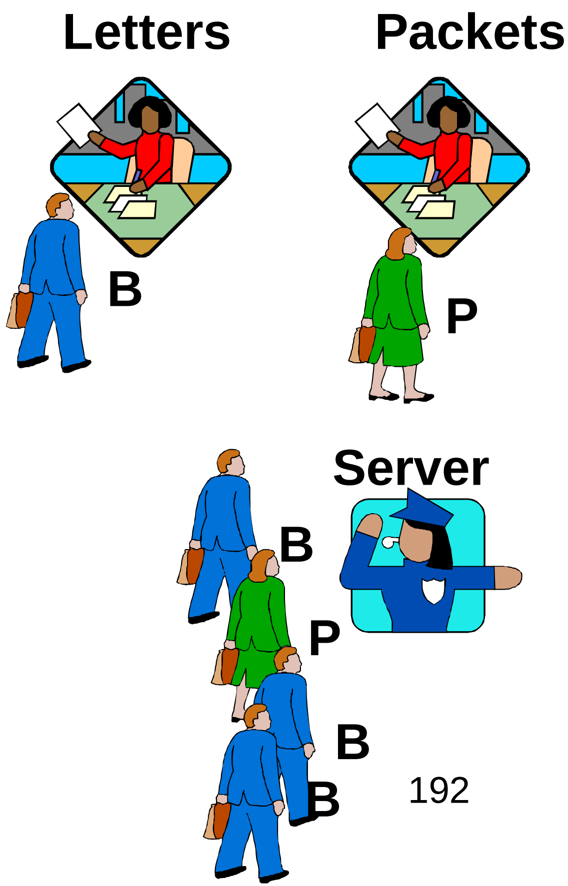




Central Queue

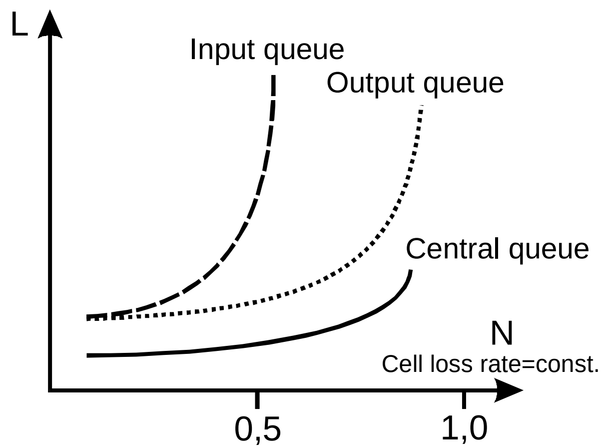
- Store all cells in a common queue
  - „*Server*“ is selecting a cell for further processing

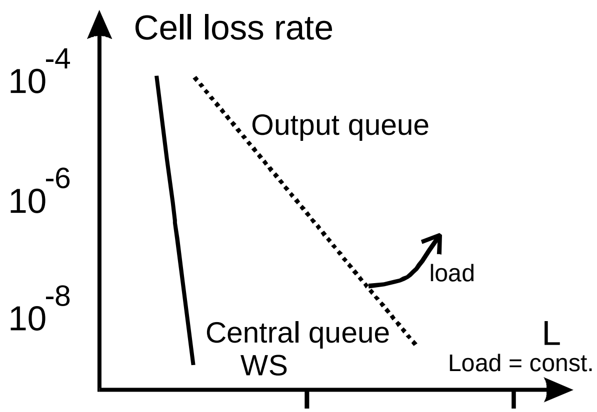




? Comparing Architectures

- *Queue*-**length (L) is an indicator for the delay, (load N)**





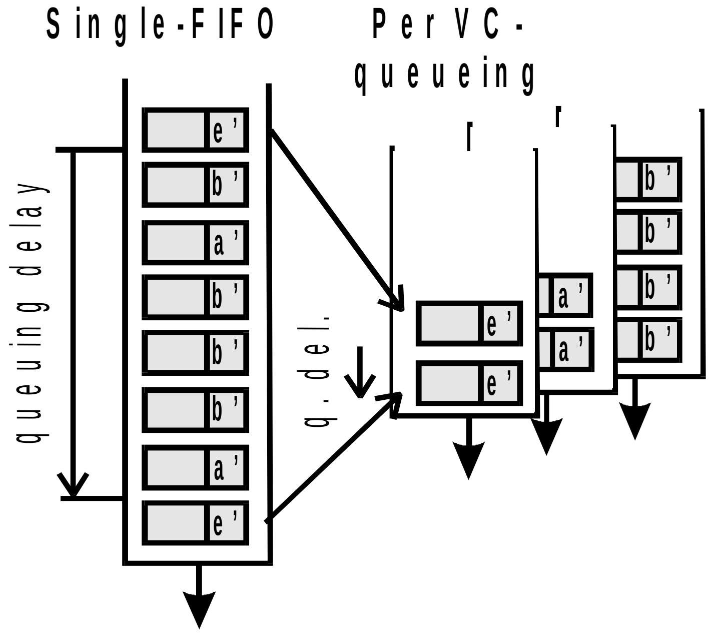
- **Switch-buffer memory access time**

W: memory width (16 Bit), **F**: throughput/port (150 Mbit/s), **N**: ports (16)

	Input-	Output-	Central Queue
single-ported memory	**W/(2*F)**	**W/((N+1)*F)**	**W/(2*N*F)**
example (ns)	53.3	6.3	3.3
dual-ported memory	**W/F**	**W/(N*F)**	**W/(N*F)**
example (ns)	106.6	6.7	6.7

ATM Switches and QoS

- CBR/VBR traffic is sensitive concerning **delay**
  - delay at higher loads is determined by queue fill levels.
- The challenge
  - *Per-VC queuing*
    - One queue per VC
    - Traffic with higher priority can be chosen first.

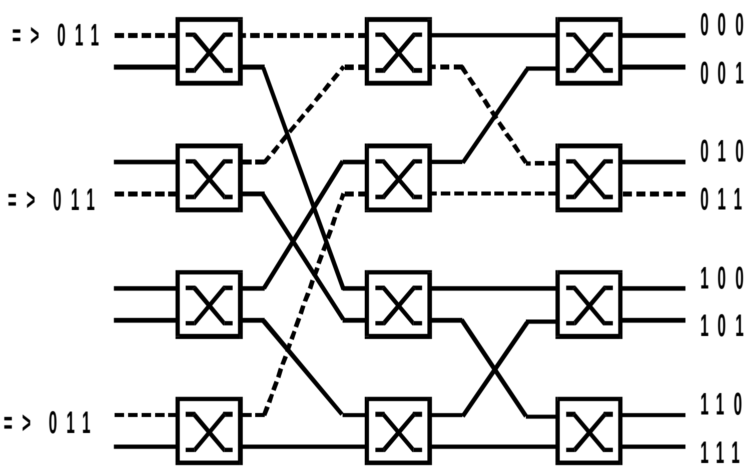


ATM-switch Examples

- *Batcher-Banyan* architecture
  - Based on *multistage interconnection network* (MIN) *(non-blocking switches 50th - first Batcher networks in 1972)*
  - Requires input-logic and input queues *(to prevent output blocking)*
- *Knock out* switching elements
  - Output queues
  - Allow just low, internal cell loss probability
- *Distributed shared memory* architecture
  - Central buffer

MINs, Banyan Networks

- **C. Clos (1950):** non-blocking MINs, circuit switching networks
- **Goke & Kipvski (1972):** Banyan network, Delta network
  - **Self-routing networks**
    - if 0-bit select left output port, else select right output port
  - **… if used as ATM-*switch* → problem with conflicts**



Example: output **010** not reachable, as long as there is a connection to **011**
=> Banyan ATM-*switch* is **internal blocking**

Banyan Network (solutions to solve output conflicts)

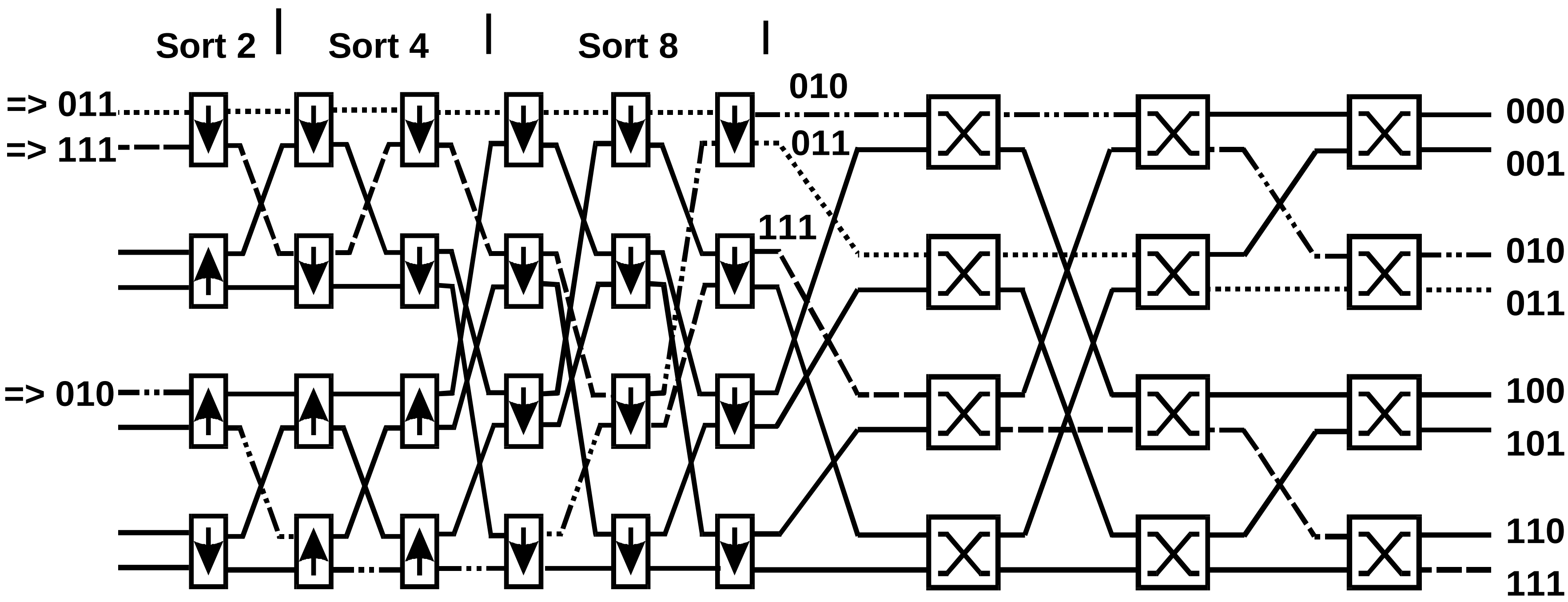
- buffer in each (2*2) basic module
- increase internal processing speed, to process more than one cell at a time unit.
- use several, parallel networks
- couterpressure-mechanism: delay blocking cells
- sorting cells at the input port, in such a way as to prevent the Banyan-network from being blocking

  => ***Batcher-Banyan***

Batcher Banyan ATM Switch Basics

- Input logic and input buffer ensures, that there is always just one cell per output port
- *Batcher* network: sorting incoming cells (see next slide)
  - Sorting cell: “arrow” indicates the output port whereto the “bigger” cell is sent to
  - Is there just a single cell at the input port – this cell will be considered as the “smaller” cell.
- *Banyan* network is no longer blocking, if sorted input ports are used. A Banyan network is working as an expander

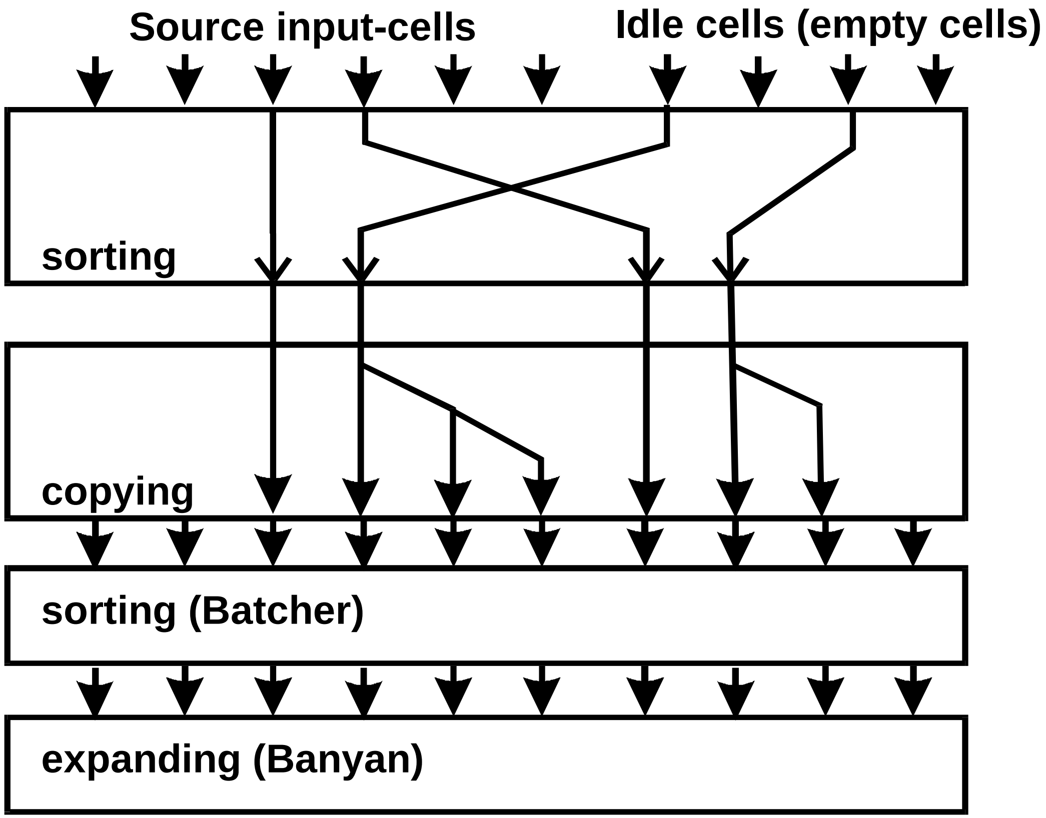
Batcher Banyan ATM Switch



Batcher Network (sorting)

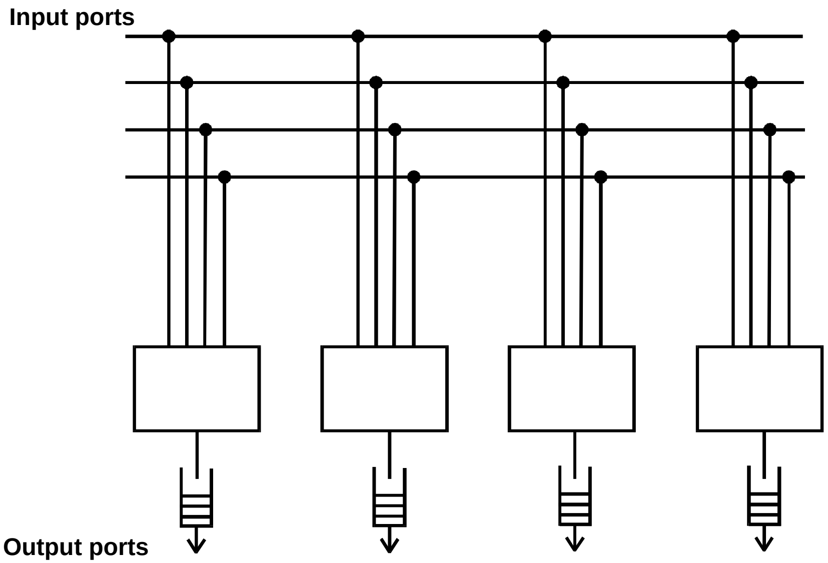
Banyan Network (expanding)

Batcher Banyan ATM Switch
multicast, broadcast



? Knock Out Switches

- Basic idea: bus-system, output queue (AT&T, 1987)
- Problem: buffer with n input ports (n-time as fast as input queue)
- *Knock Out* element
  - Accept a low internal cell loss probability to achieve acceptable memory access times.

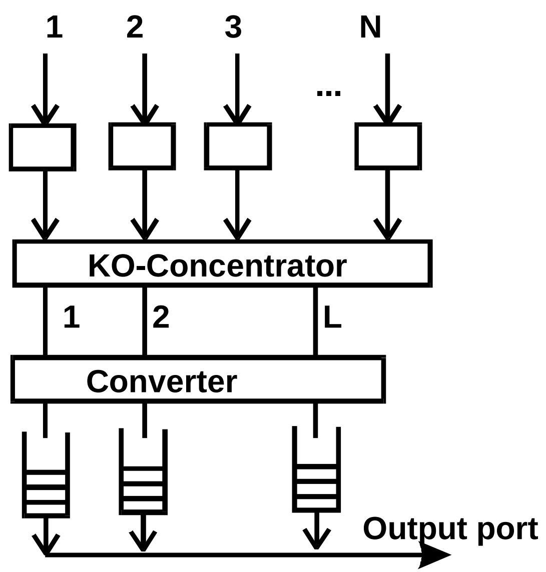


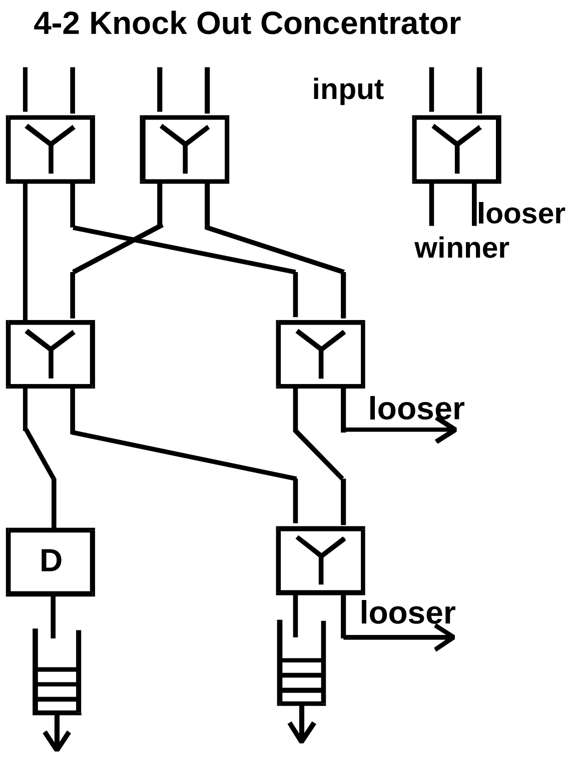
Knock Out Element

(1) Cell filter: is filtering cells, not dedicated for the output port

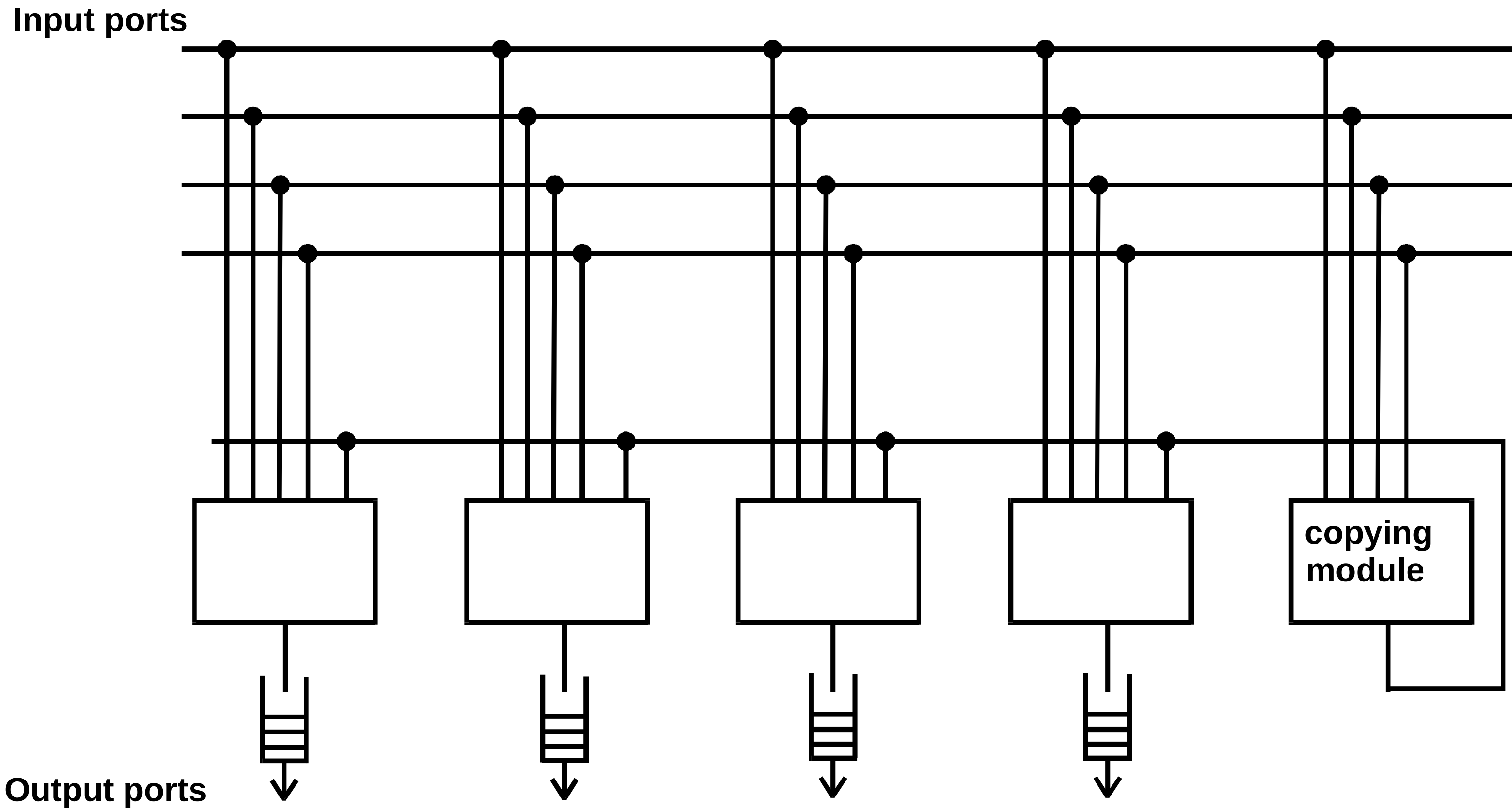
(2) Concentrator: concentrates **N** input ports onto **L** *queues*

(3) Converter: ensure a steady-going load on each of **L** *queues*



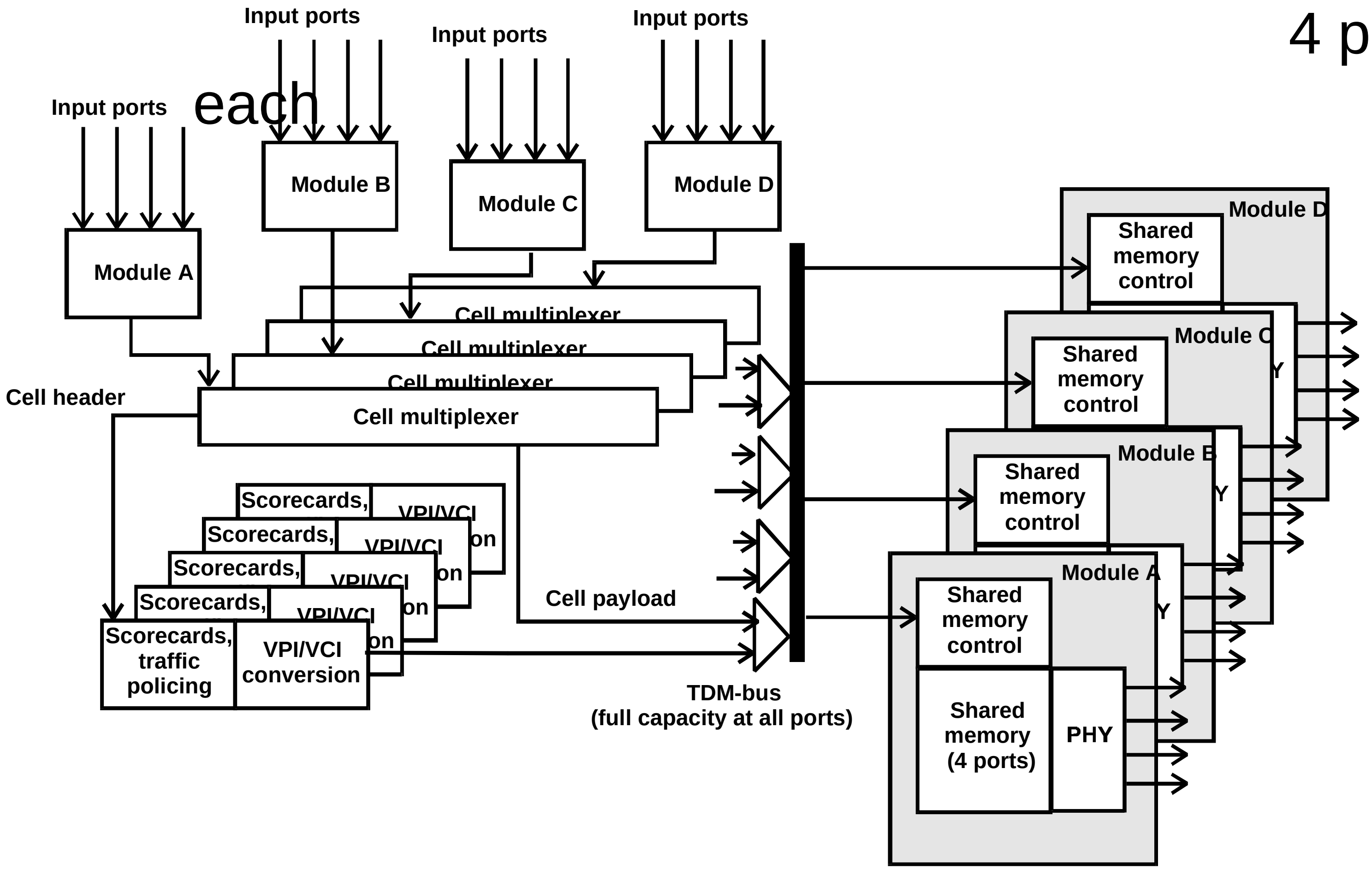


Knock Out Switch multicast, broadcast



Distributed shared memory ATM switch

Example: 4 modules, 4 ports each



QoS Terms I

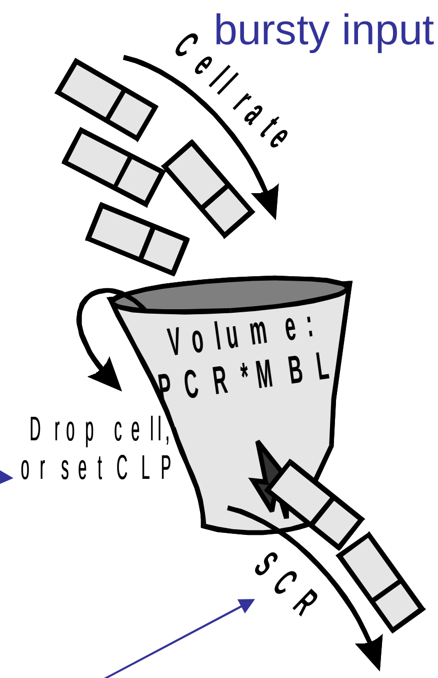
- **negotiation of QoS parameters during connection establishment**
  - *Sustainable cell rate* (**SCR**), *peak cell rate* (**PCR**), *maximum burst length* (**MBL**), *cell delay variation* (**CDV**), *cell transfer delay* (**CTD**)
- ***Connection admission control* (CAC)**
  - The network has to decide, whether a new connection can be accepted (or not) (at a specific current load)

?

QoS Terms II

- ***Traffic shaping*** (*burst attenuation*)
  - End-station is „forming“ the traffic according to the SCR
- ***Traffic policing***
  - Network (*switch*) is checking the data rate, by using statistical methods (is a problem in real-time applications)
    … or:
  - *generic cell-rate algorithm* (GCRA) implementations:
    - *Leaky bucket* or
    - *virtual scheduling* (algorithm.)

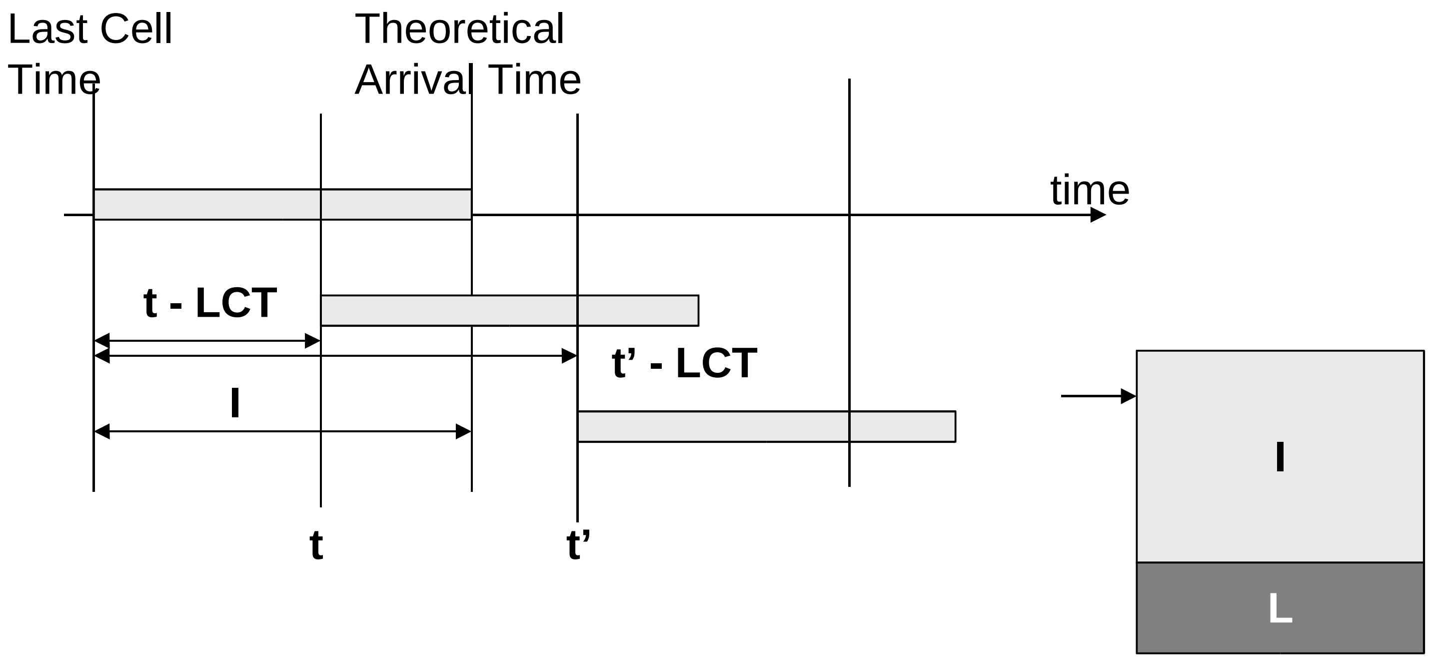
overflow
(not confirming)



One per unit of time
(confirming)

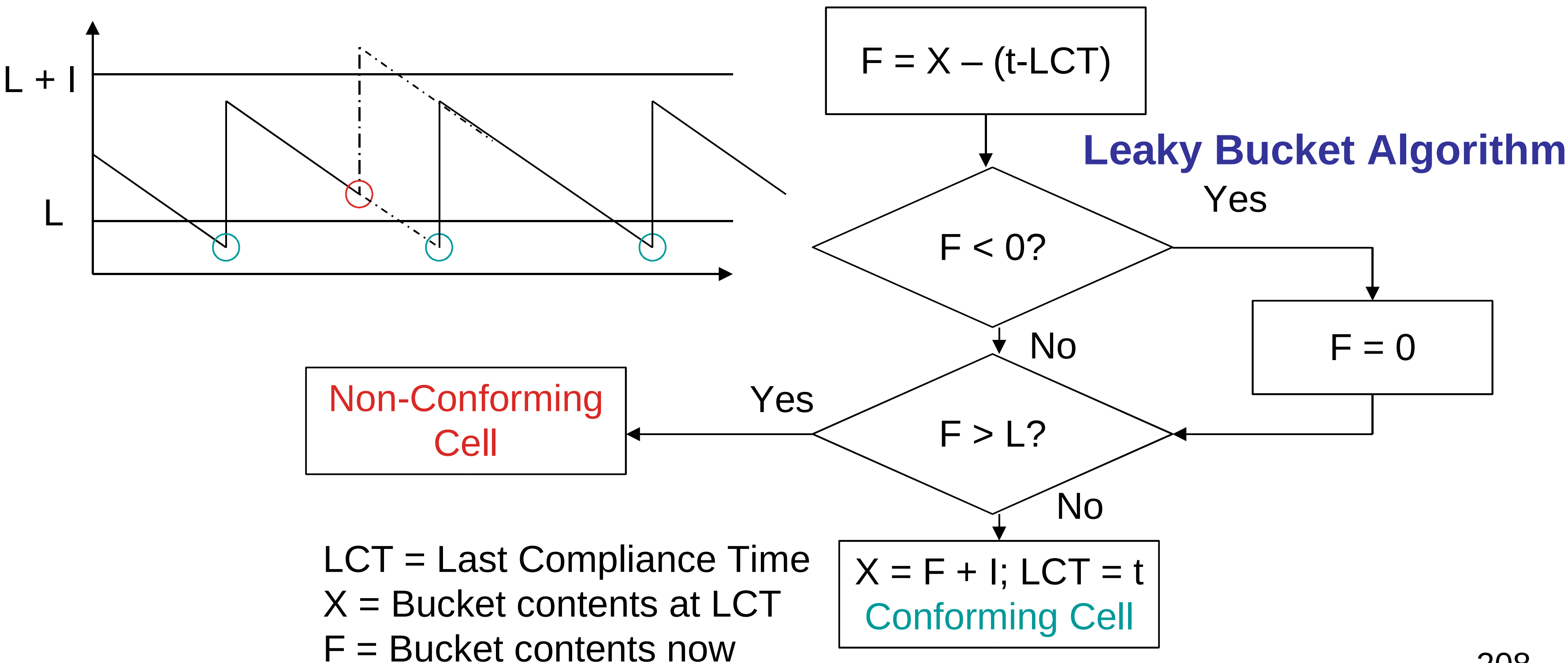
QoS Terms II

- ***Generic Cell Rate Algorithm GCRA(I,L)***
  - I = Increment = Inter-cell Time = Cell size/PCR
  - L = Limit $\rightarrow$ Leaky bucket of size I + L and rate 1



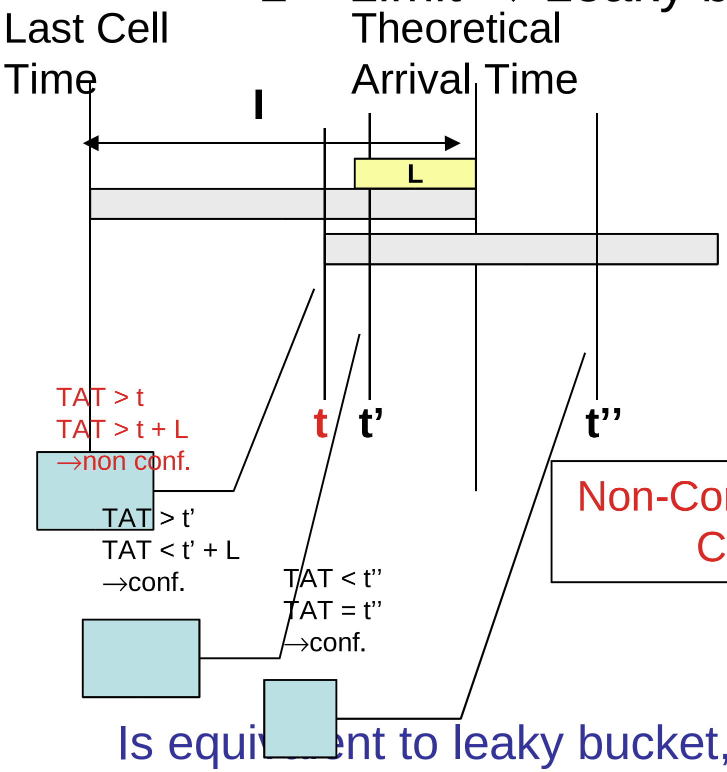
QoS Terms II

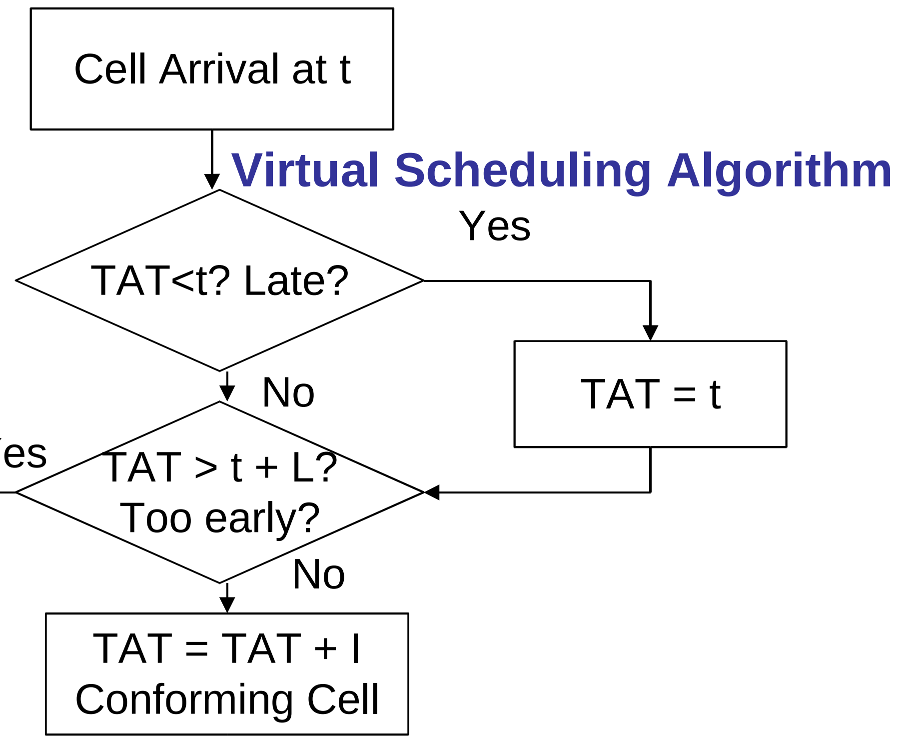
- ***Generic Cell Rate Algorithm GCRA(I,L)***
  - I = Increment = Inter-cell Time = Cell size/PCR
  - L = Limit $\rightarrow$ Leaky bucket of size I + L and rate 1



QoS Terms II

- ***Generic Cell Rate Algorithm GCRA(I,L)***
  - I = Increment = Inter-cell Time = Cell size/PCR
  - L = Limit $\rightarrow$ Leaky bucket of size I + L and rate 1



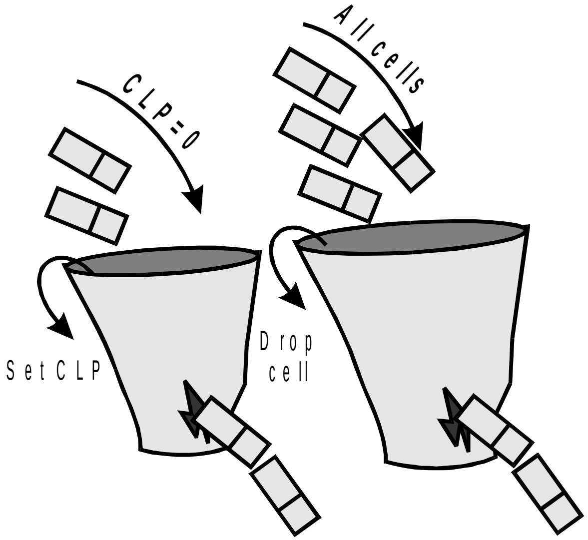


Is equivalent to leaky bucket, since the same cells are dropped.

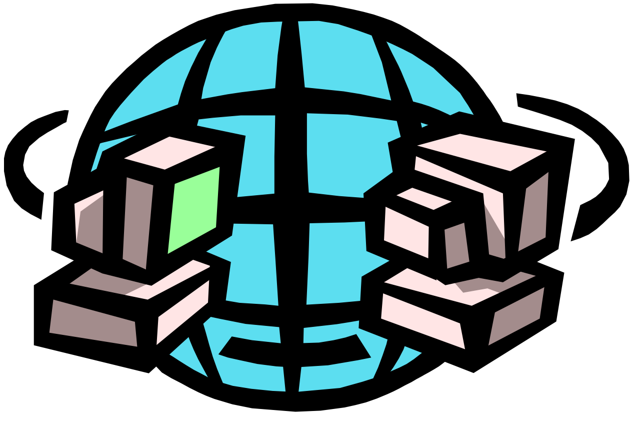
TAT =Theoretical Arrival Time

? Traffic policing II

- *Dual leaky bucket*
  - **A:** cells with CLP = 0, **B:** all cells (CLP = 0 | 1)
  - If neither rule **A**, nor rule **B** is broken:
    $\rightarrow$ cell can pass unchanged.
  - CLP will be set to „1“ if:
    - rule **A** is broken, but not rule **B**
  - Cell will be dropped, if
    - rule **A** and rule **B** is broken



Chapter 4: Internetworking



- 4.1 Basics
- 4.2 IPv4
- 4.3 IPv6
- 4.4 Routing-Protocols
- 4.5 MPLS

RN vs. [Peterson/Davie]

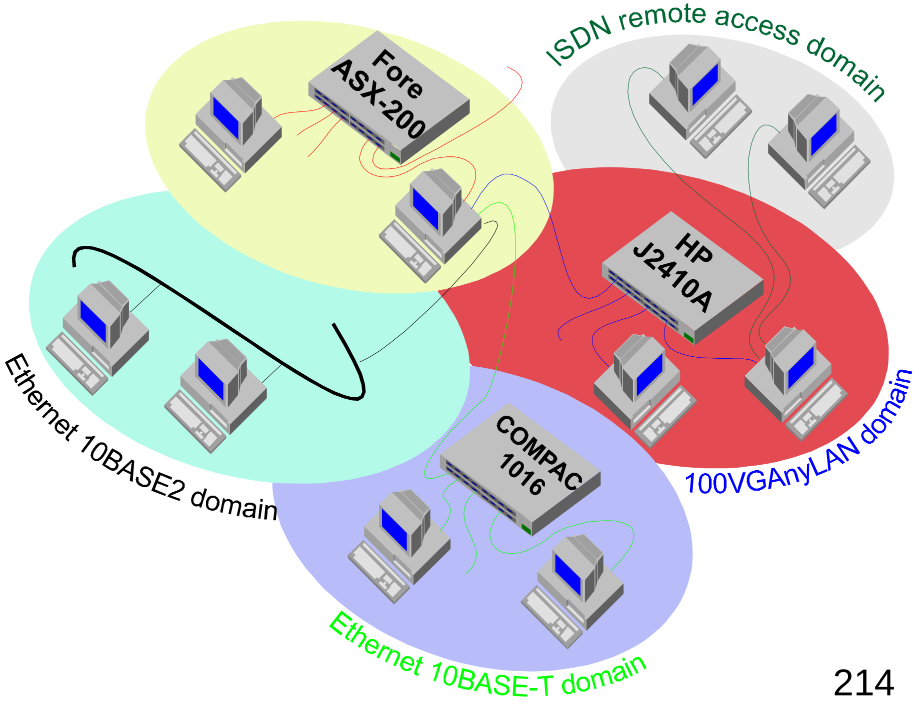
- Section 4.1 (Basics) see:
  - [Peterson/Davie] Section 4.1, 4.3
- Section 4.2 (IPv4) equivalent to:
  - [Peterson/Davie] Section 4.1
- Section 4.3 (IPv6) equivalent to:
  - [Peterson/Davie] Section 4.3.
- Section 4.4 (Routing-Prot.) equivalent to:
  - [Peterson/Davie] Section 4.2 (without 4.2.3-4.2.5)
- Section 4.5 (MPLS)

4.1 Internetworking Basics

- Repeater
- Bridges
- Router
  - Terms and definitions
- Gateways

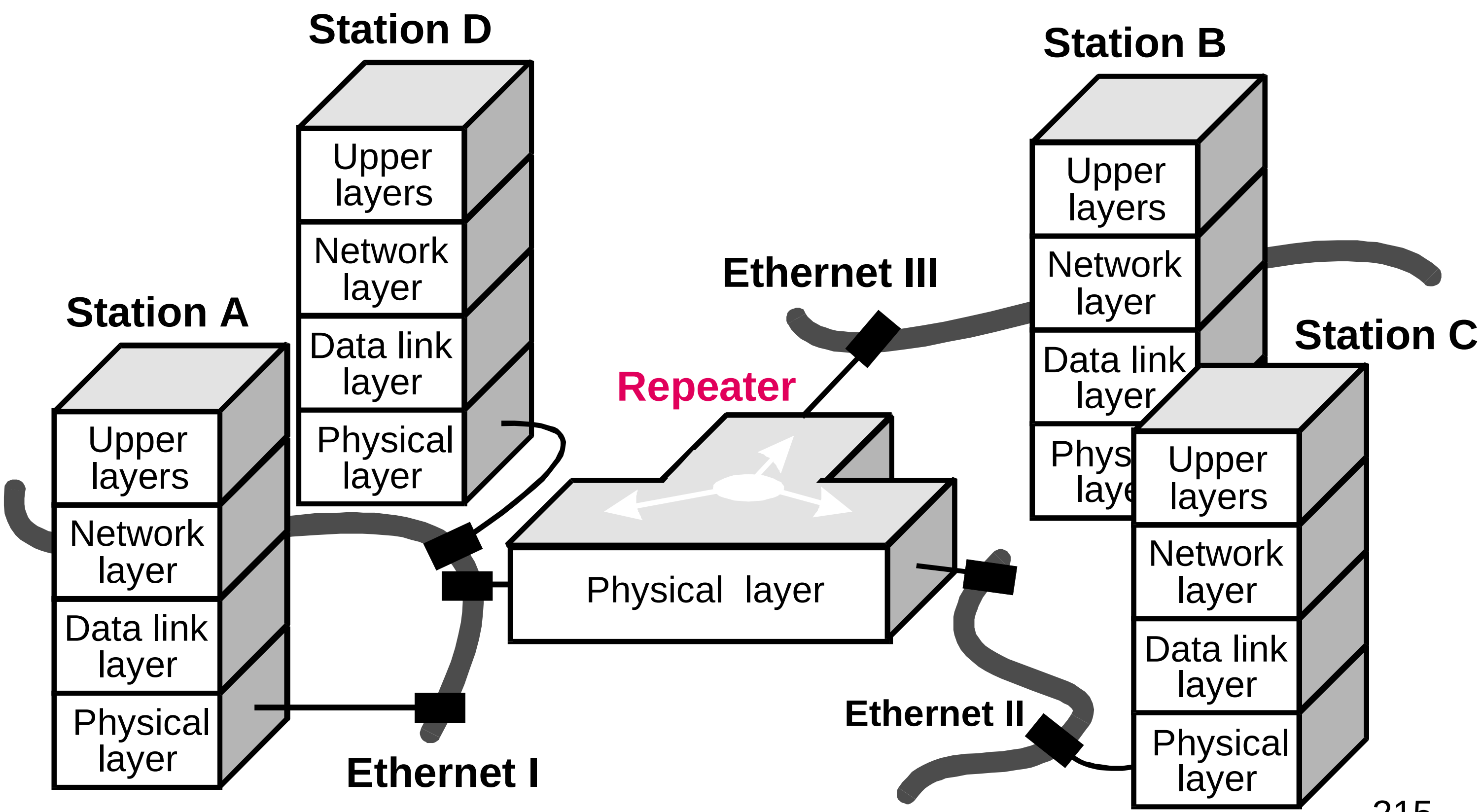
Network Connections

- Connecting homogeneous/heterogeneous networks
- Connecting isolated networks to form:
  - LAN
  - MAN
  - WAN
- Protocols ?
- Routing ?
- Addressing ?



Repeater

- Connecting at the *physical layer*

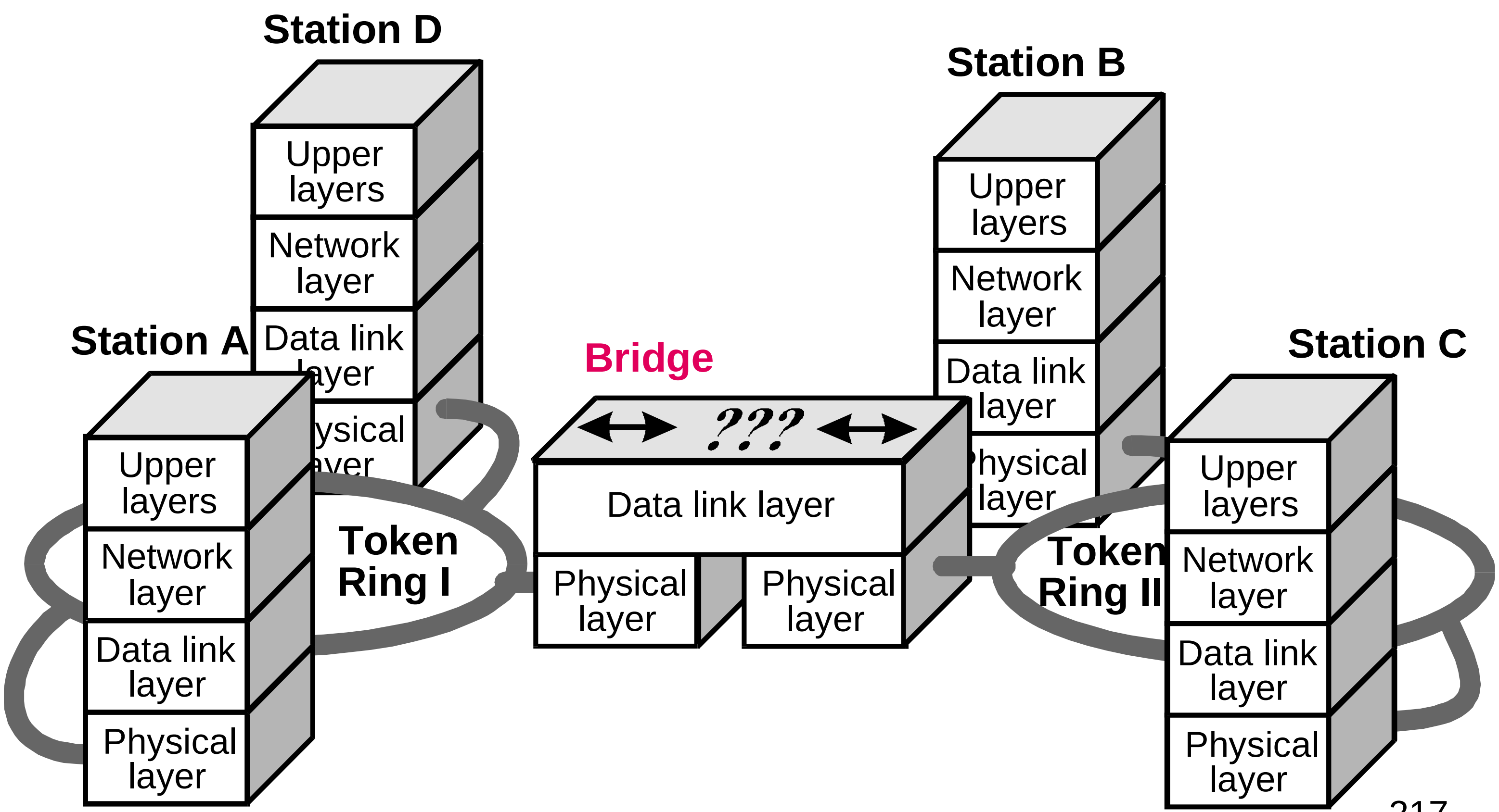


Repeater: Functionality

- Connecting homogeneous networks
- In case of Ethernet – possibly detection of:
  - *runts*
  - *babbling nodes (nodes that misbehave or disrupt normal communication)*
- Ethernet: changing the *collision domain*
  - additional delay or:
  - separation of *collision domains*

Bridge

- Connecting networks at the *datalink layer*

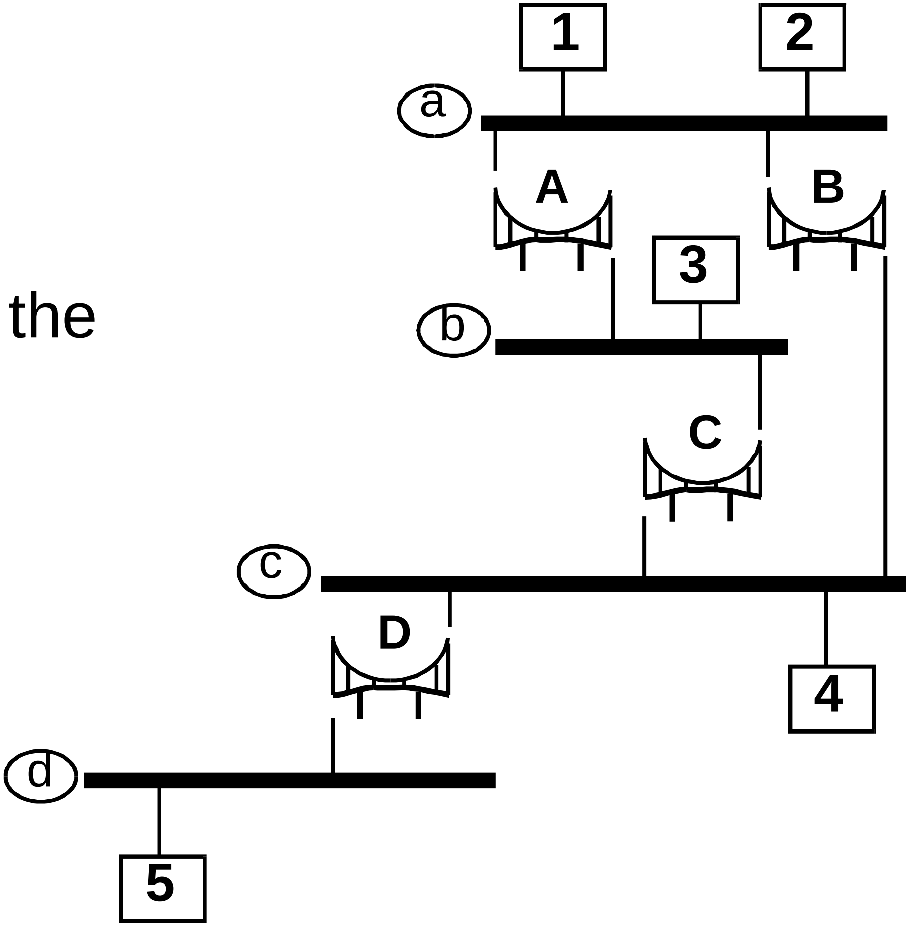


Bridge: Functionality 12.05

- Definition: No *frame-changing,* but:
  - Length field: Ethernet vs. Token Ring
  - *Encapsulation:* DIX Ethernet vs. 802.3
  - MTU: Ethernet vs. Token Ring
  - MAC-Address: Ethernet vs. 802.5, FDDI
- Address filtering (MAC)
  - Packets are passing the *Bridge,* only if there is a direct path to the target host (Routing)
- Packet buffering

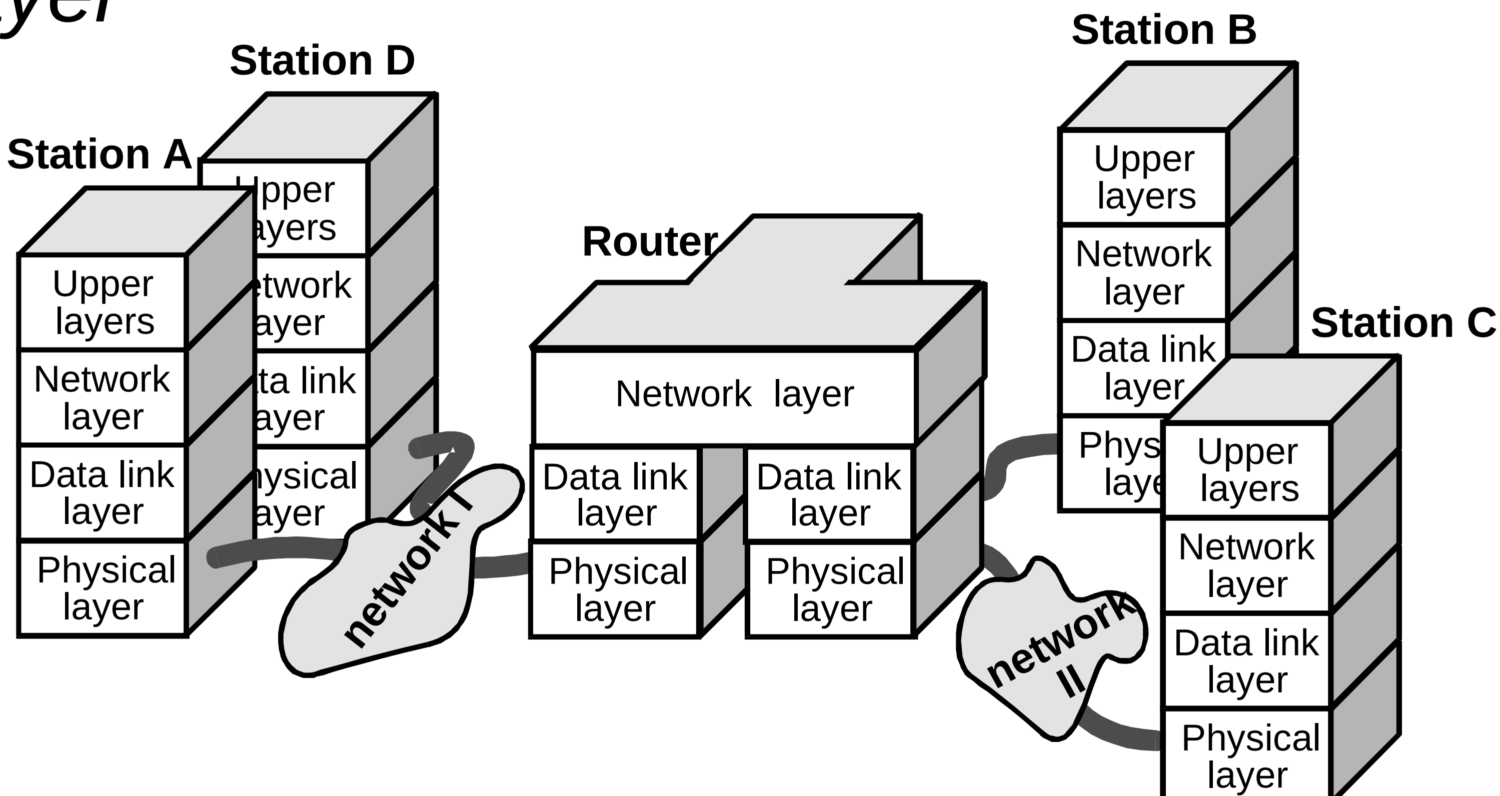
Bridge: Routing

- *Fixed routing*
  - *routing* matrix
- *Source routing*
  - *Routing* Information in the Token Ring *header*
- *Spanning tree*
  - IEEE 802.1



Router

- Connecting networks at the *network layer*

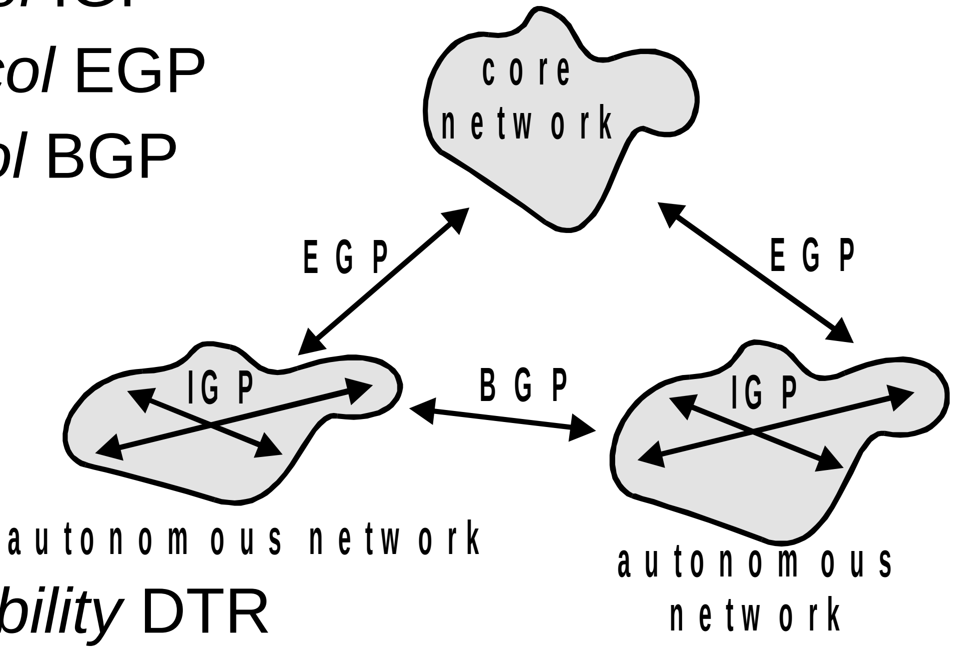


Router: Characteristics

- to decouple networks
  - e.g. *broadcast storms*
- Select the *best route* (in terms of „costs“)
- Elasticity and control of routing
- Divide into Subnets according to
  - geographical, organizational, ... criteria
- Based on *routing tables*

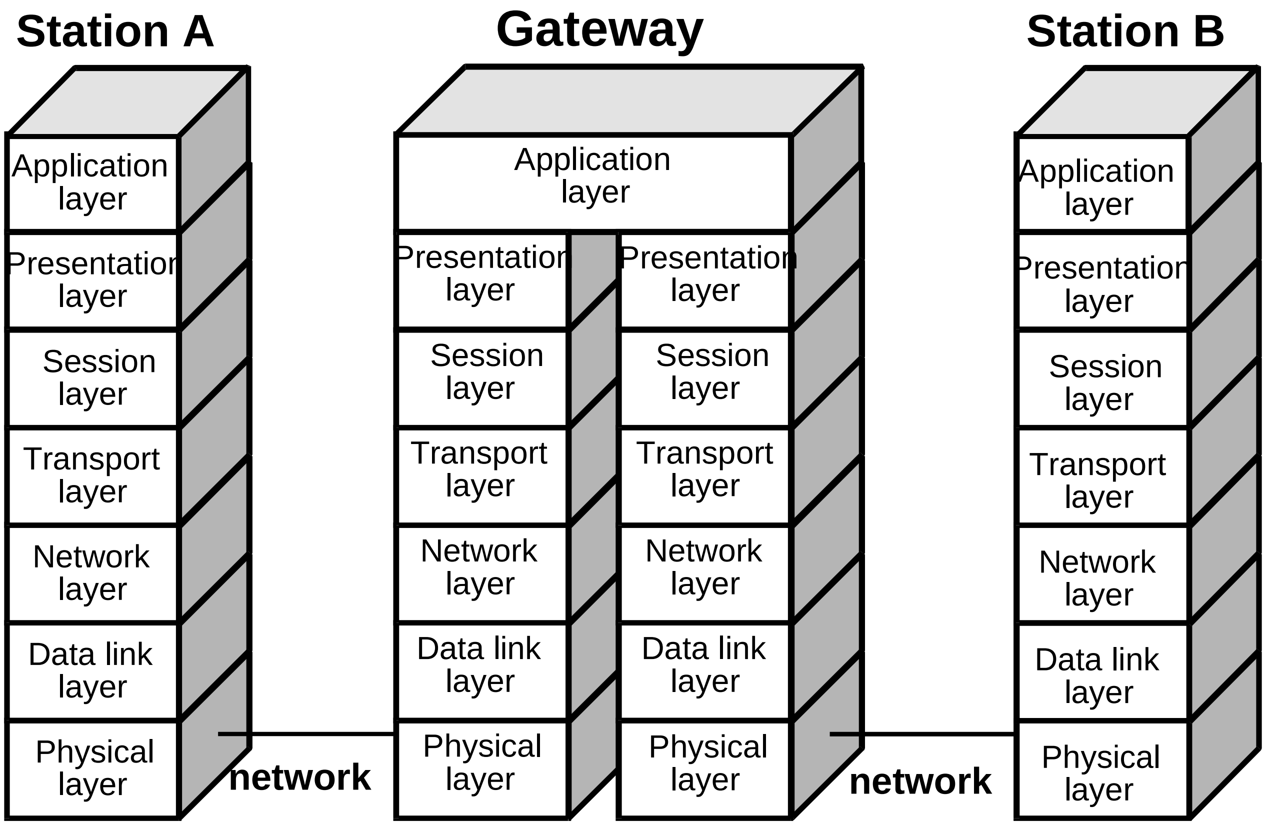
Router: Terminology

- Autonomous- vs. Core-network
  - *Interior gateway protocol* IGP
  - *Exterior gateway protocol* EGP
  - *Border gateway protocol* BGP
- Anomalies
  - *loops*
  - *black holes*
- *Metric*
  - *Delay, throughput, reliability* DTR

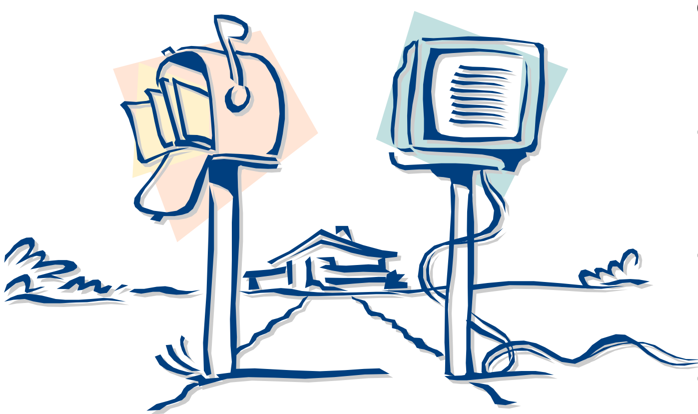


Gateway

- Connecting „above“ the *network layer*

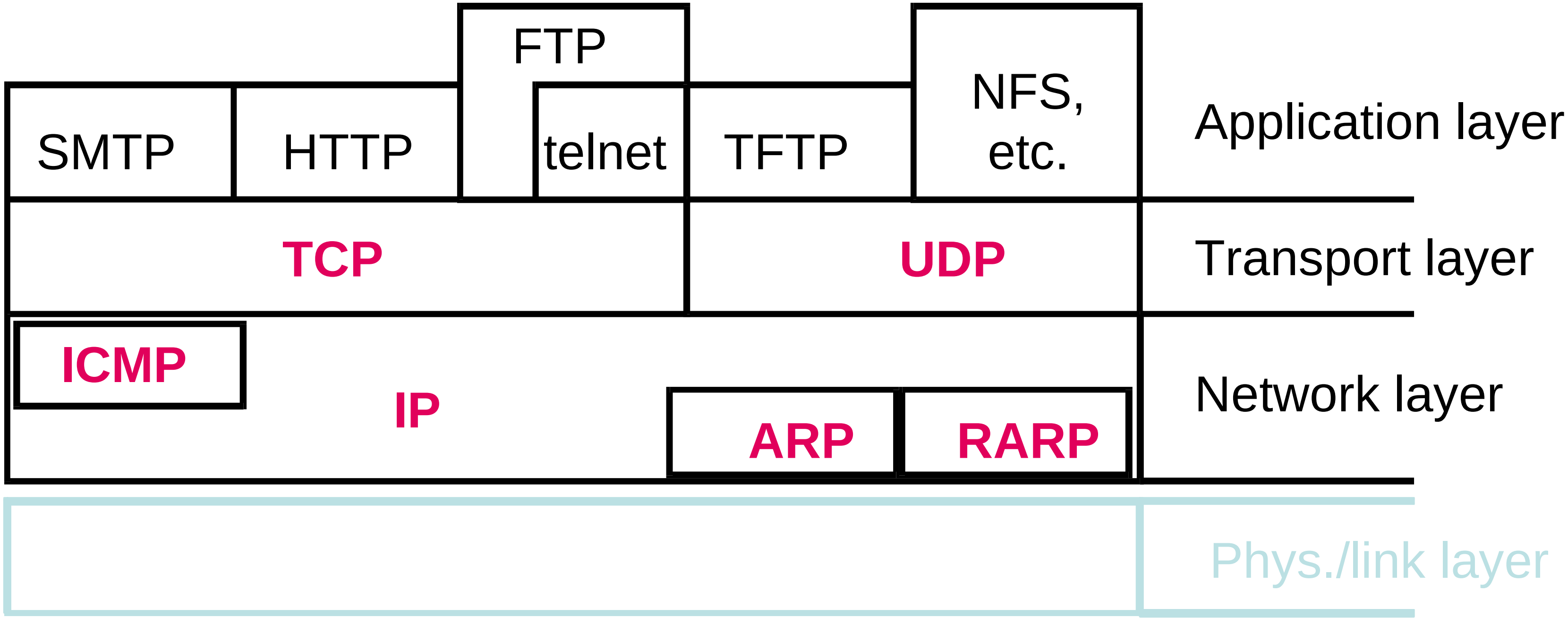


4.2 Internet Protocol version 4



- Characteristics
- IP Datagram
- Addressing, sub-networking
- ARP & ICMP

TCP/IP Protocol



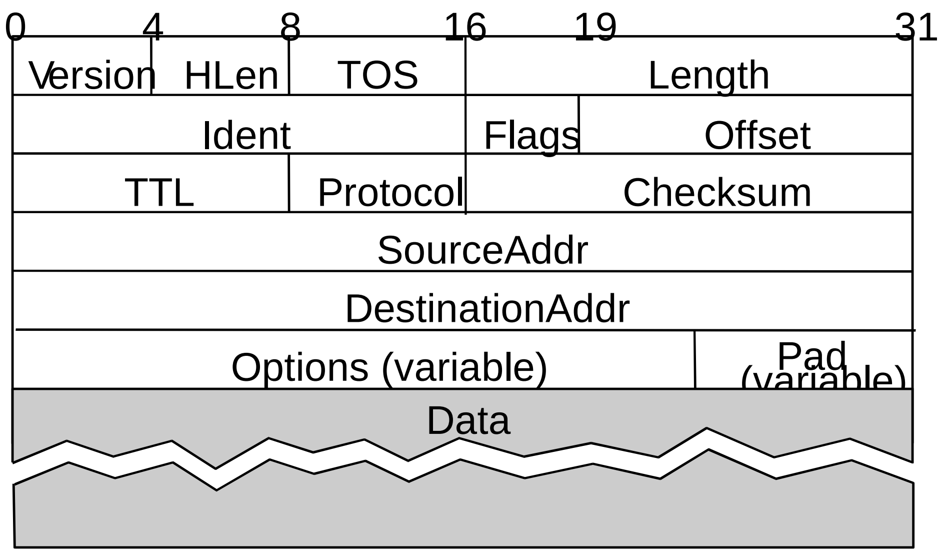
IPv4 Characteristics

- connectionless, based on datagrams
  - potentially lost datagrams
  - sequence of datagrams is not granted
  - potentially duplicated datagrams
  - Checksum by considering the whole IP header
- Datagrams
  - min. 20 byte header *(HLen=4b, counts 32b words !!)*
  - max. 65,536 byte *(including the header, Length=16b)*

IPv4-header (contd.)

- min. 20 Byte header
  - Version (current version = 4), HLen, TOS (2 Byte)
  - Total length *(65,536 including the header → fragm.&reass.)* (2 Byte)
  - Identification (2 Byte)
  - Flags, Fragment offset (2 Byte)

    Info to fragm. & reass.
  - Time to live *(catch routing loops)* (1 Byte)
  - Protocol *(multiplexing key to identify higher layer protocols)* (1 Byte)
  - Header Error Checksum (2 Byte)
  - Source IP address, Destination IP address (8 Byte)

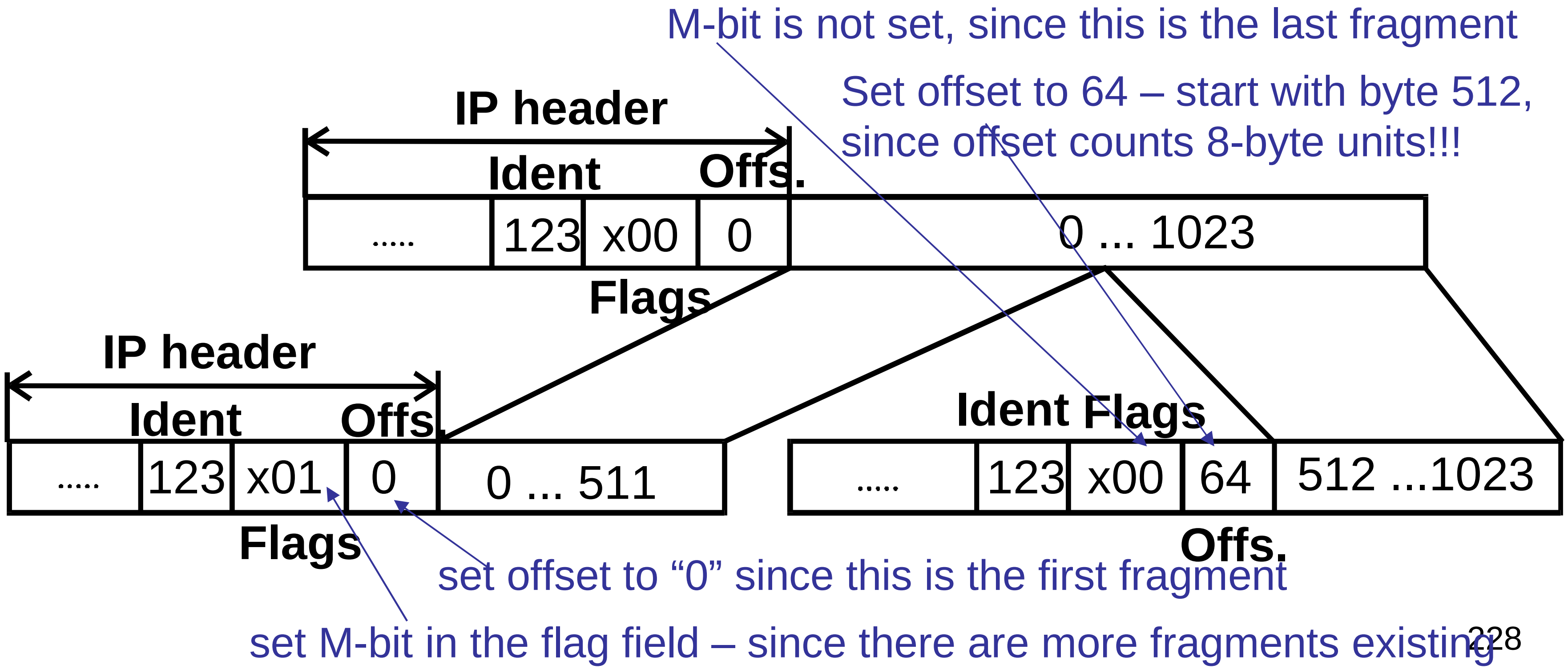


- Options (variable)

number ob option words can be checked by HLen

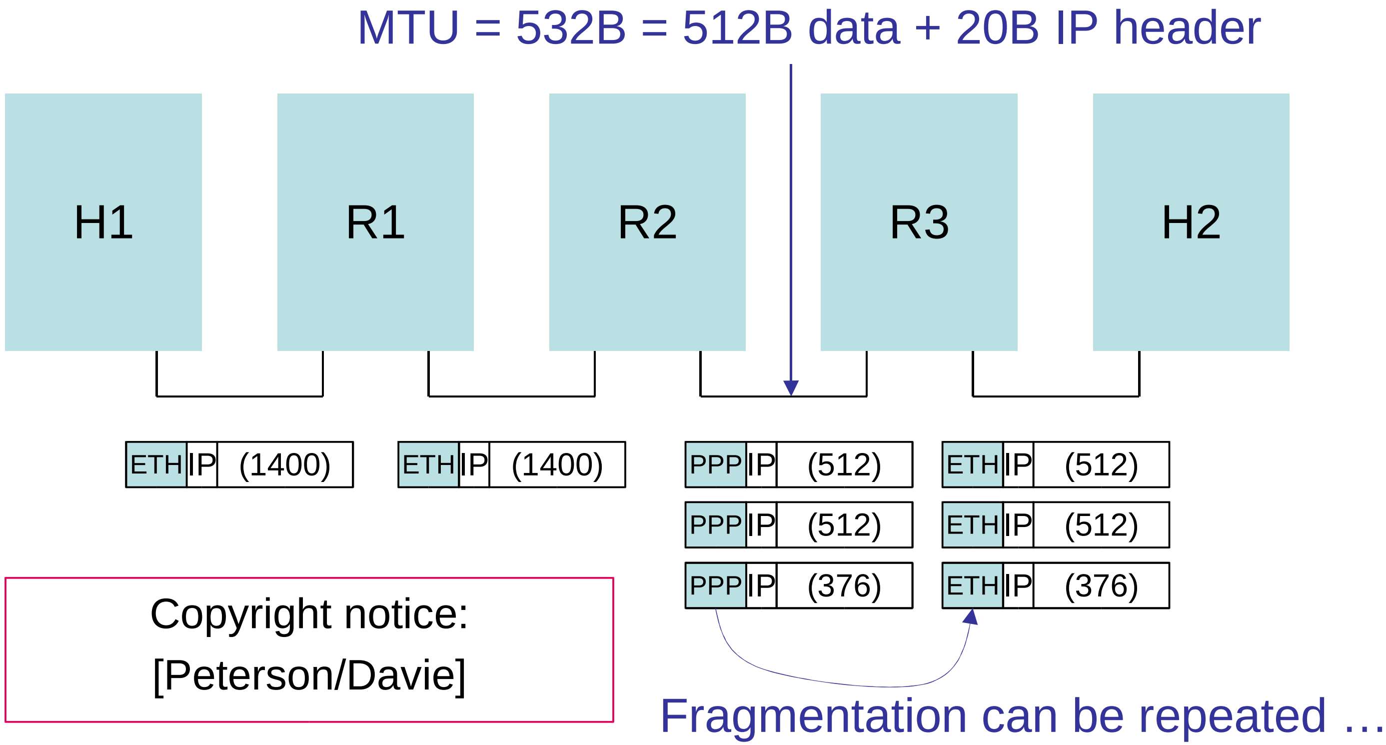
? IP Fragmentation

- Is sometimes required between two different networks:
  - Ethernet up to 1500B packet length, FDDI may be 4500B
  - Every network type has a MTU $\rightarrow$ specifying the largest IP datagram



IP Fragmentation

- Two important points:
  - Each fragment is a self-contained IP packet that is transmitted independent of other fragments
  - Each IP datagram is re-encapsulated for each physical network over which it travels



? IP Addresses

- IP Addresses consists of:
  - Class bits, Network part, Host part
- IP Network classes
  - Class A: 1.h.h.h up to 126.h.h.h *(126 class A networks, 0 and 127 are reserved → 2^{24}-2=about 16 million hosts)* (0…)
  - Class B: 128.1.h.h up to 191.254.h.h *(65,534 hosts each)* (10…)
  - Class C: 192.0.1.h up to 223.255.254.h *(256 unique hosts)* (110…)
  - Class D: class bits 1110, *(the other 28 bits are used to identify the group of computers the **multicast message** is intended for.)*
  - Class E: class bits 1111, (IAB*)) intern, is used for experimental purposes only

  - Broadcast: Messages that are intended for all computers on a network are sent as **broadcasts**. These messages always use the IP address **255.255.255.255**.

*) *IAB: Internet Architecture Board*

Datagram Forwarding Algorithm

```
if (dest. network-number == network-number of one of my interfaces)
then
   deliver packet to destination over that interface
else
   if (dest. network-number == in my forwarding table)
   then
        deliver packet to next hop router
   else
        deliver packet to default router

// If there is just one interface installed…

if (dest. network-number == my network-number)
then
   deliver packet to destination directly
else
   deliver packet to default router
```

ARP/RARP

- How to get IP datagrams to the right physical interface??
  - Derive IP from MAC address xx-..-xx-21-51→ y.y.33.81 class C
  - Maintain a table for address pairs
  - Dynamically learn the content of the table (ARP)

… if sending and receiving IP address is in the same network:

- Phase 0:
  - Check the cache for a mapping → Phase 1 if no mapping exist
- Phase 1: Request
  - Broadcast an ARP query onto the network
- Phase 2:
  - Each host receives the query and checks to see if it matches its IP address
  - If it does match – send a response message *(containing the dest. MAC address)*
  - Add this information to the table
- Query includes IP and MAC of the sending host – thus, each host on the network **"can"** learn the senders IP-MAC mapping

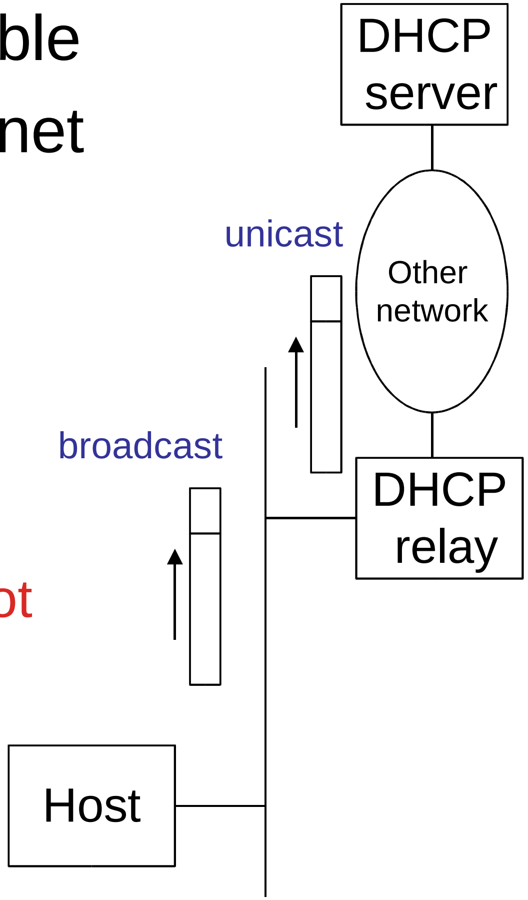
```
arp -a

Interface: 129.27.152.200

Internetadresse Physikal. Adresse   Typ
 129.27.152.30  00-08-c7-ec-af-d0   dyn.
 129.27.152.34  00-20-48-0e-42-59   dyn.
```

DHCP

- IP addresses needs to be configurable
- Additional required information: subnet mask, and default gateway.
- DHCP server is providing this information, maintaining a pool of available addresses.
  - Client is sending a DHCPDISCOVERY message to FF.FF.FF.FF (routers do not forward this messages $\rightarrow$ 1 server per network or DHCP relays)
  - Server will reply to the host
  - Scaling of network management

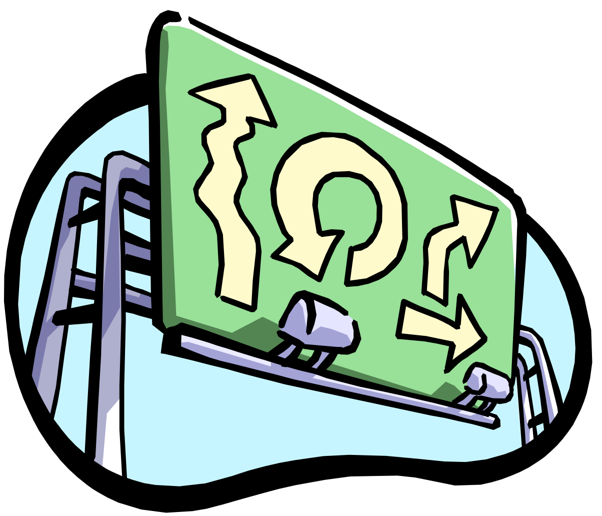


ICMP

- several control messages
  - Echo request, Echo reply (ping)
  - Destination unreachable
  - Source quench *The source quench message is a request to the host to cut back the rate at which it is sending traffic to the internet destination.*
  - Redirect *tells the source host, that there is a better route to the destination*
  - Time exceeded *- the TTL has reached 0*
  - Parameter problem
  - etc.

In eigener Sache:

4.4 Routing - Protocols



- RIP
- OSPF
- ARPANET „history“

Some Routing Protocols

- *Routing information protocol* RIP (IGP)
- *Hello* (IGP)
- *Open shortest path first* OSPF (IGP)
- *Interior gateway routing protocol* IGRP (Cisco IGP)
- *Integrated intermediate system to intermediate system* IS-IS (OSI IGP)
- *Exterior gateway routing protocol* EGRP (IGP,EGP,BGP)
- *Border gateway protocol* BGP (BGP)
- *Gateway to gateway protocol* GGP (Internet *core net.*)

?

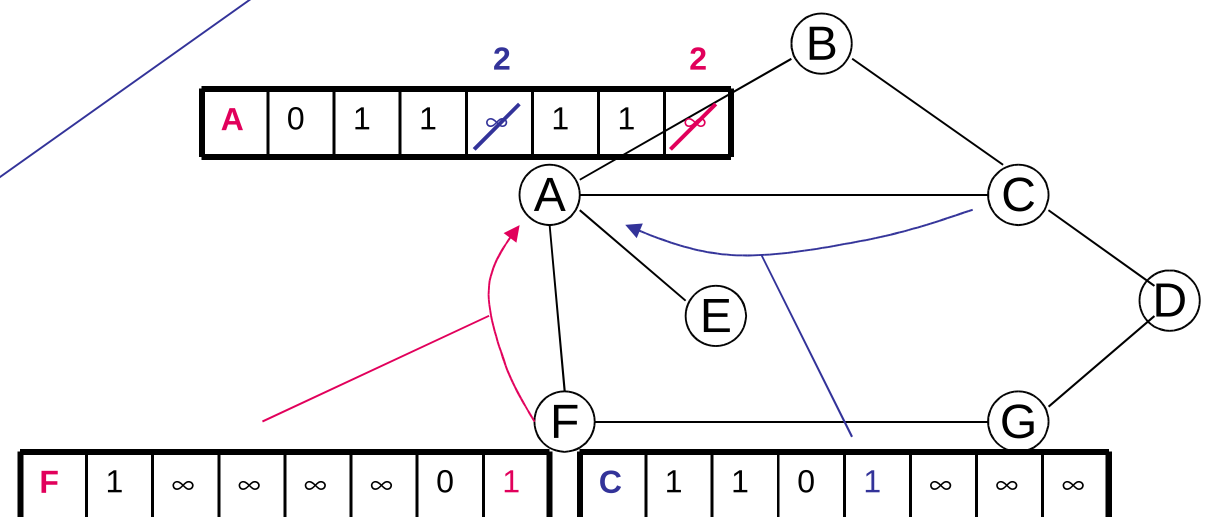
Distance Vector

- Each node constructs a distance vector
- … distributes this vector to its neighbors
- A link that is down is assigned an infinite cost

final distances

	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0

	A	B	C	D	E	F	G
A	0	1	1	∞	1	1	∞
B	1	0	1	∞	∞	∞	∞
C	1	1	0	1	∞	∞	∞
D	∞	∞	1	0	∞		1
E	1	∞	∞	∞	0	∞	∞
F	1	∞	∞	∞	∞	0	1
G	∞	∞	∞	1	∞	1	0



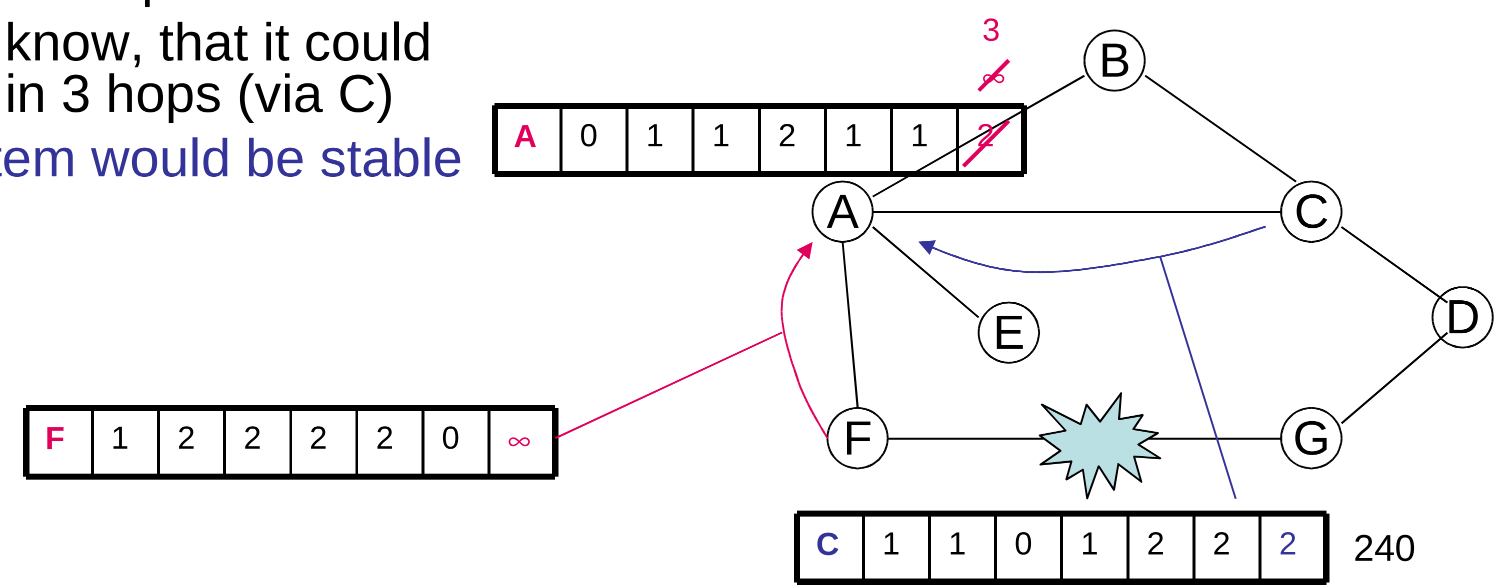
Distance Vector

- If there are no topology changes, it only takes a few messages to complete all routing tables
- This process is called “convergence”
- Updates:
  - Periodically: … even if nothing has changed – just to show that this node is still running
  - Triggered Update: whenever a node receives an update from one of its neighbors that causes it to change one route
- Detecting Link Failures:
  - Nodes continually tests the links to other nodes
  - If a node does not receive the expected periodic routing update
- The node that notice first sends new lists of ∞-distances

Distance Vector

- F detects, that its link to G has failed.
- F sets its new distance to G to ∞ and passes this info to A
- A would also set its distance to G to ∞
- A would learn from C that there is a 2-hop link to G
- A would know, that it could reach G in 3 hops (via C)
- The system would be stable again

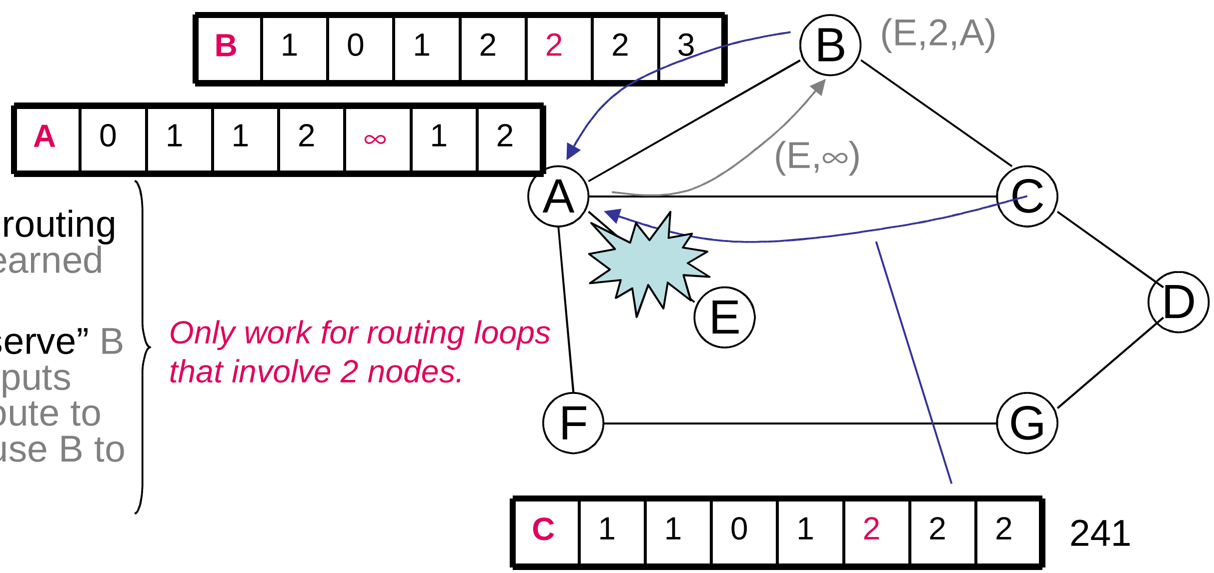
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Distance Vector

- Link A-E is broken
- A advertises a distance ∞ to E
- But B and C advertise a distance of 2 to E
- Depending on the timing:
  - B hears, that E can be reached in 2 hops via C – concluding that it can reach E in 3 hops – advertises this to A
  - A concludes, that it can reach E in 4 hops
  - C concludes, that it can reach E in 5 hops
  - …
- Count of infinity problem
- Solutions:
  - Us small numbers as infinity
  - Improve the time to stabilize routing "split horizon" (do not send learned routs back to the neighbor)
  - "Split horizon with poison reserve" B sends route back to A, but it puts negative information in the route to ensure, that A will not try to use B to get to E

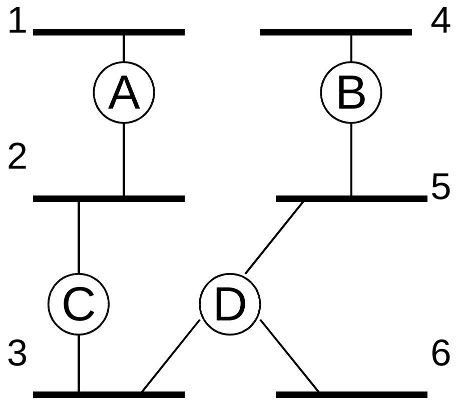
	A	B	C	D	E	F	G
A	0	1	1	2	1	1	2
B	1	0	1	2	2	2	3
C	1	1	0	1	2	2	2
D	2	2	1	0	3	2	1
E	1	2	2	3	0	2	3
F	1	2	2	2	2	0	1
G	2	3	2	1	3	1	0



Only work for routing loops that involve 2 nodes.

Distance Vector (RIP)

- *Routing Information Protocol*
  - Protocol, build on distance-vector algorithm
  - Distributed with Berkeley Software Distribution (BSD)
  - The routers advertise the costs of reaching networks
  - Example: C advertises to A that it can reach networks:
    - 2 and 3 @ cost 0
    - 5 and 6 @ cost 1
    - and 4 @ cost 2



Distance Vector (RIP)

- *Routing Information Protocol*
  - Routers running RIP every 30 seconds
  - A router sends an update message whenever it was forced to change its routing table
  - Supports multiple address families (not just IP)
  - It always tries to find the minimum hop route
  - Valid distances: 1-15, 16 representing ∞
  - Thus, limited to small networks (no paths longer than 15)

? Link State (OSPF)

- Basic idea:
  - Every node knows its directly connected neighbor
  - This knowledge is disseminated to every node.
  - Link state routing protocols rely on two mechanisms:
    - Reliable dissemination, and the
    - Calculation of routes from the sum of all knowledge
- Reliable Flooding:
  - Flooding := each node sends its link state information to all its directly connected links – this continues until the information has reached all nodes in the network
  - Link state package:
    - The ID of the node that creates the LSP
    - List of directly connected neighbors
    - A sequence number
    - A time to live for this package

calculate route info

flooding

Link State (OSPF)

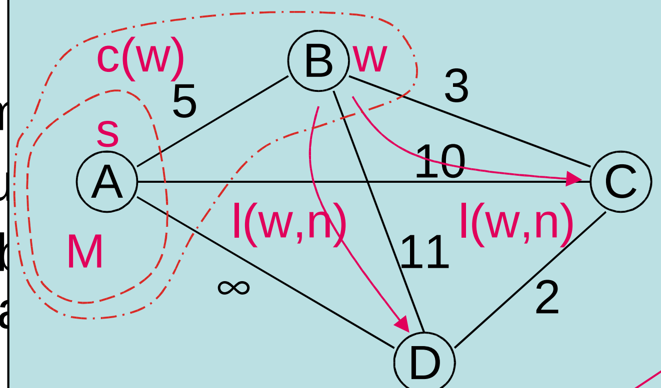
- Transmission of LSPs is <u>based on reliable protocols</u> (like link layer protocols – to ensure, to have the most recent copy of the LSP)
  - ACKs and retransmission
  - Using sequence numbers to differ between new and old LSPs
  - LSPs are not sent back to the node from which the LSP was received (bringing an end to the LSP flooding)
  - Route calculation:

```
M={s} //M, a set of nodes - incorporated so fare. Start with “s”
for each n in N-{s}         //N a set of nodes in the graph
  C(n)=l(s,n)               //costs from s to each node n
while (N M)
  M=M ∪{w} such that C(w) is the minimum for all w in (N-M)
  for each n in (N-M)
    C(n)=Min(C(n),C(w)+l(w,n))
```

Link State (OSPF)

- Transmission of LSPs is based on reliable protocols (like link layer protocols – to ensure, to have the most recent copy of the LSP)
  - ACKs and retransm
  - Using sequence nu
  - LSPs are not sent b received (bringing a
  - Route calculation:

```
M={s} //M, a set of
for each n in N-{s}
  C(n)=l(s,n)
while (N M)
  M=M ∪{w} such that C(w) is the minimum for all w in (N-M)
  for each n in (N-M)
    C(n)=Min(C(n),C(w)+l(w,n))
```



s	C(n)s	C(n)	C(n)
B	5	5	5
C	10	8	8
D	∞	16	10

w

w'

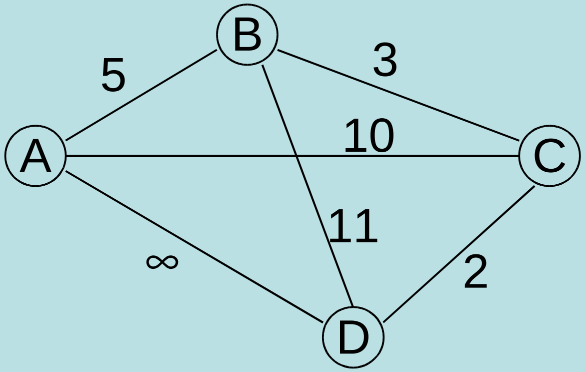
Dijkstra: Forward Search

Link State (OSPF)

- In practice, each switch computes its routing table from the LSPs it has collected
- Each switch maintains two lists: Tentative and Confirmed
- … with entries of the form: (*destination, cost, next hop*)

1. Initialize Confirmed with an entry of myself (D,0,-)
2. Call this node "Next"
3. For each neighbor of "Next" calculate the cost to reach Neighbor as a sum of (Myself2Next and Next2Neighbor)
    1. If neighbor is on no list, add (Neighbor, Cost, NextHop) to Tentative
    2. If Neighbor is on Tentative and Cost is less than listed cost of Neighbor, then replace the current entry with (Neighbor, Cost, Nexthop), where Nexthop is the direction I go to reach Next
4. If Tentative is empty, stop. Otherwise pick an entry from Tentative with lowest cost, move it to Confirmed, and return to step 2

Link State (OSPF)



Step	Conf.	Tent.	com.
1	(D,0,-)		D is the only new member – look at its LSP
2	(D,0,-)	(B,11,B) (C,2,C)	Ds LSP says, that we can reach B at cost 11 and C at cost 2. Put it on Tentative
3	(D,0,-) (C,2,C)	(B,11,B)	Put lowest cost member of Tentative onto confirmed. Examine LSP of C
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). Cs LSP tells us that we can reach A at 12
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest cost member of Tentative to Confirmed, than look at its LSP
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative Entry
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative to Confirmed, and we are all done

Link State (OSPF)

- Open Shortest Path First (OSPF)- additional features:
    - Open: nonproprietary standard
    - SPF: alternative name for link state routing
  - Authentication of routing messages (strong cryptographic authentication is required)
  - Additional hierarchy: Domains can be partitioned in areas. A router within a domain does not need to know how to reach every network within this domain (reduction of information)
  - Load balancing: allows multiple routes to the same place to be assigned the same cost and will cause traffic to be distributed evenly over those routes

„Original“ ARPANET Routing

- Let´s assume: the „*queue length*“ applies with the delay
- Is using 2 vectors (used from 1969 to 1979)
  - $D_i = [d_{ij}]$ min. *delay (queue length)* from *node*$_i$ to *node*$_j$ (d_{ii}=0)
  - $S_i = [s_{ij}]$ the next “*minimum delay node*” on the way from *node*$_i$ to *node*$_j$
- Each node is sending its D_i to all neighbors every 128 ms, each neighbor k is calculating a new …
  - $d_{kj} = Min[d_{ij} + d_{ki}]$ for all neighbors i
  - $s_{kj} = i$ for those i, which is forming **min. d**, whereas d_{ki} are actual costs from k to i.
- Disadvantage
  - *Link speed* is not taken into account
  - *Queue length* is a bad indicator for *delay*

ARPANET Routing, 2.Attempt

- direct measurements of *delay‘s* (from 1979 to 1987)
  - Packets are provided with indicators for: *arrival time* and *departure time* (at each router)
  - If there is a positive *acknowledgement,* the *delay* will be calculated
  - Average delay’s were calculated every 10 seconds
- If there are significant changes in the *delay vectors,* this information is flooded into the net.
- Each *router* is calculating its *routing table* (using Dijkstra)
- Disadvantage
  - Even the measurement of delay’s is not sufficient to find the right routing entries (especially in times of high loads)
  - => *Routing* Table can be wrong, as soon as it is recalculated.

ARPANET Routing (once more)

- The problem of „Version-2“ is, that each node tries to use the best route. This leads to *link oscillation* in the case of a high loaded network.
- Approach: In the case of a highly loaded network, assign just an “average-routs” to a specific path – and do not try to find the perfect route for all paths.
- Another way to calculate *link costs*.
  - Queuing theory methods are used to transform the measured delay into a link-load assessment (each 10 seconds).
  - Additional, a weighted average value is calculated – to take into account present, and passed activities (by using different weights)
- Finally, the routing table is calculated, by using this modified *delay vectors.*

Subnetting

- Provides an elegantly simple way to reduce the number of network numbers
- Use a single IP network number and allocate IP addresses to several physical networks which are now referred as *subnets*
  - The subnets should be close to each other (since the router will only be able to select <u>one route to reach any of the subnets</u>)
  - All hosts on the same network must have the same network number
  - All hosts on the same physical network will have the same subnet number

Network number	Host number

Class B

111111111111111111111111	00000000

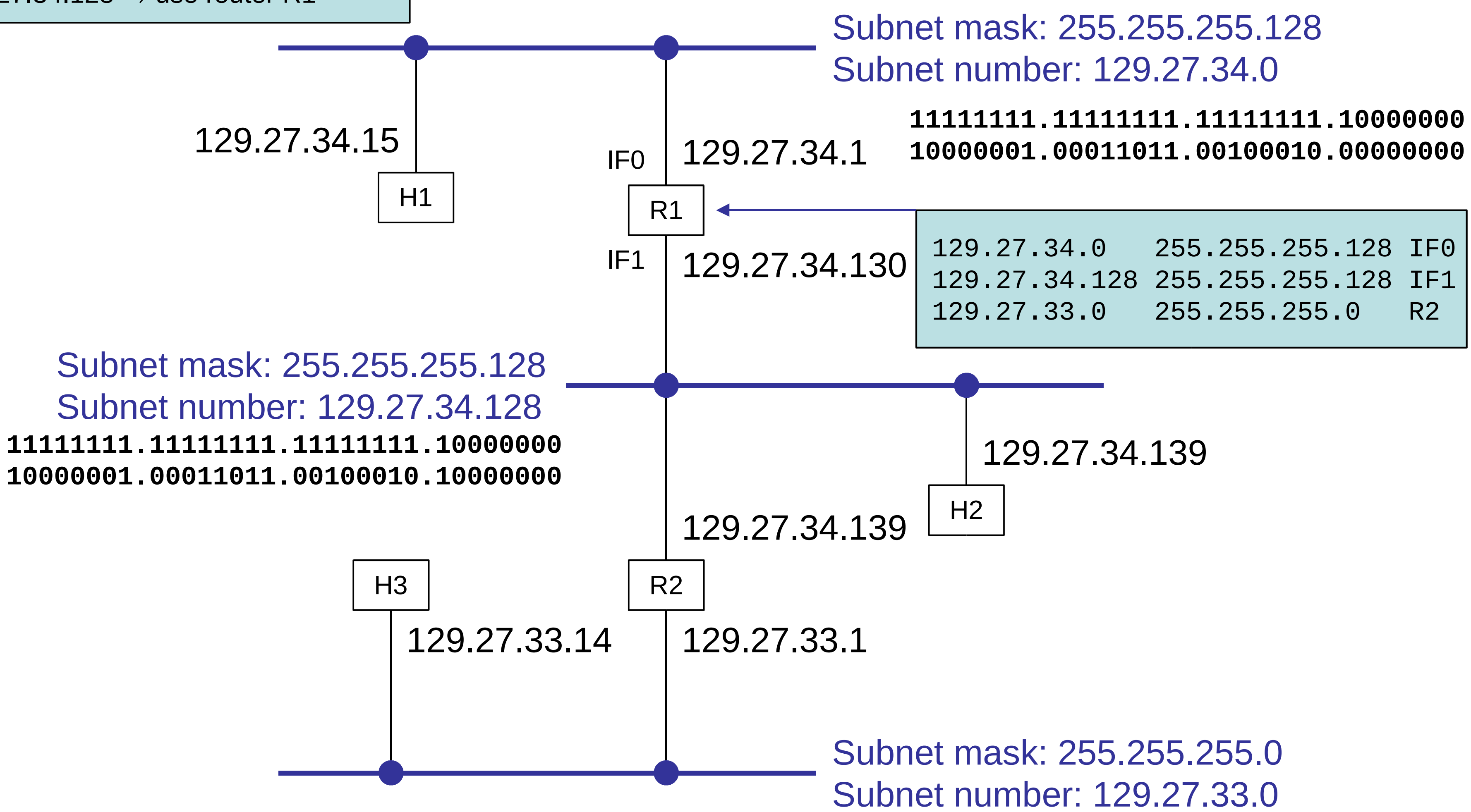
Network number	Subnet ID	Host ID

Subnetting

Send packet to 129.27.34.139

129.27.34.139 AND 255.255.255.128 =

129.27.34.128 → use router R1



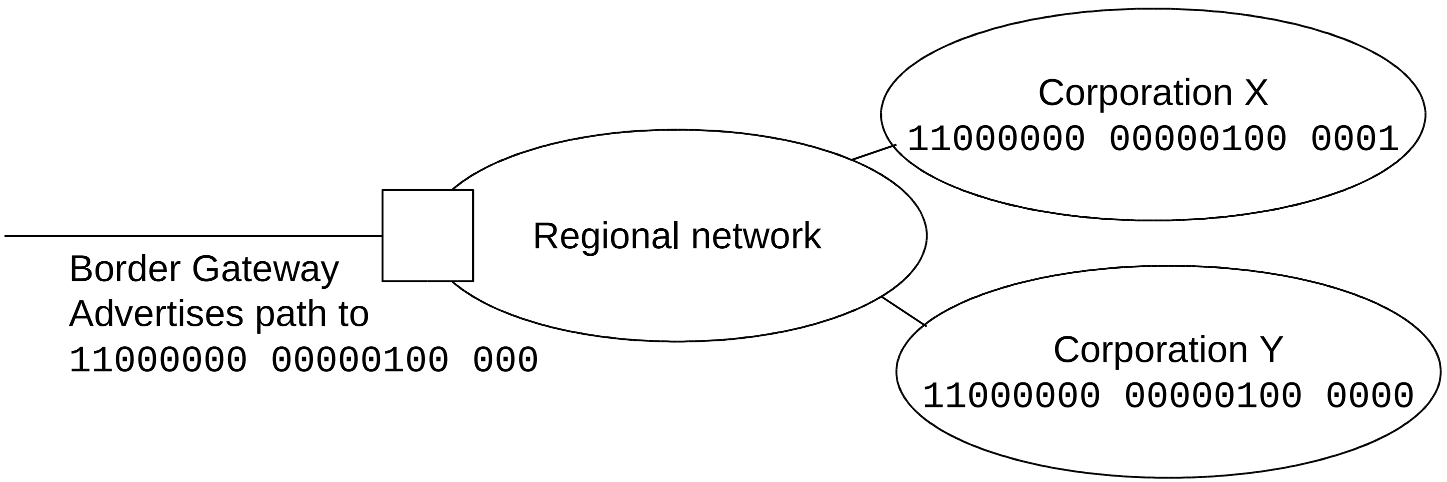
Campus Subnet mask: 255.255.0.0
Campus Subnet number: 129.27.0.0

Classless Routing

- Classless interdomain routing (CIDR) is addressing two scaling concerns:
    - The **grow of backbone routing tables**
    - **Increasing** the potential of 32-bit **IP addresses**
  - Suppose we assign class C network numbers from 192.4.16 through 192.4.31 the top 20 bits are always the same: 11000000.00000100.0001
  - We have created a 20-bit network number (between class B and class C) $\rightarrow$ high address efficiency and handling 256 nodes at a time !!!
  - We need to hand out blocks of class C addresses – the number of blocks is a power of 2
  - Therefore we need a routing protocol that can deal with classless addresses (since, a network number can be of any length)

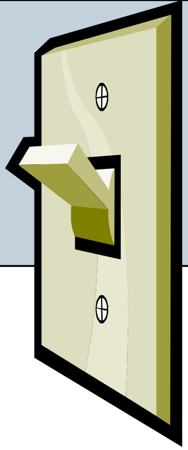
Classless Routing

- Network numbers in such routing protocols are represented by: <length, value>
- Network address are represented by: <mask, value>



- If there are overlapping prefixes in the forwarding table: 171.69 (16 bit) and 171.69.10 (24 bit) – a packet 171.69.10.5 matches both prefixes $\rightarrow$ a rule for this is based on *“longest match”*
- Finding the longest match between IP address and variable length prefixes: PATRICIA tree algorithm

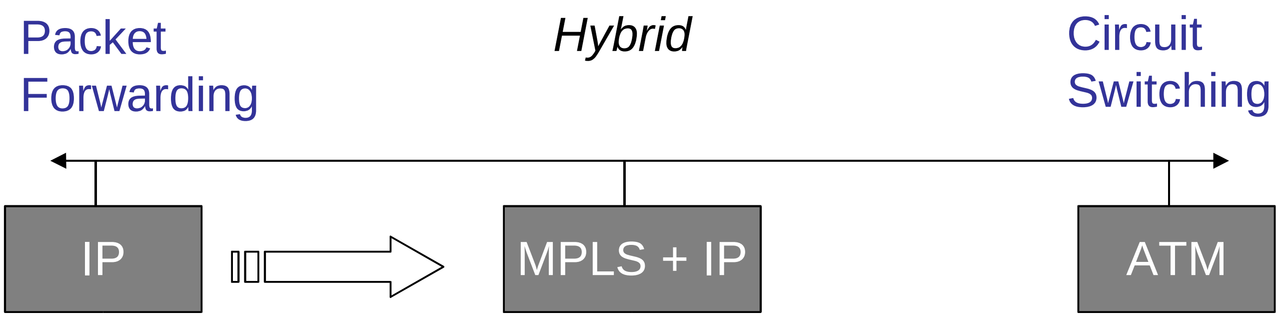
IP ATM and MPLS



- **IP**, the "only" protocol for global internet working
- … but there are other contenders: most notable **ATM**
- ATM could not displace other technologies in the local area network
- An interesting development in the relationship between IP and ATM is the appearance of **MPLS** (based on IP switching and Tag switching)
- "Merge" the forwarding algorithm used in ATM with IP control protocols
  - A Label switching router (LSR) forwards packets by looking at fixed length labels – using this label to find the output interface
  - LSR has to rewrite the label before it can send the packet
  - This is exactly how ATM switches are forwarding cells
    - Significantly simpler than longest-match
    - LSRs will be cheaper at a given performance (than conventional router)
    - Forwarding decisions can be based on more complex criteria
    - Supporting VPs to be merged into one single VP

IP ATM and MPLS

- MPLS + IP form a middle ground that combines the best of IP and the best of circuit switching technologies.
- ATM and Frame Relay cannot easily come to the middle so IP has!!



Problems of IP Routing

- **Hop-to-hop routing**
  - In connectionless IP, each router has to make independent forwarding decisions based on the IP header (32 bit v4, 128 bit v6)
  - But the header contains much more information than the router needs to find the next hop
- **Basically there are two routing functions:**
  - dividing the whole address space in *forward equivalence classes* (FEC)
    - e.g. *longest address prefix match*
  - allocate the FEC to the next hop

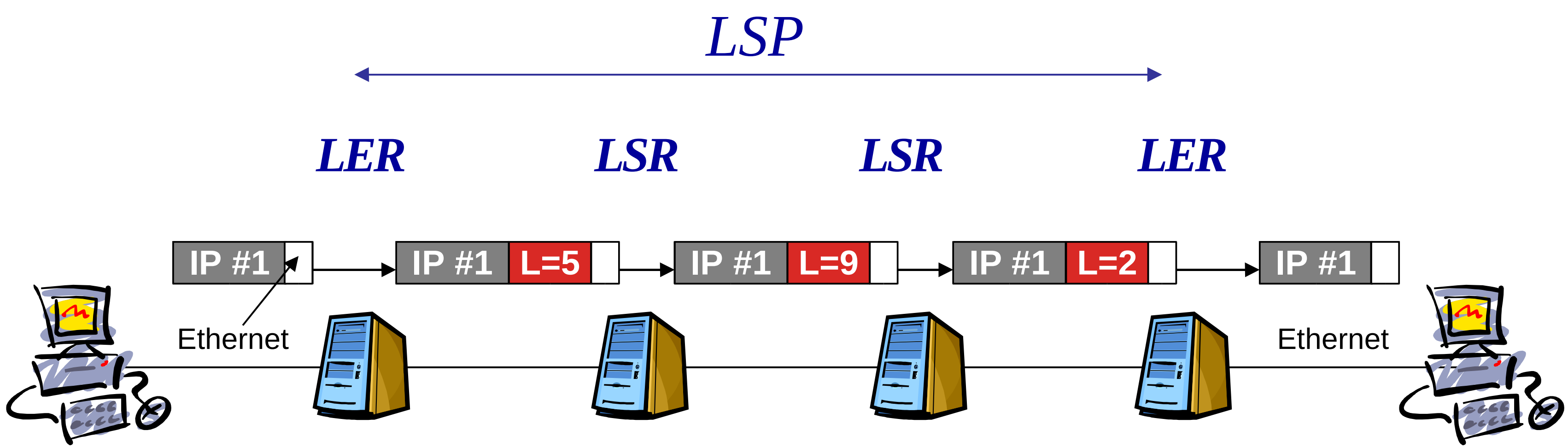
MPLS

- **MPLS performs the following functions:**
  - specifies mechanisms to manage traffic flows of various granularities, such as flows between different hardware, machines.
  - remains independent of the Layer-2 and Layer-3 protocols
  - provides a mechanism to map IP addresses to simple, fixed-length labels
  - interfaces to existing routing protocols (RSVP, OSPF)
  - supports the IP, ATM, and frame-relay Layer-2 protocols

MPLS

- data transmission occurs on label-switched paths (LSPs).
- The labels are distributed using label distribution protocol (LDP) or RSVP or piggybacked on routing protocols like border gateway protocol (BGP) and OSPF.
- Each data packet encapsulates and carries the labels during their journey from source to destination
- Hardware can be used to switch packets quickly between links.
- LSRs and LERs
  - An **LSR** is a high-speed router device in the core of an MPLS network
  - An **LER** is a device that operates at the edge of the access network and MPLS network - supports multiple ports connected to dissimilar networks (such as frame relay, ATM, and Ethernet)

Example



IP Addr	Out Label
192.4/16	5
Layer 2 Transport	**Assign init label**

In Label	Out Label
5	9
Label Swapping	

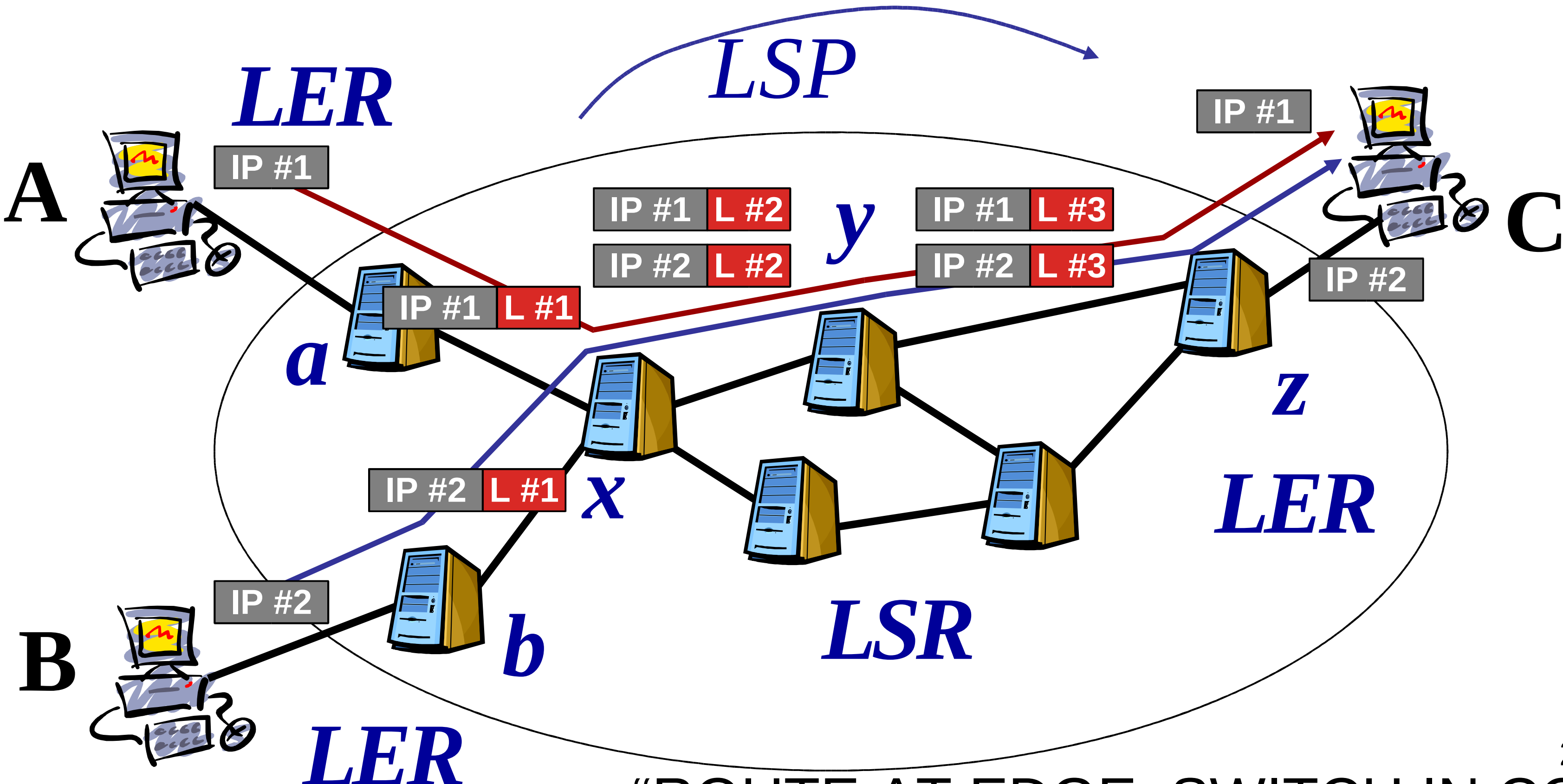
In Label	Out Label
9	2
Label Swapping	

In Label	Next Hop
2	192.4/16
Remove Label	Layer 2 Transport

“ROUTE AT EDGE, SWITCH IN CORE”

Example

- Communication A => C resp. B => C
  - mapped onto the same FEC at x, y, z



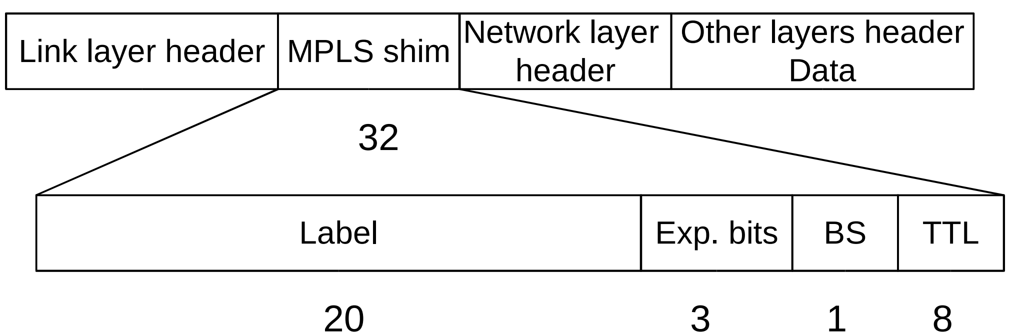
“ROUTE AT EDGE, SWITCH IN CORE”

MPLS

- **FEC**
  - The forward equivalence class (FEC) is a representation of a group of packets that share the same requirements for their transport.
  - the assignment of a particular packet to a particular FEC is done just once, <u>as the packet enters the network</u>
- **Labels and Label Bindings**
  - A label is carried or encapsulated in a Layer-2 header along with the packet
  - The label values are derived from the underlying data link layer.
  - The receiving router examines the packet for its label content to determine the next hop
  - Once a packet has been labelled, the rest of the journey of the packet through the backbone is based on label switching.
  - The label values are of local significance

MPLS

- Label assignment decisions may be based on forwarding criteria such as the following:
  - destination unicast routing
  - “traffic engineering”
  - multicast
  - virtual private network (VPN)
  - QoS
- Generic label format:



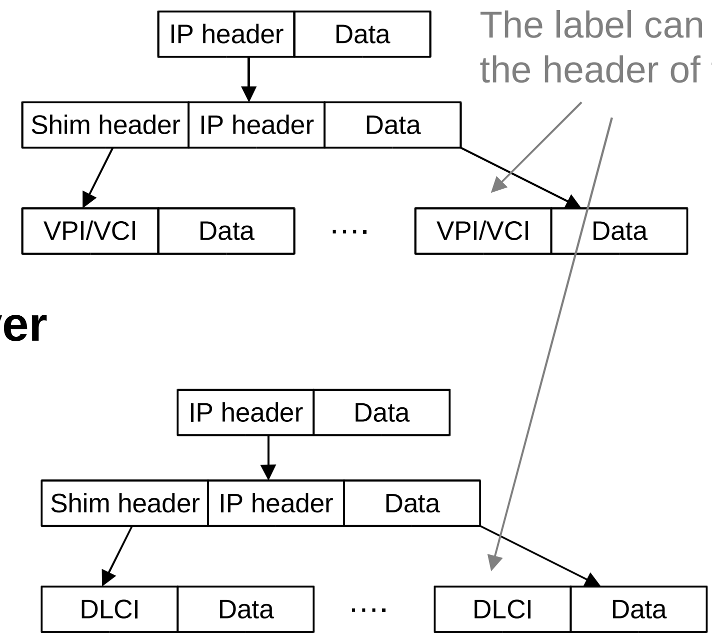
MPLS

ATM as the Data Link Layer

IP Packet

Labelling of the packet

ATM cells



The label can be embedded in the header of the data link layer…

Frame Relay as the Data Link Layer

IP Packet

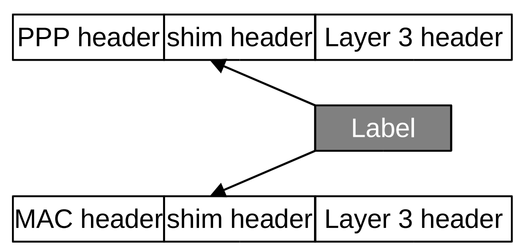
Labelling of the packet

FR frames

Point-to-Point (PPP)/Ethernet as the Data Link Layer

PPP header (Packet over Sonet/SDH)

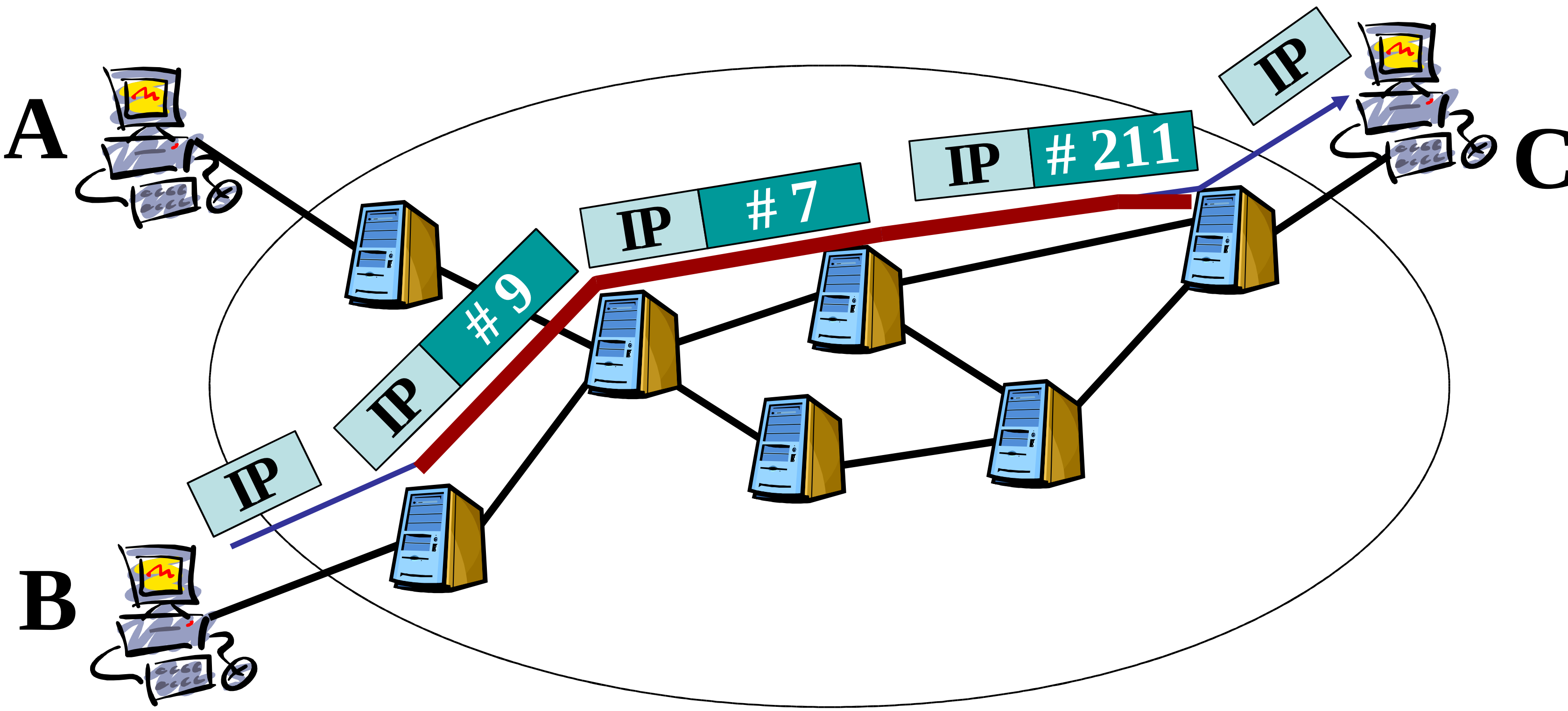
LAN MAC Header



… or in the shim (between the Layer-2 data-link header and Layer-3 network layer header,

Basic Idea (II)

- “route at edge”, and “switch in core“



MPLS

- **Label distribution:**
  - MPLS does not mandate a single method of signaling for label distribution
  - BGP has been enhanced to piggyback the label information within the contents of the protocol
  - RSVP has also been extended to support piggybacked exchange of labels.
  - IETF has also defined a new protocol known as the label distribution protocol (LDP) for explicit signaling and management
  - Extensions to the base LDP protocol have also been defined to support explicit routing based on QoS requirements.

MPLS

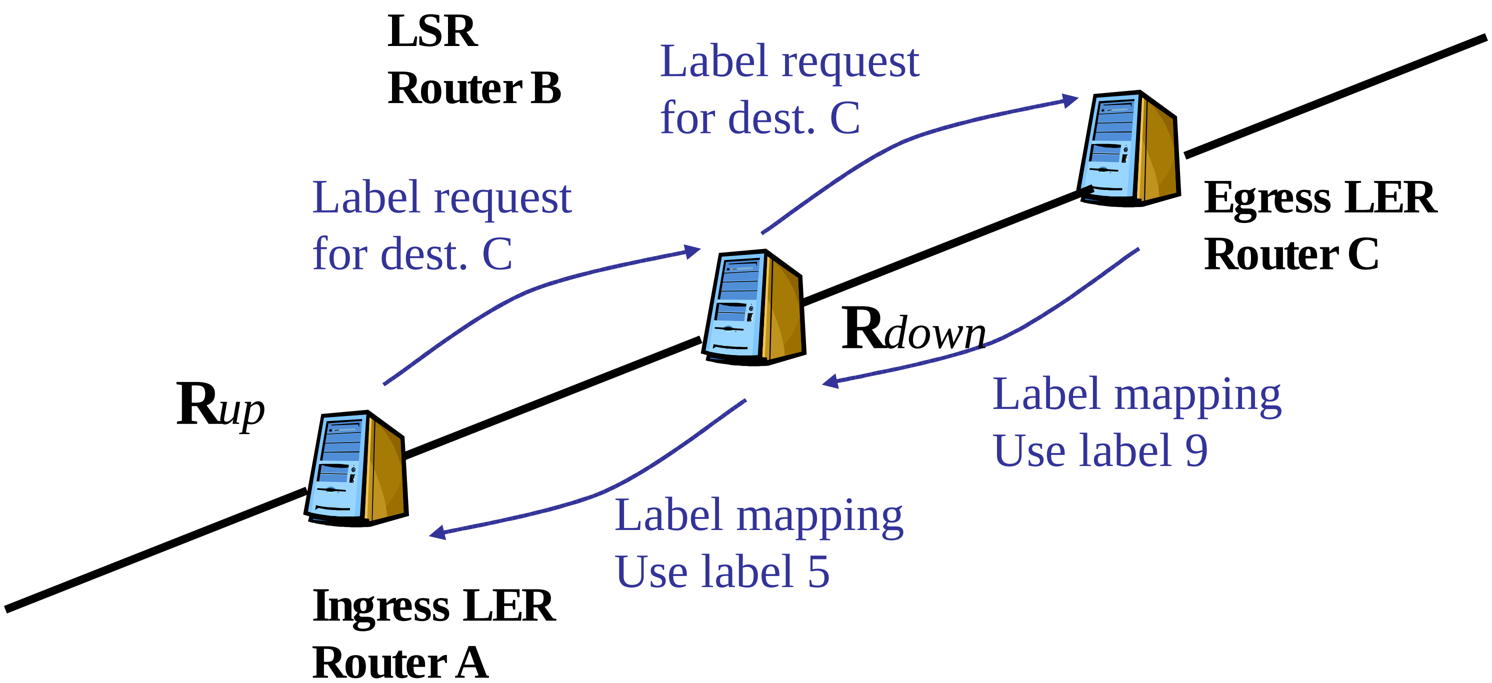
– **Label-Switched Paths (LSPs)**:
  - A collection of MPLS—enabled devices represents an MPLS domain. Within this domain, a path is set up for a given packet to travel on an FEC. The LSP is set up prior to data transmission. MPLS provides the following two options to set up an LSP:
    – **hop-by-hop routing:** Each LSR independently selects the next hop for a given FEC. The LSR uses any available routing protocols, such as OSPF, ATM private network-to-network interface (PNNI), etc.
    – **explicit routing:** is similar to source routing. The ingress LSR (i.e., the LSR where the data flow to the network first starts) specifies the list of nodes through which the ER–LSP traverses.

    The LSP setup for an FEC is unidirectional in nature. The return traffic must take another LSP.

MPLS

- **Label Merging:**
  - Incoming traffic from different interfaces can be merged together and switched using a common label if they are traversing the network toward the same destination (*stream merging* or *aggregation of flows*)
  - If the underlying transport network is ATM, LSRs could employ VP- or VC-merging. (avoid cell interleaving problems)
- **Signaling Mechanisms**
  - **label request:** A LSR requests a label from its downstream neighbor so that it can bind to a specific FEC. This mechanism can be employed down the chain of LSRs up to the egress LER
  - **label mapping**: In response to a label request, a downstream LSR will send a label to the upstream initiator using the label mapping mechanism

Upstream vs. downstream



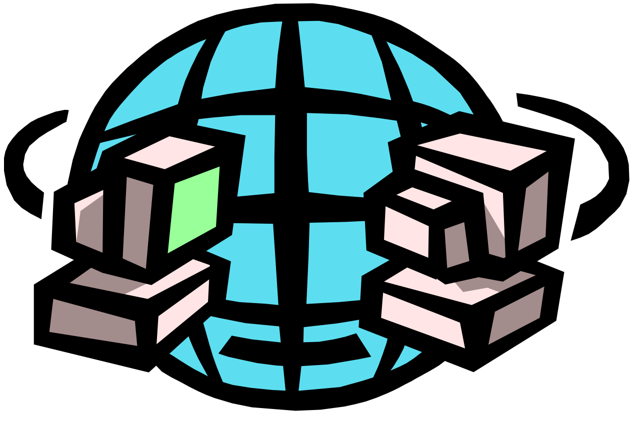
MPLS

- **Label Distribution Protocol (LDP)**
  - The LDP is a new protocol for the distribution of label binding information to LSRs in an MPLS network. It is used to map FECs to labels, which, in turn, create LSPs. LDP sessions are established between LDP peers in the MPLS network (not necessarily adjacent). The peers exchange the following types of LDP messages:
    - discovery messages— announce and maintain the presence of an LSR in a network
    - session messages— establish, maintain, and terminate sessions between LDP peers
    - advertisement messages— create, change, and delete label mappings for FECs
    - notification messages— provide advisory information and signal error information

MPLS Applications

- Enabling IP over ATM
  - The LER devices are responsible for IP flow classification and label imposition.
  - The LSR devices located in the core are responsible for forwarding at Layer 2 while participating in the exchange of Layer 3 routing information.
- Traffic Engineering
  - Best effort delivery is not (always) sufficient.
  - MPLS provides for explicit routing
- Class of Services
  - The head end LSR could place high-priority traffic in one LSP, medium-priority traffic in another LSP, best-effort traffic in a third LSP, and less-than-best-effort traffic in a fourth LSP.
- VPN´s
  - Each ingress LSR places traffic into LSPs based on a combination of a packet's destination address and VPN membership information

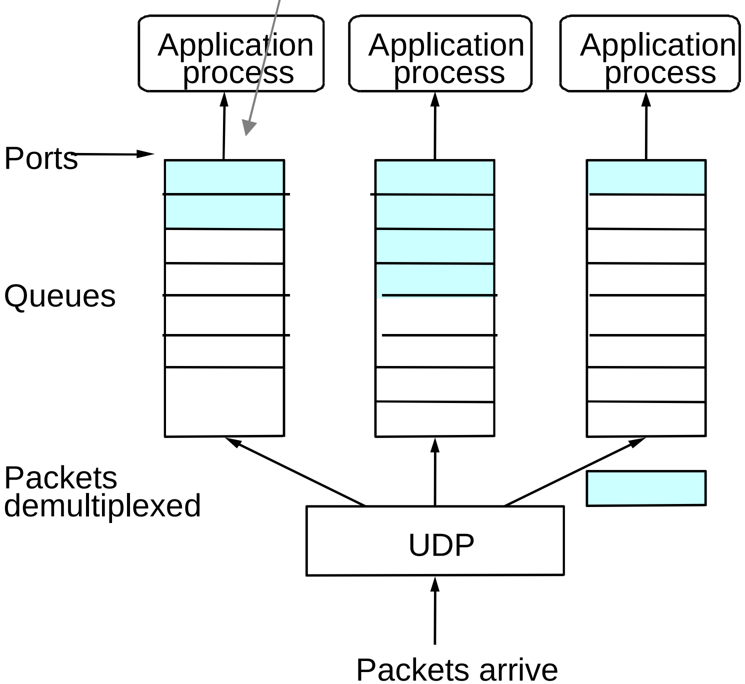
End-to-end Protocols
(Chapter 5)



- 5.1 UDP
- 5.2 TCP

? 5.1 User Datagram Protocol (UDP)

port := abstraction, depending on the OS



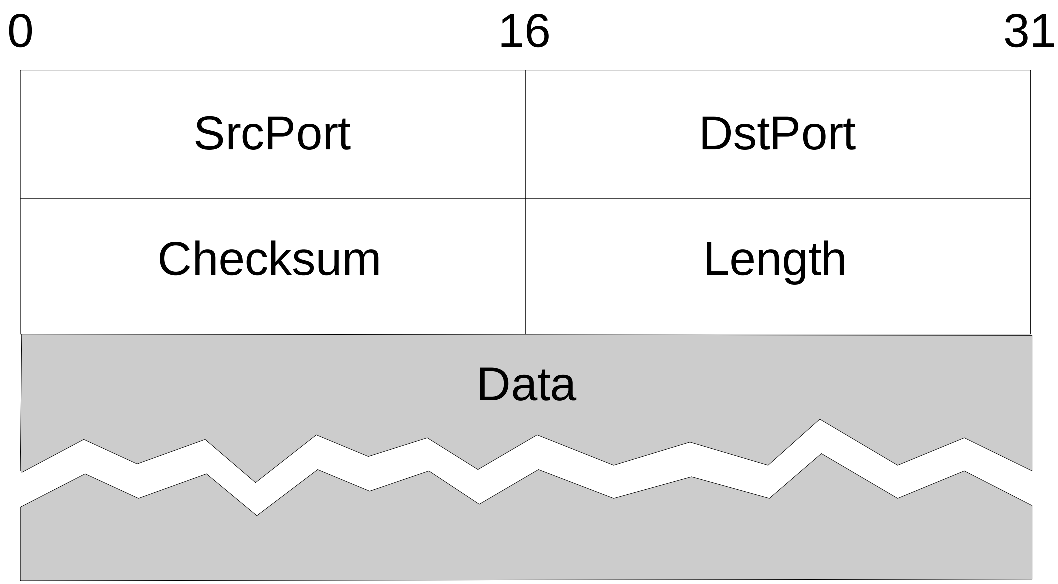
- connectionless, datagrams
  - no guarantees for message delivery
  - no guarantee for right order
  - duplicated datagrams
  - error correction incl. „Pseudo-header“ *(three fields from IP header —protocol number, source IP, destination IP— plus UDP length field)*
  - no flow control mechanism
- Ports can be used to distinguish different services
- Port & IP-Address = SOCK_DGRAM

UDP/Pseudo header

- UDP header
  - Source port (2 Byte)
  - Destination port (2 Byte)
  - Length, Checksum (4 Byte)
- Checksum*) is calculated over the UDP header, the content of the message body, and something called the "Pseudo header"
  - Source IP address
  - Destination IP address
  - Protocol
  - UDP length (included twice in the checksum calculation)

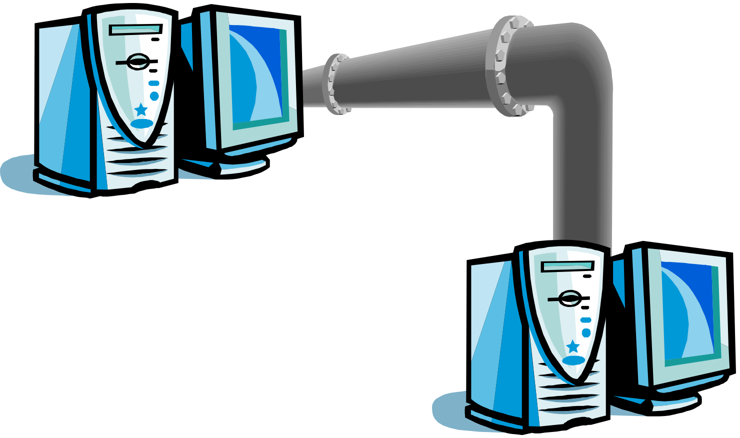
UDP checksum: optional in IP4 mandatory in IP6

UDP header



?

5.2 Transmission Control Protocol (TCP)



- Header format
- Connection establishment
- Sliding Window Algorithm

TCP

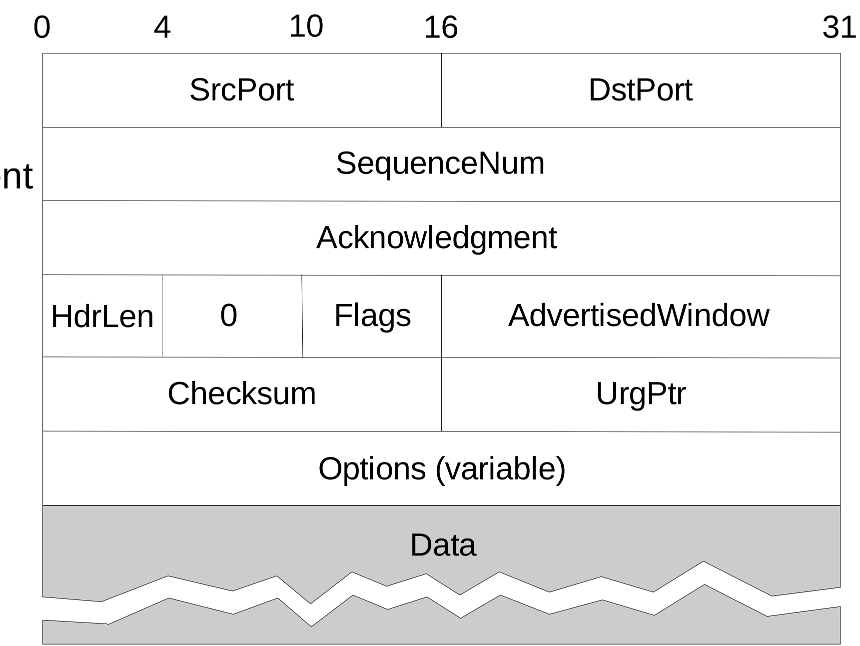
- The heart of TCP is the "Sliding Window" algorithm
- Explicit *connection establishment/teardown* phase
- Different RTT´s (even during a single connection)
  - SW-timeout mechanism must be adaptive
- Reordered packets on the link (but not in PPP!!)
- TCP uses SW on an end-to-end base
- X25 within the network on a hop-by-hop base
  - Hop-to-hop does not (necessarily) add up to an end-to-end guarantee (SW A-B and B-C does not guarantee that B behaves perfectly – therefore it is necessary to provide true end-to-end checks to guarantee reliable/ordered services)

TCP Segment Format

- TCP is byte oriented (sending/receiving “byte streams”)
- TCP buffers bytes to fill a reasonably sized packet – and then sends this packet to its destination host
- Packets are called segments *(-of a byte stream)*
- 3 mechanism to trigger the transmission:
  - Send as soon as maximum segment size (MSS) is reached (MSS = MTU – header, MTU of the local network)
  - Sending process has ask to do so (push operation – e.g. Telnet)
  - Periodically firing trigger
- Ports are used to distinguish between services
- Port & IP-Adresse = SOCK_STREAM

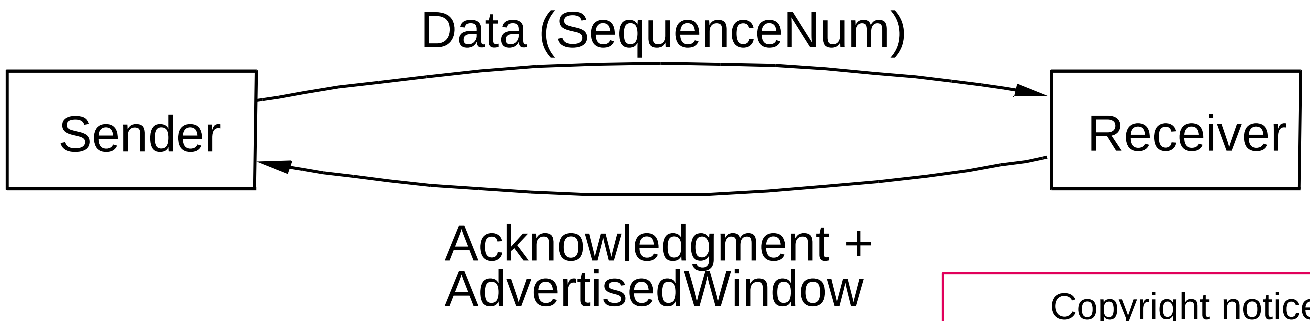
TCP header

- Source port, destination port (+ IP = TCP demux-key) (4 Byte)
- Sequence number (SW algorithm) (4 Byte)
- Acknowledgement number (SW algorithm) (4 Byte)
- Data offset, reserved (2 Byte)
- Flags
  - URG, ACK, PSH, RST, SYN, FIN
- Advertised Window (SW algorithm) (2 Byte)
- Checksum (computed over the TCP header) (2 Byte)
- UrgPtr (if Flag=Urg, this is a pointer to nonurgent data)
- Options
  - MSS during connection establishment

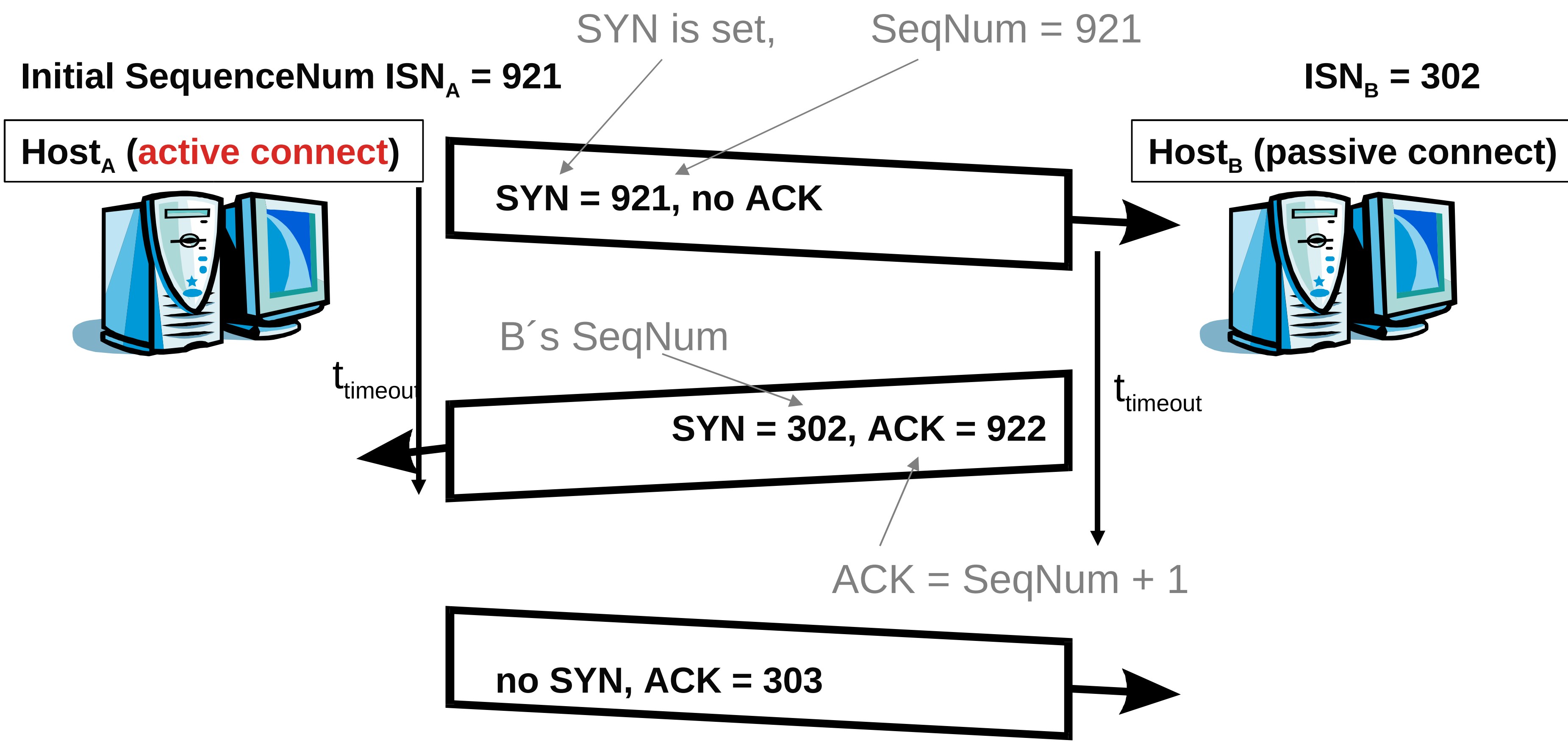


TCP Principle

- SYN FIN flags to establish and terminate a TCP connection
- ACK is set, if Acknowledgement Field is valid
- If URG is set, the segment contains urgent information – the user level process has to be informed about this fact
- If PUSH is set, the receiving process has to be informed about this fact
- If RESET is set, the receiver became confused, and therefore wants to close the connection
- SequenceNum: number of already sent byes
- Acknowledgement: “the number of the expected byte”

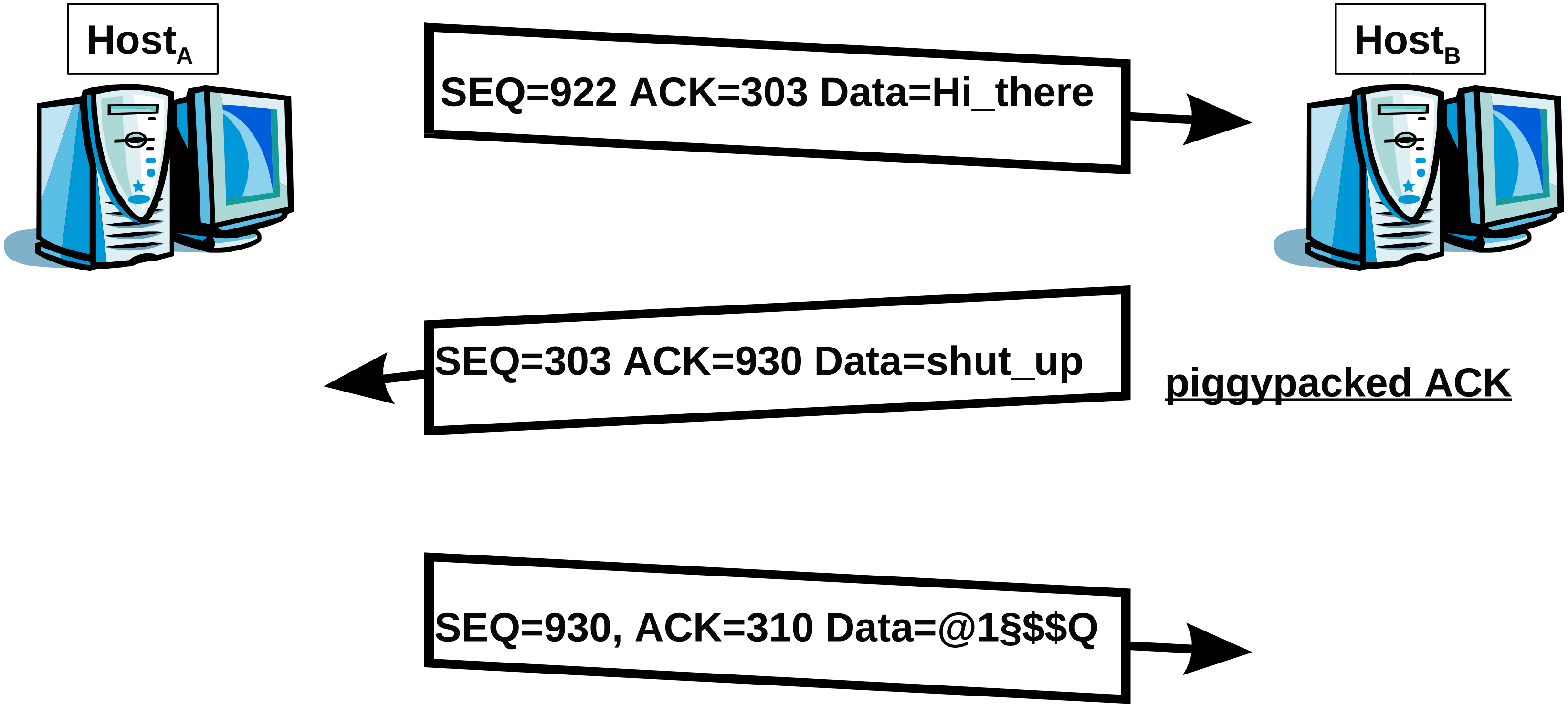


TCP *Connection Establishment*

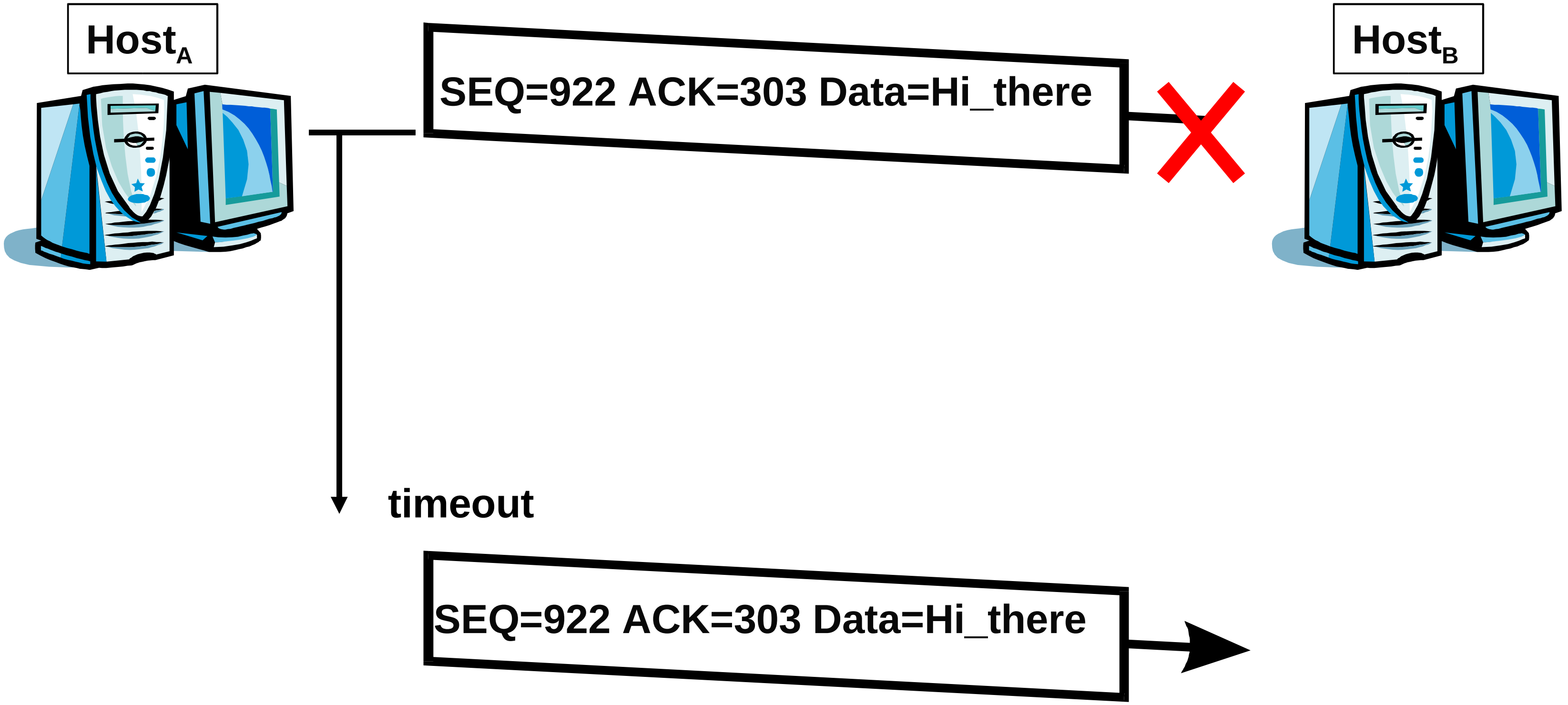


Setup is an asymmetric activity

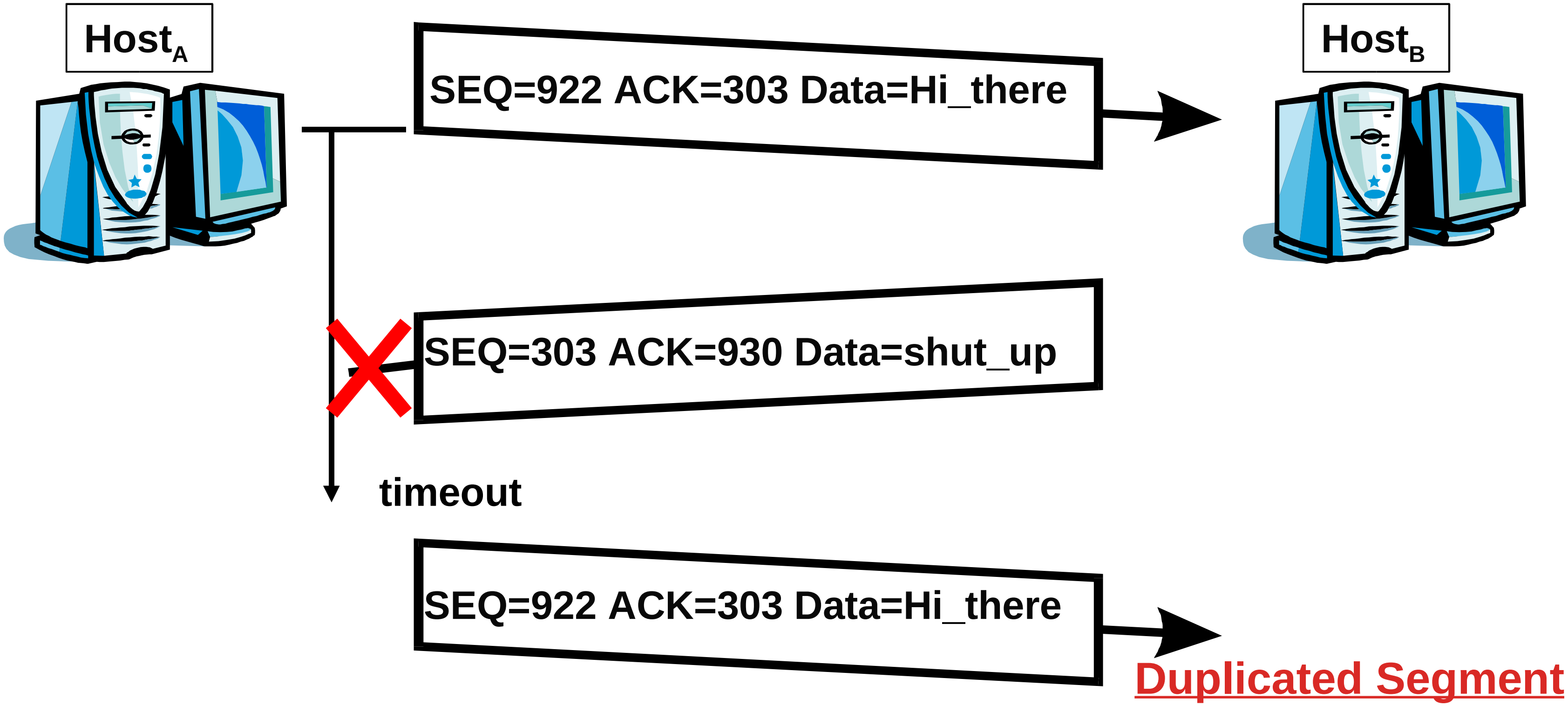
TCP *Data Transmission*



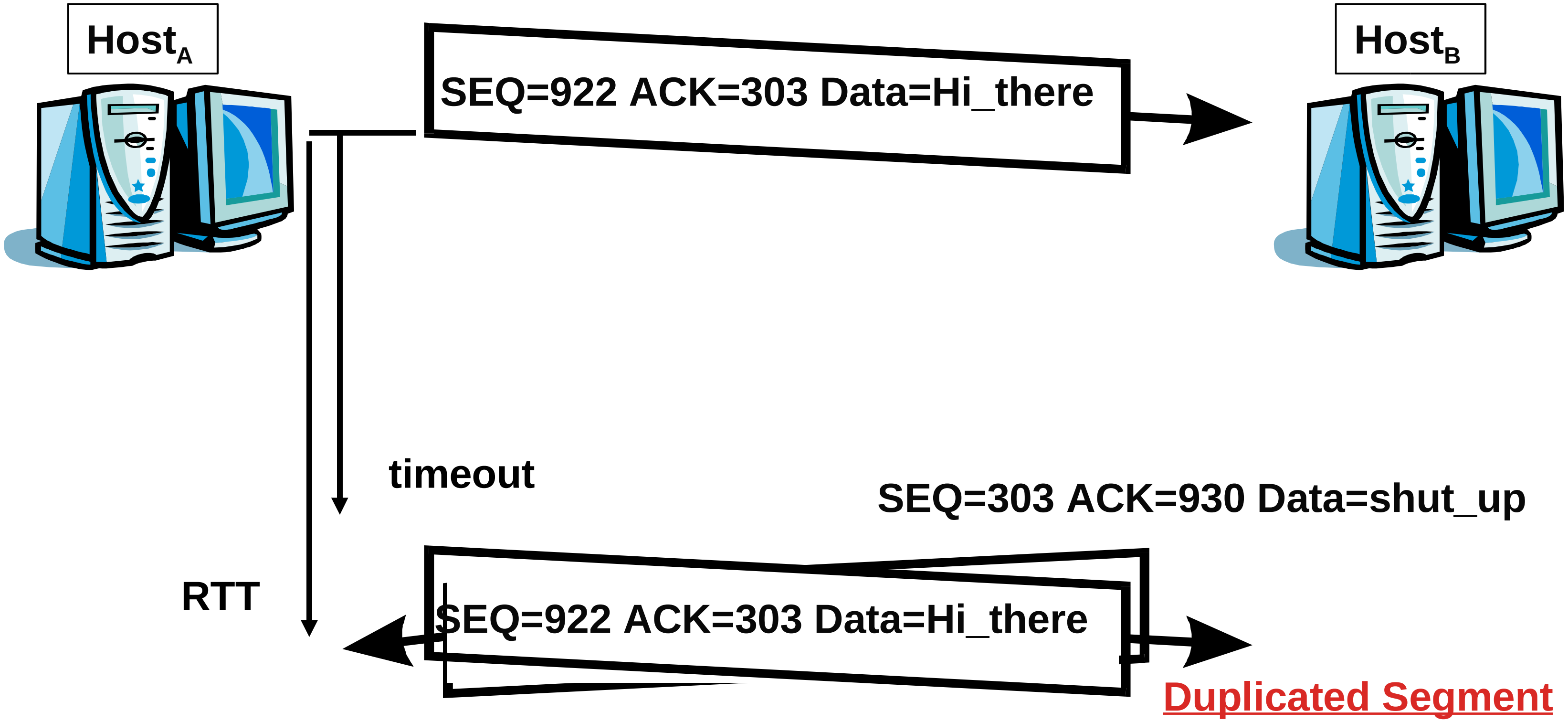
TCP *Lost Segment*



TCP *Lost ACK*


HostA
HostB
SEQ=922 ACK=303 Data=Hi_there
SEQ=303 ACK=930 Data=shut_up
timeout
SEQ=922 ACK=303 Data=Hi_there
Duplicated Segment

TCP Timeout < RTT



TCP char. echo (Nagle)

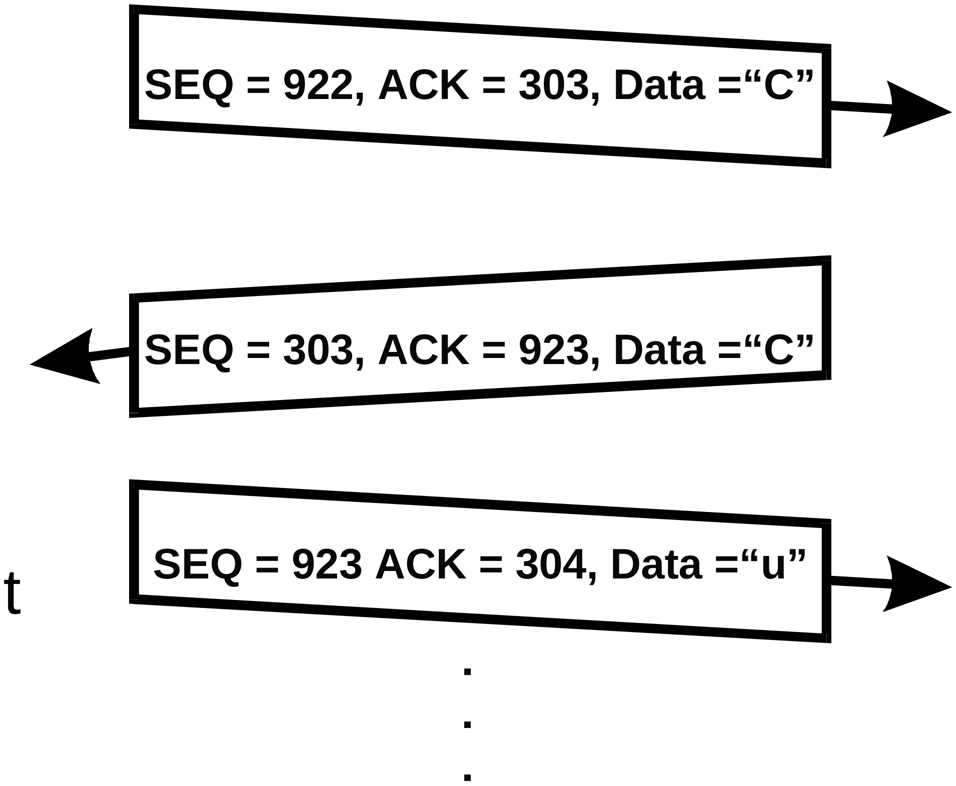
TCP is inefficient at small payload

Example: Telnet

n Byte link-layer overhead
20 Byte IP header
20 Byte TCP header
1 Byte TCP payload

Nagle: Increasing the efficiency by introducing small segments

Rule: Just „small“ can be transmitted without acknowledgement



TCP State Transmission Diagram

- All connections starts in the CLOSED state
- Each arc (state transition) is labeled with (*event/action*)
- Connection can be switched into the LISTEN state
- Now 2 kinds of events can trigger a state transition
  - A segment arrives from the peer *(SYN/SYN+ACK)*
  - Local application invokes an operation *(Send/SYN)*

  The TCP state transition diagram defines the semantics of peer-to-peer and service interfaces
- Typical transition:
  - When opening a connection, the server invokes a *passive open* $\rightarrow$ goto LISTEN
  - Client does an active open (*Active open/SYN)* $\rightarrow$ goto SNY_SENT
  - When SYN arrives at the server $\rightarrow$ goto SNY_RECEIVED and send an ACK (SYN/SYN+ACK)

TCP State Transmission Diagram

- The arrival of this segment causes the client to go to ESTABLISHED and sends an ACK *(SYN+ACK/ACK)*
- The arrival of ACK a the server causes the server to change to ESTABLISHED as well (3-way handshake)

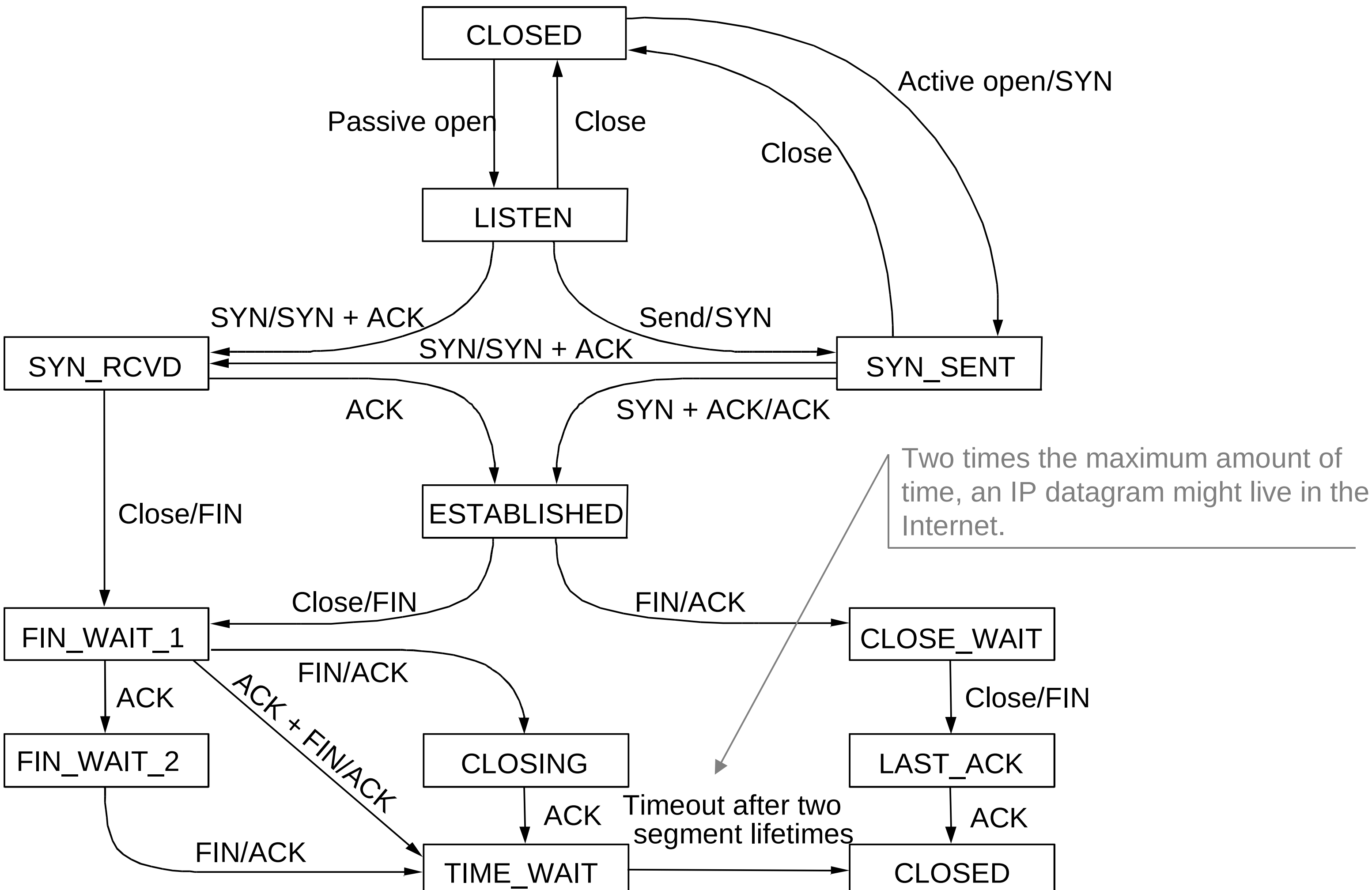
- Note1: if the clients ACK is lost, the client side is already in the ESTABLISHED state!!! $\rightarrow$ the local application can start to send. Each of this segments will have ACK set and a correct Acknowledgement # set $\rightarrow$ the server will move to ESTABLISHED
- Note2: the local process can invoke SEND out of LISTEN (*Send/SYN*)
- Note3: most of the transitions are scheduling a timeout – causing segments to be resent (if the response does not happen) – these transitions are not shown in the diagram!

TCP State Transmission Diagram

– There are 3 of combinations to get a connection from ESTABLISHED to CLOSED

- *This side closes first:* ESTABLISHED → FIN_WAIT_1 → FIN_WAIT_2 → TIME_WAIT → CLOSED
- *The other side closes first:* ESTABLISHED → CLOSE_WAIT → LAST_ACK → CLOSED
- *Both sides are closing at the same time:* ESTABLISHED → FIN_WAIT_1 → CLOSING → TIME_WAIT CLOSED

State-transition diagram



? TCP Sliding Window Protocol

- TCP uses the sliding window protocol as discussed in section 2
- …and it folds a flow control-function in as well
- TCP does not use a fixed size window – the receiver advertises a window size to the sender (AdvertisedWindow field) based on the amount of memory allocated to the connection
- Flow Control
  - Send- and receive-buffer are of some finite size (MaxSend-MaxRcvBuffer)
  - The window size sets the amount of data that can be sent without receiving an ACK, and ..

  LastByteRcvd - LastByteRead $\leq$ MaxRcvBuffer therefore …

  AdvertisedWindow = MaxRcvBuffer – (LastByteRcvd - LastByteRead)

TCP Sliding Window Protocol

- TCP on the sender side must then adher to the advertised window it gets from the receiver. At any given time it must ensure that …

LastByteSent – LastByteAcked ≤ AdvertisedWindow

- … the sender computes an effective window that limits how much data it can send:

EffectiveWindow = AdvertisedWindow – (LastByteSent – LastByteAcked)

- When this is going on, the sender has to make sure, that the process does not overflow the send buffer:

(LastByteWritten – LastByteAcked) ≤ MaxSenderBuffer

- If the sender process tries to write y bytes to TCP, but

(LastByteWritten – LastByteAcked)+ y > MaxSenderBuffer

- … then TCP blocks the sending process.

TCP Sliding Window Protocol

- Slow receiving process can stop fast sending processes
  - The receiving buffer fills up → advertise windows shrinks to 0

    → Send buffer fills up , which causes TCP to stop the sending process
  - As soon as the receiving process starts to read again – the receive side opens its window again which allows the send-side TCP to send again out of its buffer.
  - When this data is acknowledged, LastByteAcked is incremented, the buffer holding this acknowledged data is freed and the sending process is unblocked.
- Protection against wraparound
  - SequenceNum field is 32 bit long and …
  - AdvertisedWindow field is 16 bit long → TCP easily satisfies the requirement of "sliding window" $2^{32} >> 2 \times 2^{16}$

TCP Sliding Window Protocol

- But we have to make sure, that the sequence number does not wrap around within a 120-second period of time

T1 (1.5Mbps)	6.4 hours
Ethernet (10Mbps)	57 minutes
T3 (45Mbps)	13 minutes
FDDI (100Mbps)	6 minutes
STS-3 (155Mbps)	4 minutes
STS-12 (622Mbps)	55 seconds
STS-24 (1.2Gbps)	28 seconds

- Fortunately, the IETF has already worked out an extension to TCP to extend the sequence number space to protect against wrap around

TCP Sliding Window Protocol

- Keeping the pipe full …
  - Advertised window must be big enough to allow the sender to keep the pipe full
  - Since, it is the delay x bandwidth product to that dictates how big the advertised window needs to be
  - Assume: RTT=100ms …

T1 (1.5Mbps)	18kB
Ethernet (10Mbps)	122kB
T3 (45Mbps)	549kB
FDDI (100Mbps)	1.2MB
STS-3 (155Mbps)	1.8 MB
STS-12 (622Mbps)	7.4 MB
STS-24 (1.2Gbps)	14.8 MB

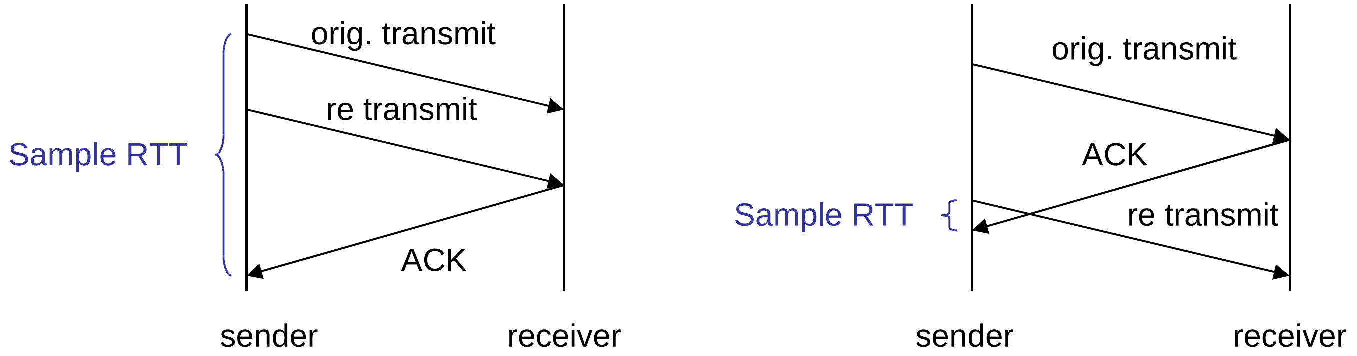
TCP´s advertise window field is not big enough to handle even a Ethernet connection (16b $\rightarrow$ 65.536)

? TCP Adaptive Retransmission

- Since there are …
  - different RTT´s between any pair of hosts
  - variations in RTT between the same two hosts
- TCP uses an adaptive retransmission mechanism
- Original mechanism:
  - Every time TCP sends a data segment, it records the time
  - When an ACK arrives, TCP reads the time again and calculates a SampleRTT
  - TCP calculates a weighted average between all samples:
  - EstimatedRTT = α x EstimatedRTT + (1- α) x SampleRTT
  - $0.8 \leq \alpha \leq 0.9$ and TimeOut = 2 x EstimatedRTT

TCP Adaptive Retransmission

- Karn/Partridge Algorithm



  - The solution is simple:
    - Stop taking RTT samples after a TCP retransmit
    - Another small change to TCP´s timeout mechanism:
      - After each retransmit, it sets the next timeout to be twice the last timeout (exponential backoff – similar to Ethernet) since congestion is the most likely cause for packet loss

TCP Adaptive Retransmission

- Jakobson/Karels Algorithm
  - The Karn/Partridge algorithm does not really solve the problem of congestion, since it does not take the variance of the RTTs into account
  - If variation is small, EstimatedRTT can be better trusted (no reason for multiplying by 2)
  - If variation is large $\rightarrow$ do not couple the timeout too tightly to EstimatedRTT

Difference = SampleRTT – EstimatedRTT

EstimatedRTT = EstimatedRTT + (δ x Difference) $0 \leq \delta \leq 1$

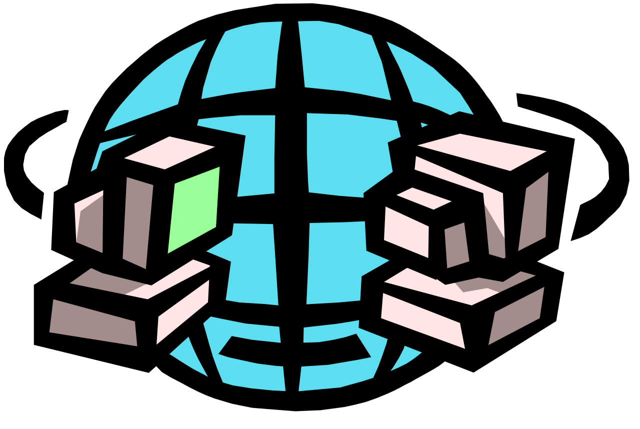
Deviation = Deviation + δ (|Difference| - Deviation)

Timeout = μ x EstimatedRTT + Φ x Deviation $\mu = 1$; $\Phi = 4$

<u>If the variance is small $\rightarrow$ Timeout is close to EstimatedRTT</u>

Congestion Control and Resource Allocation

(Chapter 6)



- TCP Congestion Control
- Congestion Avoidance Mechanism
- Quality of services

? TCP Congestion Control

- Additive Increase / Multiplicative Decrease
  - TCP maintains a state variable for each connection (CongestionWindow)
  - CongestionWindow is the congestion control counterpart to the flow control advertise window.
  - Maximum number of unacknowledged data is now the minimum of CongestionWindow and AdvertisedWindow:

    MaxWindow = MIN(CongestionWindow, AdvertisedWindow)

    EffectiveWindow = MaxWindow – (LastByteSent - LastByteAcked)
  - A TCP source is allowed to send not faster than the slowest component (the *network* or the *destination host*).
  - But how does TCP learn an appropriate value for CongestionWindow:
    - TCP interprets timeouts as a sign of congestion

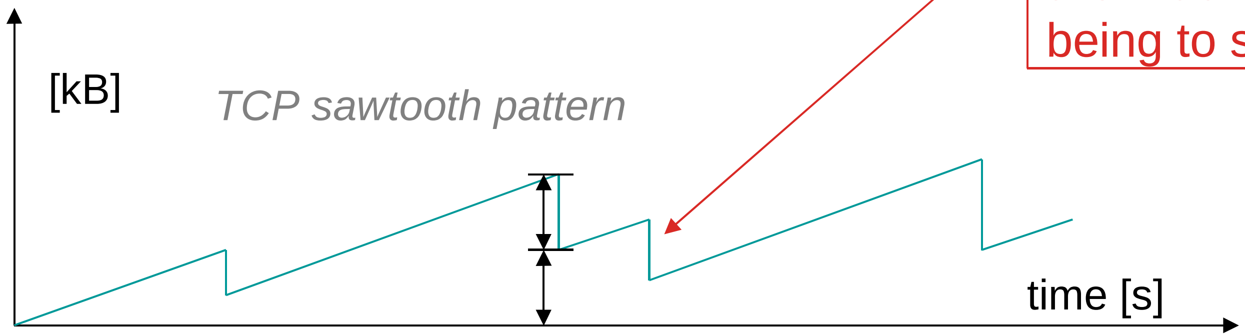
TCP Congestion Control

- Each time a timeout occurs, the source sets CongestionWindow to half of its previous value (multiplicative decrease)
- Every time the source "successfully" sends a CongestionWindow´s worth of packets (each packet has been ACK´ed) it adds 1 packets worth to CongestionWindow.
- TCP does not wait for a for an entire window´s worth of ACKs to add 1 packet – TCP increments instead CongestionWindow by a little for each arriving ACK:

  Increment = MSS x (MSS/CongestionWindow)

  CongestionWindow += Increment

To large windows are much worse than being to small



TCP Congestion Control

- Slow Start
  - Additive increase takes too long to ramp up a connection (when it is starting from scratch)
  - Therefore TCP provides “Slow Start” (*increase CongestionWindow rapidly from a cold start*)
  - Send 2 windows for each ACK (double the number of packets)
  - “Slow Start” is an exponentially mechanism – but is slow, compared with the original behavior of TCP (send as many packets as AdvertiseWindow allows)
  - 2 situations:
    1. At the beginning of a connection (the source has no idea how many packets to send $\rightarrow$ double CongestionWindow each RTT, until a loss, at which CongestionWindow is divided by 2)
    2. On packet-loss, the source may have sent a whole window and is waiting for an ACK – eventually, a timeout happens – the source will receive a single cumulative ACK, that would reopen the entire window

TCP Congestion Control

The source then uses “slow start” to restart. It now knows more information - the current CongestionWindow which is divided by 2 as a result of loss

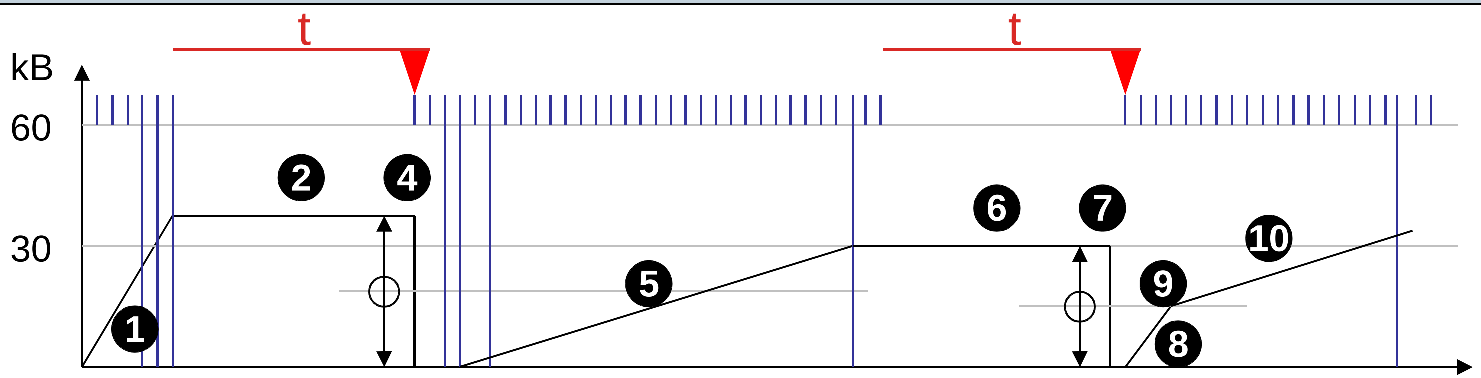
“Slow Start” is used to reach this “target” congestion window very soon, and the additive increase it, beyond this point.

To store this value, we need an additional variable “CongestionTreshold”:

```
{
    u_int           cw = state->CongestionWindow;
    u_int           incr = state->maxseg;

    if (cw > state->CongestionTreshold)
         incr = incr * incr / cw;
    state->CongestionWindow = MIN(cw + incr, TCP_MAXWIN)
}
```

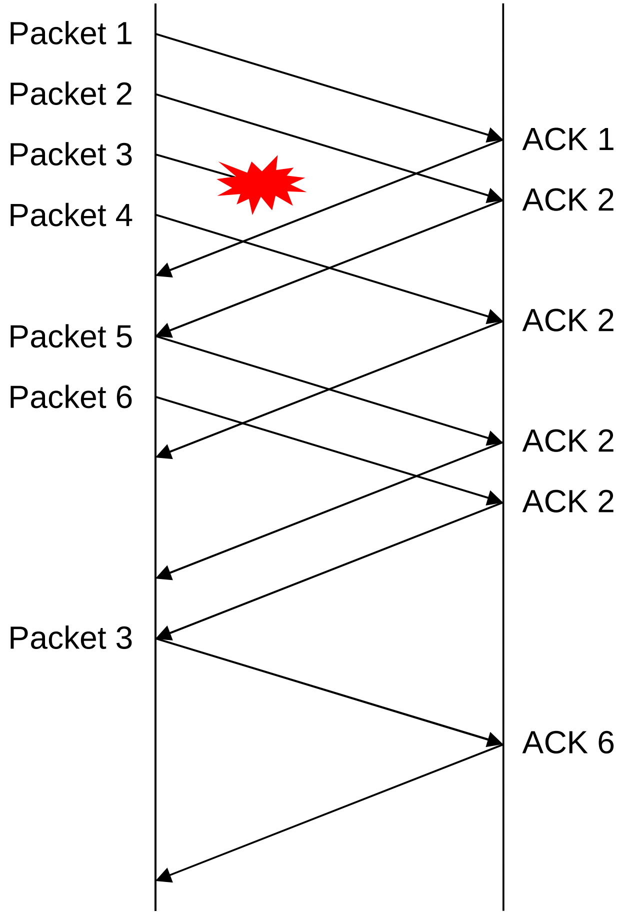
TCP Congestion Control



1. Initial “Slow start” phase - continues until several packets are lost
2. Congestion window flattens out (no ACK´s are arriving)
3. No new packets are sent during “2”
4. A timeout happens at “4” → CongestionWindow is devided by two and CongestionTreshold is set to this value
5. Slow start causes CongestionWindow to be set to one and start ramping up from there “additive increase”
6. CongestionWindow flattens out again - due to a packet loss
7. A timeout happens → CongestionWindow is divided by two and CongestionTreshold is set to this value
8. CongestionWindow is set to one → start “slow start”
9. CongestionWindow grows exponentially until it reaches CongestionTreshold
10. CongestionWindow then grows linearly

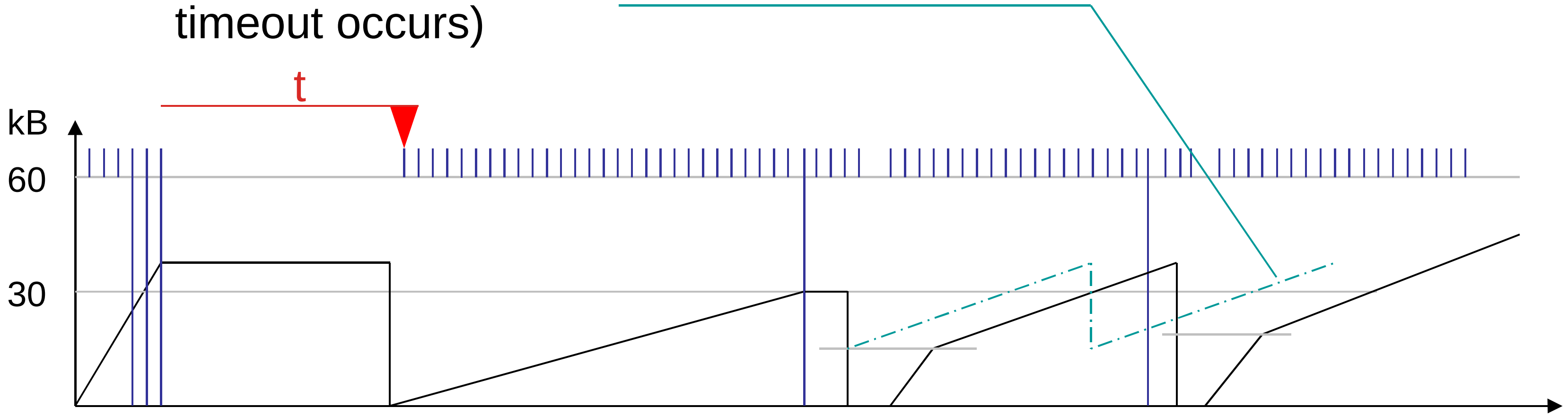
? TCP Congestion Control

- Fast Retransmit and Fast Recovery
  - It was discovered that the course-grained implementation of timeouts led to long periods during the congestion is waiting for a timer to expire → "Fast retransmit" was added.
  - Packet 3 is lost in the network.
  - Destination will send a duplicated ACK for packet 2 when 4 arrives
  - ... again when 5 arrives
  - When the sender sees three duplicated ACK´s – it retransmits 3
  - The receiver sends a cumulative ACK for everything up to packet 6



TCP Congestion Control

- Fast Retransmit and Fast Recovery
  - The long periods during which the congestion window stays flat (and no packets are sent) has been eliminated
  - 20% improvement in the throughput
  - “Fast Retransmit” can detect up to 3 dropped packets per window
  - When “Fast Retransmission” signals “congestion” (rather than drop CongestionWindow to 1 and run “slow start”) it is possible to use the ACK´s that are still in the pipe to clock the sending of packets. This is called “fast recovery” and removes “slow start” (if no coarse-grained timeout occurs)



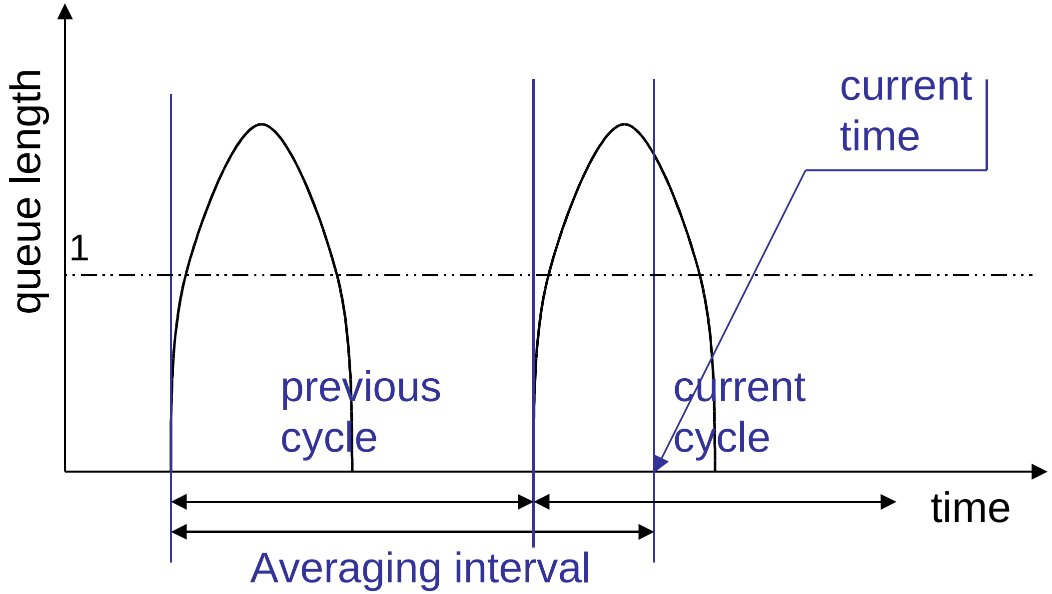
TCP Congestion Avoidance

- TCP controls congestion once it happens
- ... find the point at which congestion occurs, and then it backs off from this point
- Predict when congestion is about to happen – and then reduce the rate at which hosts send data just before packets being discarded → Congestion Avoidance
- **DECbit**
  - Developed for use on the Digital Network Architecture (DNA) (a connectionless network with a connection oriented transport protocol)
  - Idea: split the responsibility for congestion control between routers and nodes
  - Each router monitors the load and notifies the end nodes when congestion is about to occur (set a binary congestion bit DECbit).
  - Destination host copies the DECbit into the ACK it sends back to the source

? TCP Congestion Avoidance

– The source adjusts its sending rate to avoid congestion

1. A single congestion bit is added to the header
2. Router sets this bit, if its average queue length ≥ 1 at the time, the packet arrives (a queue length of 1 seems to optimize the power function)
3. The source records how many of its packets resulted in some router setting the congestion bit
4. If < 50% of all packets had the bit set → source increases CongestionWindow by one. If $\geq$50% had the congestion bit set → decrease the CW to 0.875 times the previous value



? TCP Congestion Avoidance

- **Random Early Detection (RED)**
  - Router is programmed to monitor its own queue length → notify the source in case of congestion → adjusting source CongestionWindow
  - RED differs from DECbit in two ways:
    1. RED “implicitly” notifies the source by dropping one of its packets (the source is notified by duplicated ACK´s or timeouts). The router drops the packet “earlier” than it would have to.
    2. “early random drop” – drop packets on some “drop probability” whenever the queue length exceeds some “drop level”
  - RED computes an weighted average queue length:

    AvgLen = (1-Weight) x AvgLen + Weight x SampleLen   0<Weight<1
  - RED has two queue length tresholds to trigger activities:

    MinTreshold, MaxTreshold

TCP Congestion Avoidance

- When a packet arrives at the gateway, RED compares the current AvgLen with MinTreshold and MaxTreshold

If AvgLen $\leq$ MinTreshold

→ queue the packet

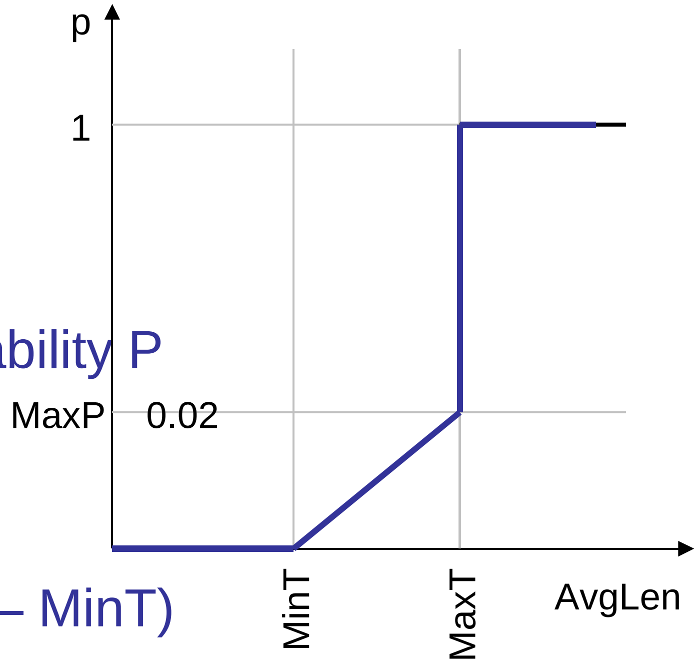
If MinTreshold < AvgLen < MaxTreshold

→ calculate propability P

→ drop the arriving packet with probability P

If MaxTrshold $\leq$ AvgLen

→ drop the arriving packet



TempP = MaxP x (AvgLen-MinT)/(MaxT – MinT)

P = TempP/(1 – count x TempP)

count: how many new packets have been queued

This algorithm ensures a roughly even distribution of drops in time

TCP Congestion Avoidance

RED vs. ATM

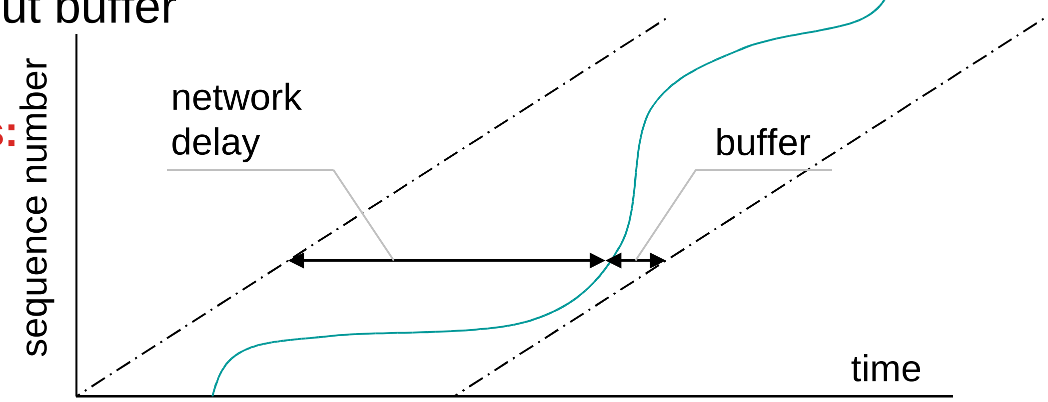
- If we are sending AAL5 packets through a congested ATM switch, and the switch has to drop one of these cells $\rightarrow$ the other cells will be useless (Partial Packet Discard, PPD)

- A switch can be made more aggressive by combining RED with PPD
  - When an ATM switch is nearing congestion and the first AAL cell arrives, the switch drops that cell and all the others
  - This enables the whole packet to be dropped (Early Packet Discard, EPD)

Quality of Service (QoS)

- Application requirements:
  - Realtime applications
  - Non-realtime application (elastic-application – since they can be stretched gracefully in the face of increasing delay)
- Voice applications:
  - At a rate of one per 125 μ s
  - Queue-length in switches an routers vary with time
    - If packets are arriving in time, the are stored in a *playback buffer*
    - If packets are late, they do not have to be stored very long
    - If packets are to late (arriving after their *playback time*) $\rightarrow$ draining of input buffer

For audio applications: maximal Mouth-to-Ear delay of 300ms



Quality of Service (QoS)

- Approaches to QoS support
  - Fine grained approaches: provides QoS in individual applications or flows:

    … here we find "Integrated Services" (developed in the IETF) and often associated with the Reservation Protocol (RSVP)
  - Coarse grained approach: provides QoS to large classes of data or aggregated traffic

    … here we find "Differentiated Services" (undergoing standardization at the time of writing)

ATM is known to have a rich set of QoS capabilities and is considered in the fine-grained category (since resources are associated with individual VCs).

ATM is often used to interconnect routers – and may choose to send a highly aggregated traffic down a single VC $\rightarrow$ so ATM can be used for coarse grained QoS as well.

Quality of Service (QoS)

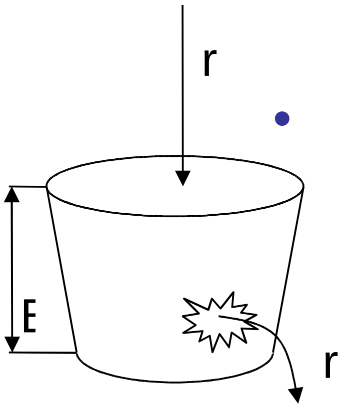
- **Integrated Services** (RSVP)
  - The “Integrated Service Working Group” developed specifications of a number of service classes, designed to meet the needs of some application types (1995-97)
  - It also defines, how RSVP can be used to make reservations, based on these service classes
  - Service classes:
    - Intolerant applications: maximum delay of any packet is guaranteed by the network $\rightarrow$ playback point can be set
    - Controlled load: By using queuing mechanisms, we can isolate the controlled load traffic from the other traffic
    - …
  - Overview of Mechanisms:
    - Best effort service: we can just decide, where to send our packets
    - Provide the network with a set of information $\rightarrow$ flowspec
    - Ask the network to provide particular services $\rightarrow$ admission control

? Quality of Service (QoS)

- User (application) and network have to exchange information to request services, *flowspecs and admission control* → resource reservation (signalling)
- Manage the way how packets are queued and scheduled in the switches and routers → packet scheduling

– Flowspecs

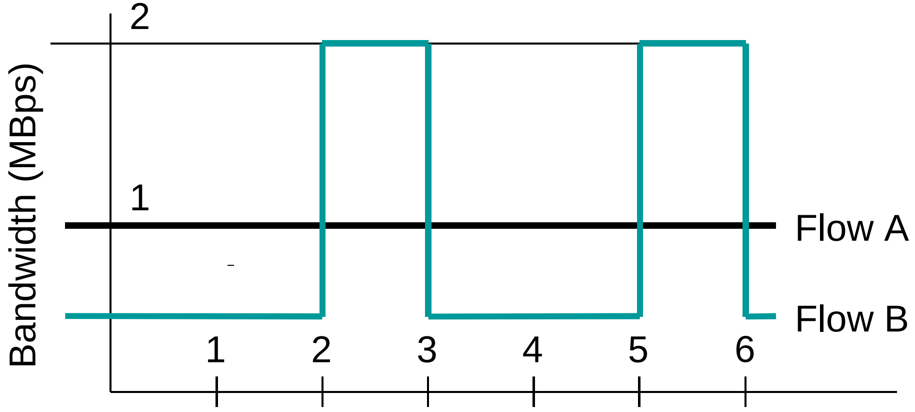
- TSpec: describing the flow's traffic characteristics – give the network enough information about the needed bandwidth → for intelligent admission control decisions.
- RSpec: describing the service request



- Token Bucket filters are used for admission control
  - Token rate r, and token depth B
  - I can send a burst of B bytes, but I cannot send more than r bytes per second for a longer period

Quality of Service (QoS)

- Token Bucket



- Flow A generates a steady rate of 1MBps (r=1MBps, B=1B)
- Flow B sends an average rate of 1MBps (r=1MBps, B=1MB)

A single flow can be described by many different token buckets – but we have to avoid over-allocation of resources in the network.

Quality of Service (QoS)

– Admission Control

- Admission control looks at TSpec and RSpec and decides, if a desired service can be provided to that amount of traffic.
- For a controlled load service, the decision may be based on heuristics *“last time I allowed a flow with this TSpec into this class, but the delay for this class exceeded the acceptable bound – so I had better say no”* or *“my delay are so fair inside the bounds, that I should be able to admit another flow”*
- Admission control := per-flow decision to admit a new flow …
- Policing := per-packet decision – to make sure that all flows are conform to their TSpec’s

? Quality of Service (QoS)

– Reservation Protocol (May 1993)

- RSVP is trying to maintain the robustness of a network, by two highly innovative features:
  - receiver initiation and soft state
  - Hard states – found in connection oriented networks
- Soft states do not need to be explicit deleted when it is no longer needed
- RSVP → Receiver-oriented approach (in contrast to connection oriented networks → resource reservation is done by the sender)
- Simple to increase or decrease the level of resource allocation:
  - Each receiver is periodically sending refresh messages to keep the soft-state in place

Quality of Service (QoS)

– Reservation Protocol

1. The receiver needs to know, what traffic the sender is likely to send → it need to know the sender's TSpec.

- … it needs too know what path the packet will go …
- Establish a resource reservation at each router on the path

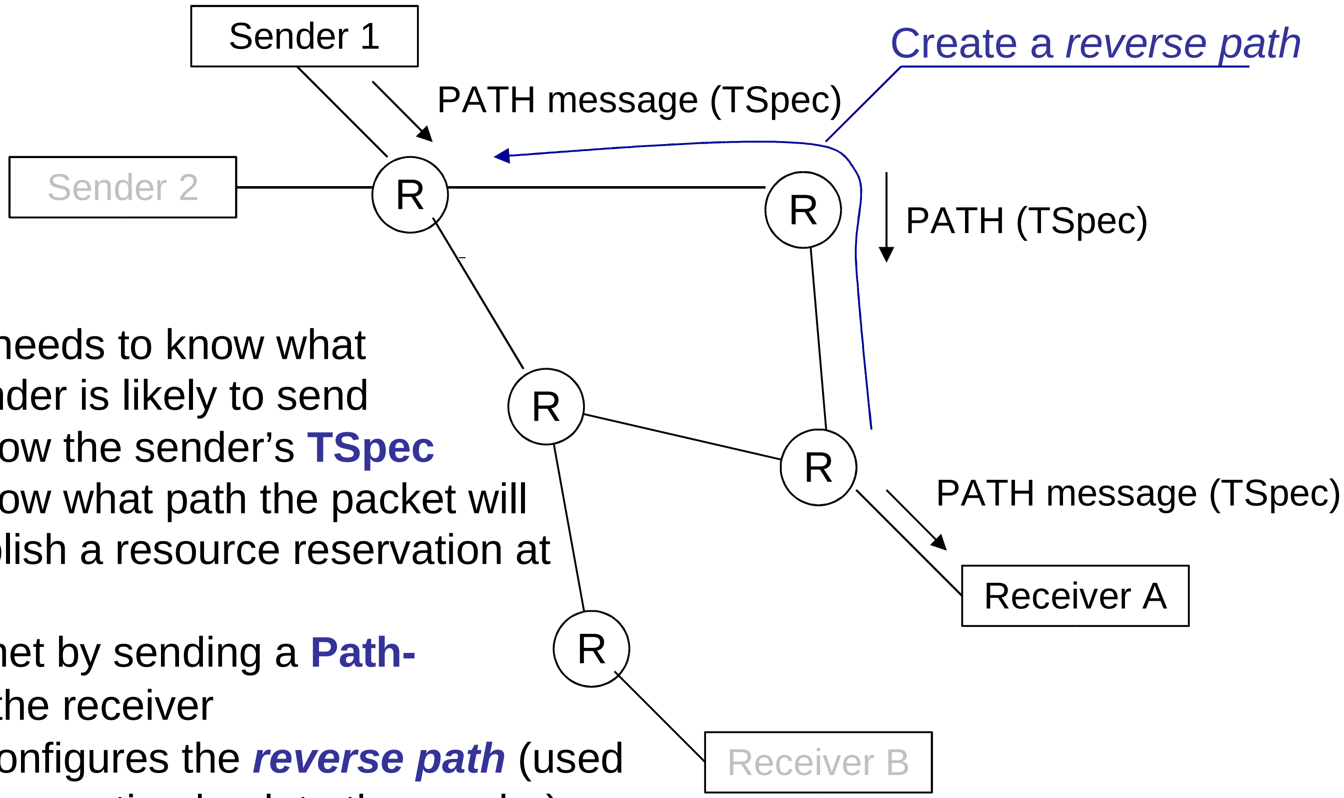
4. Send a message from sender to receiver (containing the *TSpec* → **Path Message**)
   – Each router is looking at this packet

- Reservations are sent back to the sender (**RESV Message** → containing the *RSpec's*)

… if the reservation can be made → the RESV request is passed to the next router

Quality of Service (QoS)

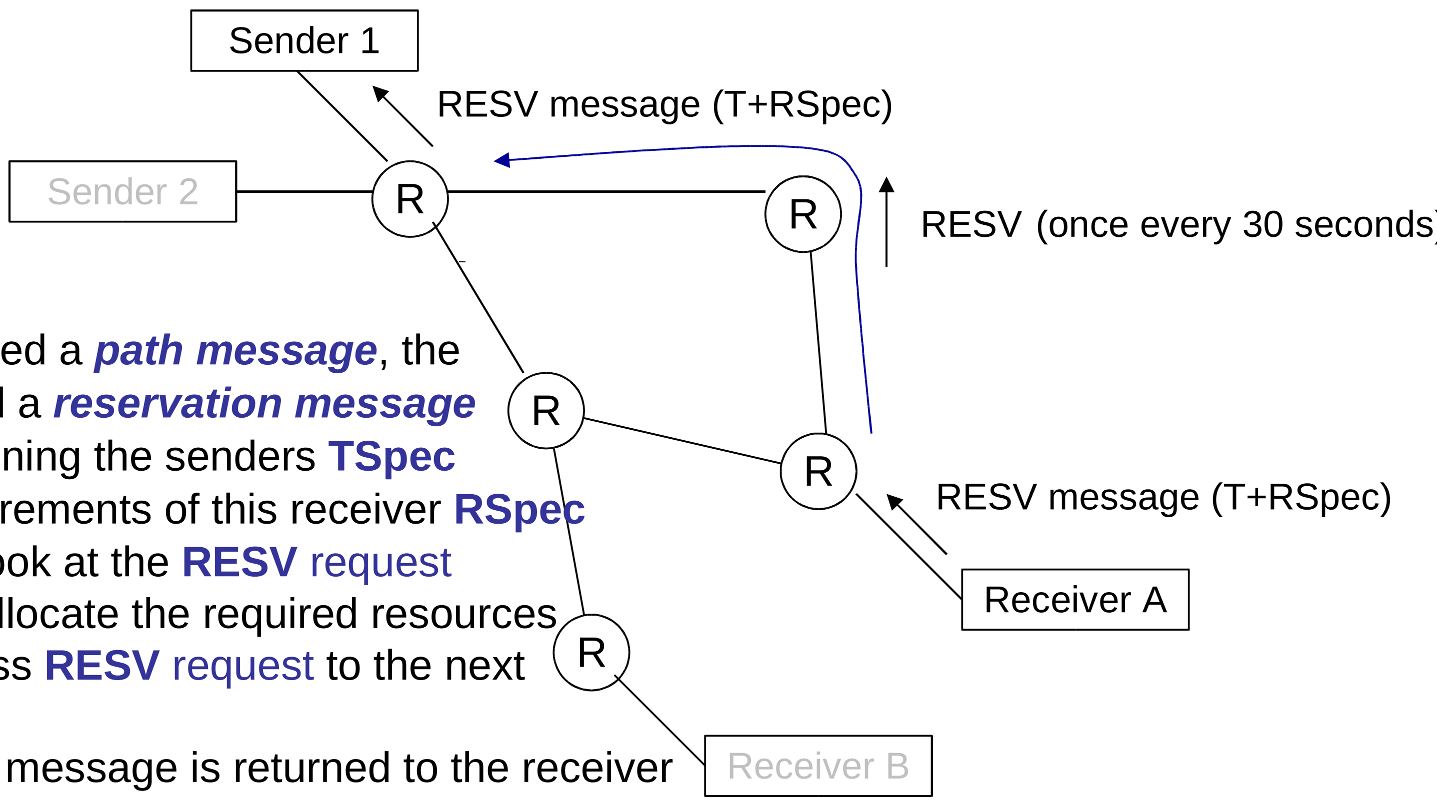
– One sender – one receiver



- The receiver needs to know what traffic the sender is likely to send
- It needs to know the sender's **TSpec**
- It needs to know what path the packet will flow (to establish a resource reservation at each router)
- This can be met by sending a **Path-message** to the receiver
- Each router configures the ***reverse path*** (used to send the reservation back to the sender)

Quality of Service (QoS)

– One sender – one receiver



- Having received a ***path message***, the receiver send a ***reservation message*** back – containing the senders **TSpec** and the requirements of this receiver **RSpec**
- Each router look at the **RESV** request and tries to allocate the required resources

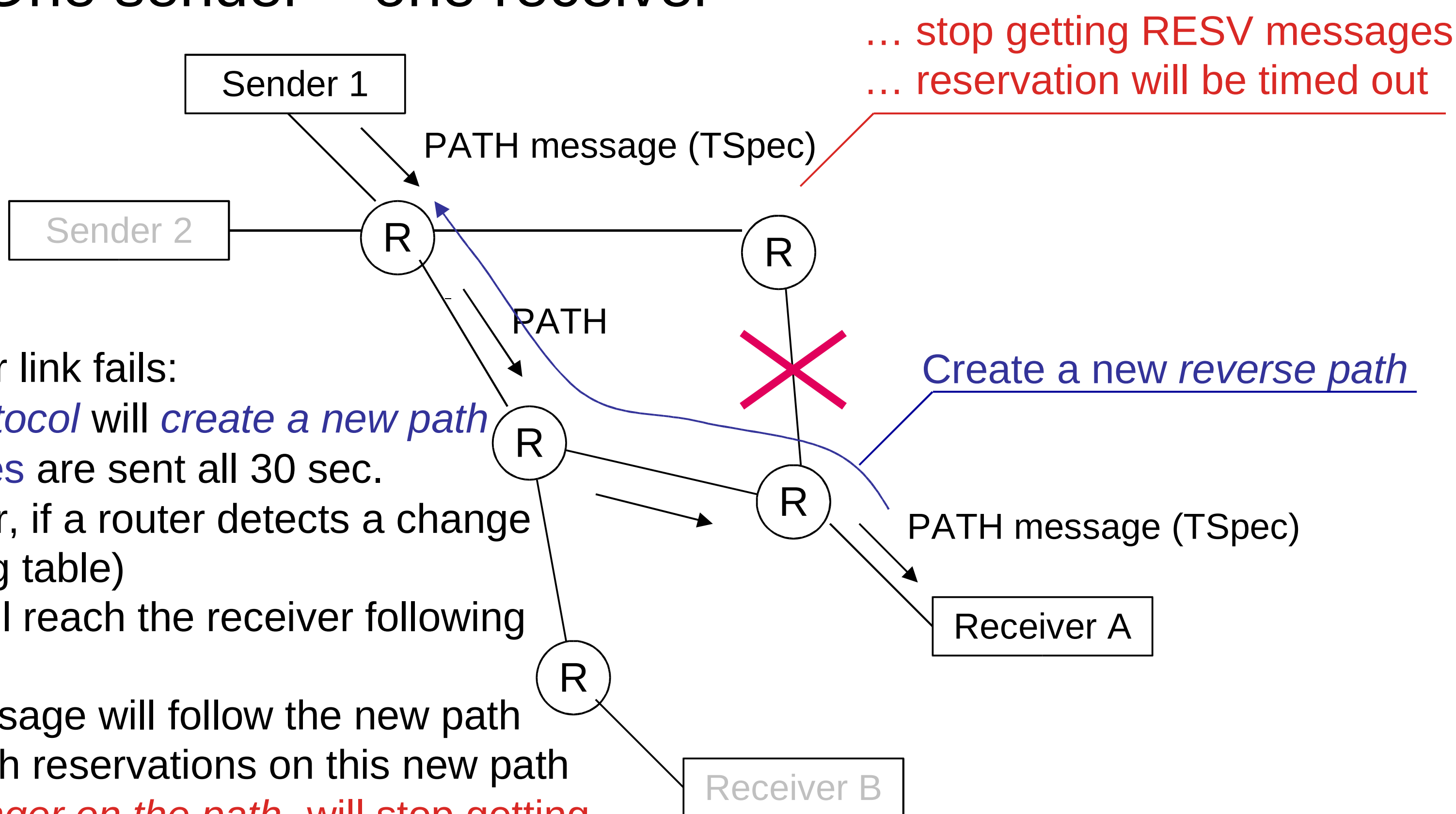
If possible: pass **RESV** request to the next router

If not: an error message is returned to the receiver

- As long as the receiver wants to retain the reservation it sends the same **RESV** message all 30 seconds

Quality of Service (QoS)

– One sender – one receiver

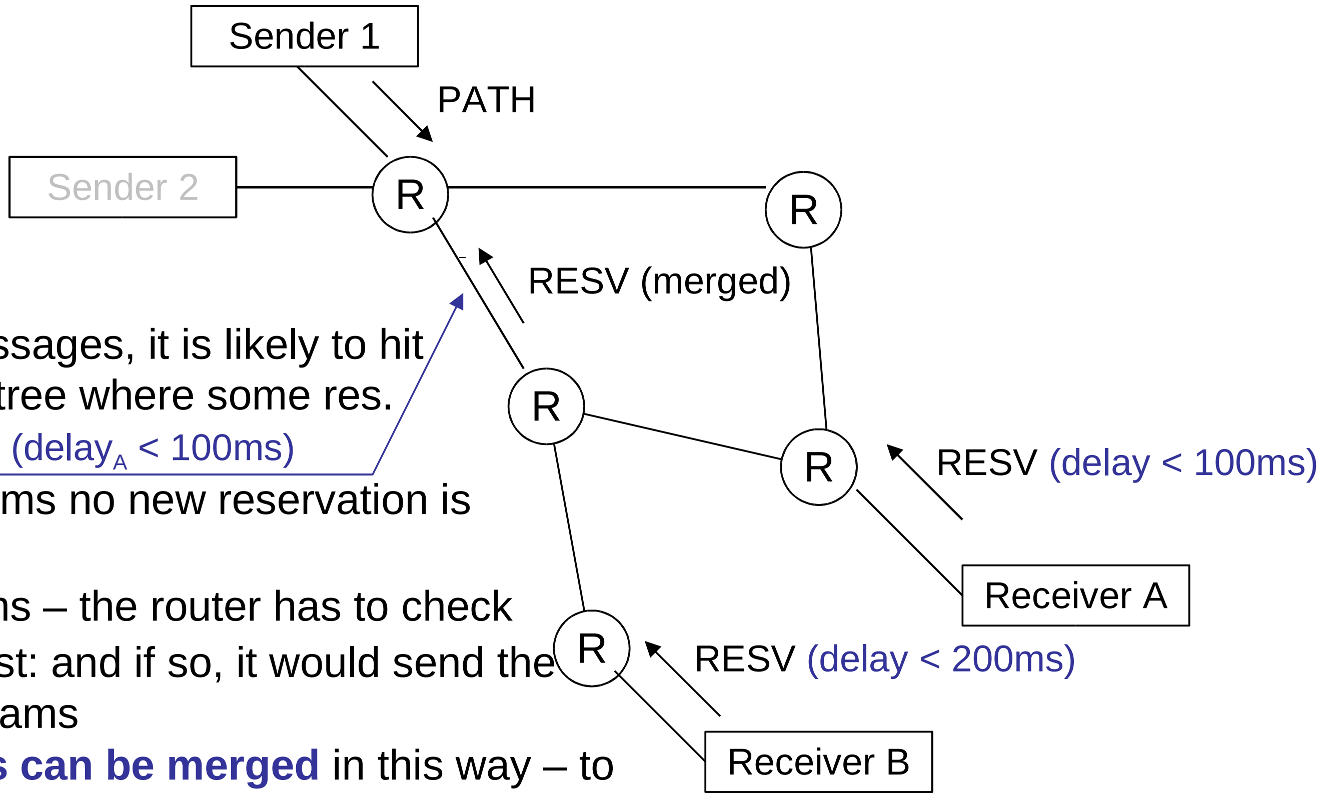


When a router or link fails:

- The *routing protocol* will *create a new path*
- **PATH** messages are sent all 30 sec. (may be sooner, if a router detects a change in its forwarding table)
- The first one will reach the receiver following the new path
- The RESV message will follow the new path
- … and establish reservations on this new path
- Routers *–no longer on the path-* will stop getting RESV messages – reservation will be timed out and released **(soft state)**

Quality of Service (QoS)

– Single sender to multiple receivers



For RESV messages, it is likely to hit a piece of the tree where some res. already exists. (delay$_A$ < 100ms)

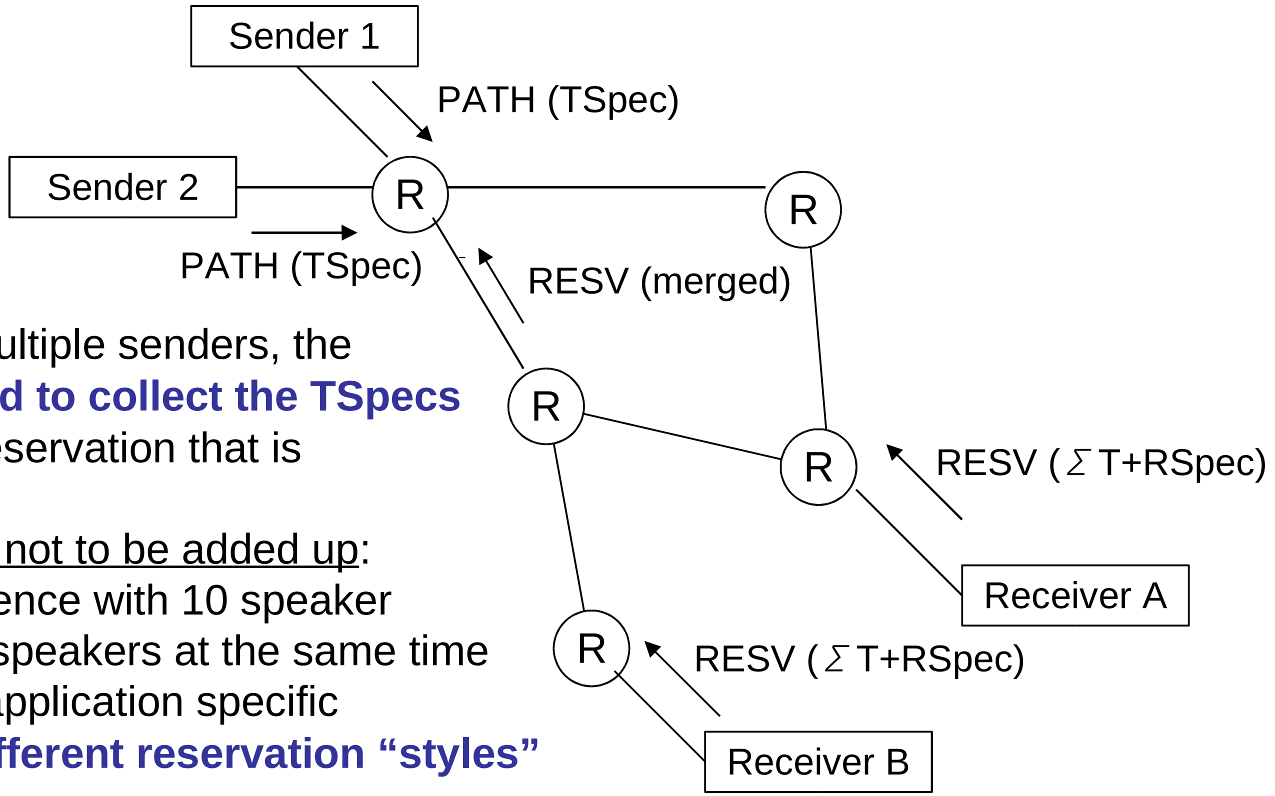
If delay$_B$ = 200ms no new reservation is Required

If delay$_B$ = 50ms – the router has to check this request first: and if so, it would send the request upstreams

- **Reservations can be merged** in this way – to meet the needs of all receivers downstream

Quality of Service (QoS)

– Multiple sender to multiple receivers



- If there are multiple senders, the **receivers need to collect the TSpecs** And make a reservation that is large enough
- TSpecs <u>need not to be added up</u>:
  Audio conference with 10 speaker
  … but just 2 speakers at the same time
- TSpec's are application specific
- **RSVP has different reservation "styles"**

Quality of Service (QoS)

– Packet scheduling

1. Associate each packet with the appropriate reservation (*classifying packets*)
   – Done by examining: source address, destination address, protocol number, source port, destination port.
   – … mapped to a class identifier: determining how the packet is handled
2. Manage the packets in the queues (*packet scheduling*)

- For guaranteed services a <u>weighted fair queue</u> (one queue per flow) will provide a <u>guaranteed end-to-end delay</u>
- For controlled load, simpler schemes may be used

Quality of Service (QoS)

- Scalability Issues
  - Integrated services and RSVP represents a ***<u>significant enhancement of the best effort service model of IP</u>***
  - … but one of the fundamental IP design-goals is not supported: **scalability**
  - In the “best-effort” service model, no flow specific states are stored at the routers
  - Thus, as the Internet grows, the only thing routers have to do is, to keep up with this links speeds.
  - Example: 64kbps audio channels on an OC-48 link (2.5Gbps):

$$2.5x10^9/64x10^3 = 39{,}000$$

  - Each of this reservations needs some memory
  - … and has to be refreshed periodically

Quality of Service (QoS)

- Scalability Issues
  - Routers have to classify, police, and queue <u>each flow</u>
  - Admission control decisions have to be made
  - "Push-back" mechanisms are required – *to prevent long term reservations*
  - The scalability problem has prevented the widespread of "Integrated Services"
  - Other approaches do not require a per-flow state
  - The next section discusses such approaches:
    - Differentiated Services
    - ATM
    - …

? Quality of Service (QoS)

- **Differentiated Services**
  - The DiffServ model allocates resources to a small number of classes of traffic.
  - Usually just two classes: Premium, Regular
  - We are using RCVP to tell the routers that some flow is sending "premium" packets.
  - *It would be much easier, if the packet identifies themselves at the routers*
    - Can be done, using a single bit in the packet header
  - Two questions:
    - Who sets the "premium" bit, and under what circumstances?
      - » Set the bit as an administrative boundary
    - What does a router differently when it sees a packet with a bit set?
      - » Different router behaviors are specified (by the DiffServe IETF WG)

Quality of Service (QoS)

- Differentiated Services
  1. Router behaviors are called: “per-hop behaviors” (PHB’s)
     - Indicates the behavior of individual routers rather than end-to-end services
     - … more than 1 bit is required
     - The old TOS byte of the IP header was taken:
       - » Six bits of this byte have been allocated for the “DiffServ Code Points” (DSCP) identifying a particular PHB (applied to a packet).
       - » The simplest PHB: “*expedited forwarding*” (EF) $\rightarrow$ should be forwarded with *minimal delay and loss*
       - » EF-arrival rate is limited by the link speed of the router
       - » EF can be implemented by:

         a.) strict priority over all other packets

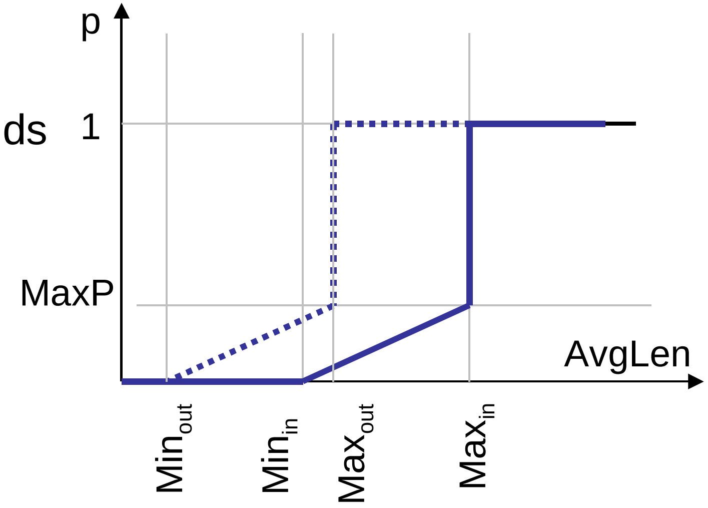
         b.) weighted fair queuing $\rightarrow$ *can prevent routing packets from being locked out*

Quality of Service (QoS)

– Differentiated Services

1. Another PHB: "*assured forwarding*" (AF) $\rightarrow$ has its roots in "RED (*random early detection*) with in and out" (RIO) or "weighted RED"

"out" has lower tresholds



- Two classes of traffic $\rightarrow$ two drop probability curves (in, out)
- In- or out- of the profile (guaranteed by the edge router – service provider)

Quality of Service (QoS)

– Differentiated Services
  – There must be enough bandwidth so that “in”-packets allone are rarely able to congest the link (RIO starts dropping “in”-packets)
  – RIO does not misorder “in”- and “out”-packets $\rightarrow$ “fast retransmit” can be falsely triggered
  – RIO can be generalized to more than two drop probabilities
    » DSCP field is used to pick one drop probability curve

1. DSCP can also be used to determine a queue to put a packet into a “*weighted fair queuing”* scheduler (WFQ).

Example:
  – One code point to indicate “best-effort” services (queues)
  – A second to select the “*premium queue”* (higher weighted than best-effort)

$B_{premium} = W_{premium} / (W_{premium} + W_{best\text{-}effort})$ … *premium bandwidth*

$= 1/(1 + 4) = 0.2$ … 20% *reserved for premium traffic*

Quality of Service (QoS)

- Differentiated Services
  - The premium class can be kept low, since WFQ will try to transmit premium packets as soon as possible.
  - If premium load ~10% $\rightarrow$ behaves as if premium traffic is running on an underloaded network
  - If premium load ~30% $\rightarrow$ behaves like a highly loaded network
  - Just as in WRED, we can generalize this WFQ-based approach to allow more than two classes represented by different code points
  - We can also combine the idea of a queue selector with a drop preference

Quality of Service (QoS)

- ATM QoS
  - ATM standardization body came up with five service classes:
    - Constant bitrate (CBR)
    - Variable bitrate – real-time (VBR-rt)
    - Variable bitrate – non-real-time (VBR-nrt)
    - Available bitrate (ABR)
    - Unspecified bitrate (UBR)
  - ATM-QoS is based on VC's – QoS reservation is included in the signaling message
  - ATM and IP service classes are quite similar
    - … just ABR has no IP-counterpart

Quality of Service (QoS)

- ATM QoS
  - VBR-rt $\rightarrow$ similar to “*guaranteed service class*” in IP Integrated Services (the basic idea is the same)
    - Source-traffic is characterized by a token bucket
    - The maximal total delay is specified
  - CBR $\rightarrow$ a special case of VBR (peak rate == average rate)
    - Important for telephone companies
    - Easy to implement
  - VBR-nrt $\rightarrow$ similar to IP’s controlled load service
    - Source is specified by a token bucket
    - … but not the same hard delay guarantee of VBR-rt or IP’s guaranteed service
  - UBR $\rightarrow$ ATM’s best effort service

Quality of Service (QoS)

– ATM QoS
  - ABR $\rightarrow$ more than a service class
    - Defines a set of congestion control-mechanisms
    - Special "*resource management*" cells (RM) $\rightarrow$ explicit "*congestion feedback*" mechanism (similar to DECbit)

3. The source sends a cell to the destination, including the required bitrate (**on VC base**)
4. Switches along the path are looking at this packet, and decide if sufficient resources are available
5. If available $\rightarrow$ the RM cell is passed unmodified
6. … otherwise the request rate is decreased
7. At the destination the RM cell is turned around and sent back to the source – learning the rate it can send.
8. RM cells are sent periodically (with higher or lower request rates)

Quality of Service (QoS)

- ATM QoS
  - Can be expanded to “virtual sources” (VS)