Real-time Ethernet

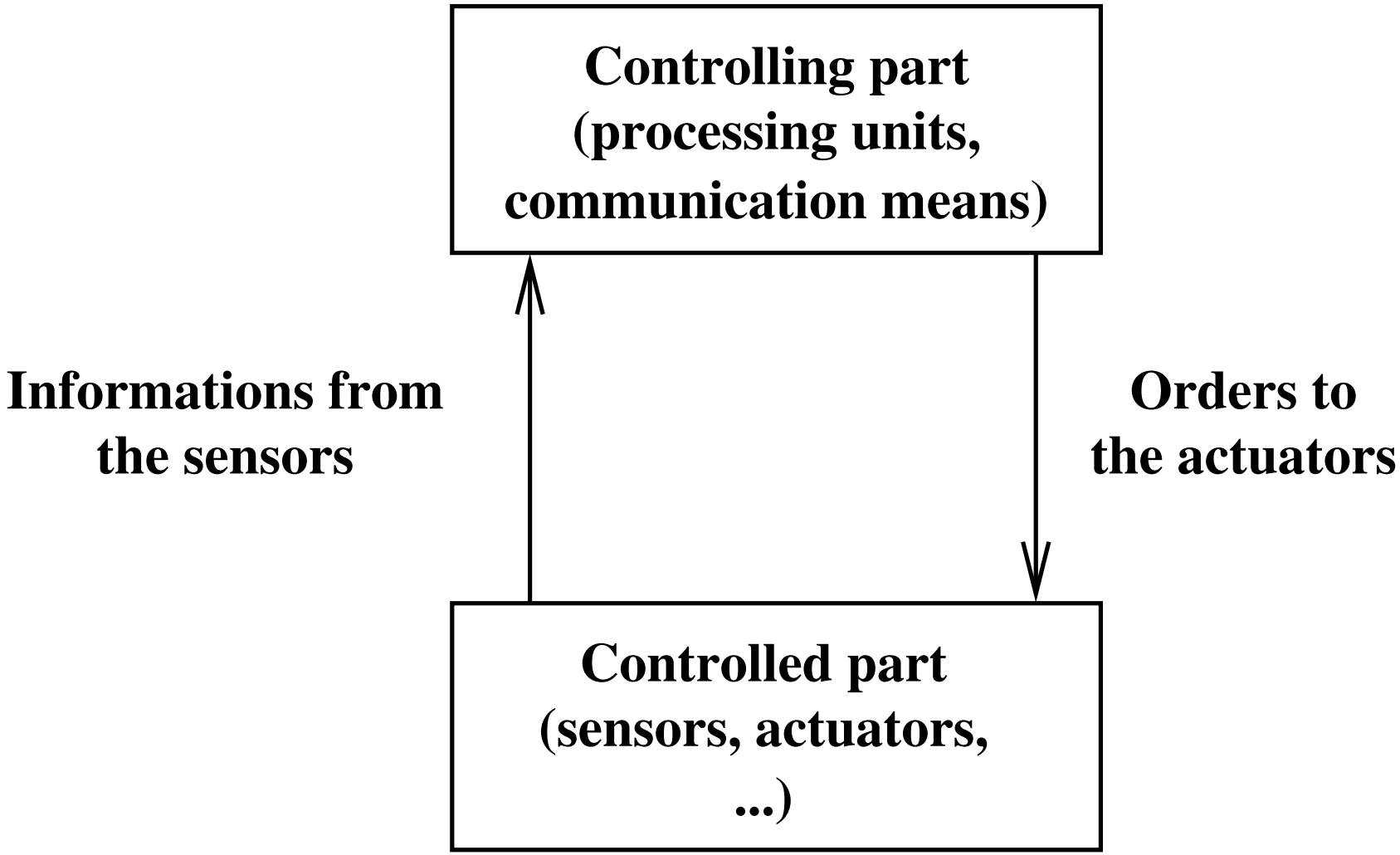
Jean-Luc Scharbarg - ENSEEIHT - Dpt. SN

October 2021

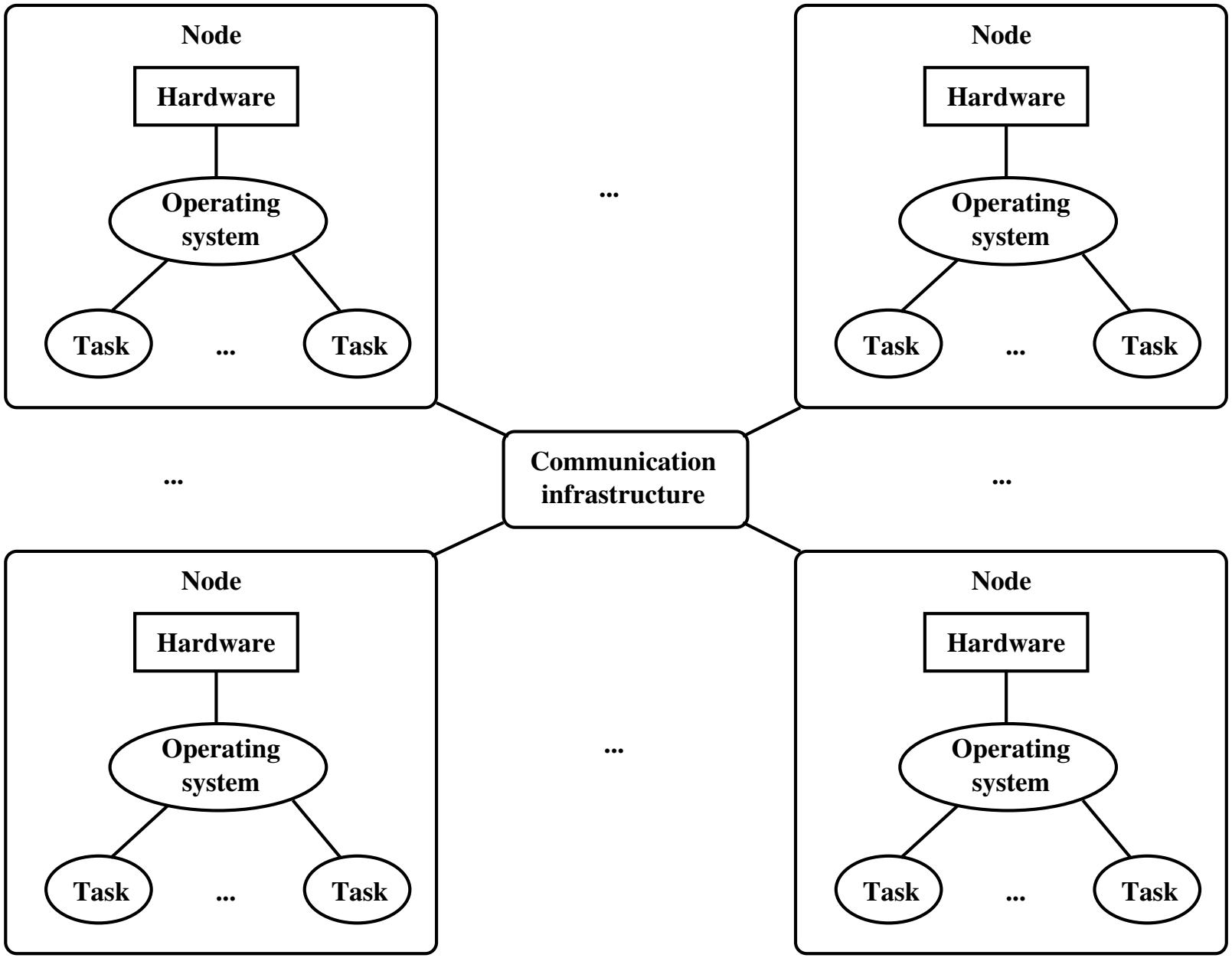
Some characteristics of an embedded system

- Embedded $\Rightarrow$ invisible to the end-user
- The concerned application: a set of domain specific functions
- Implementation of each function by hardware, software or both
  - Software is more flexible
  - Hardware gives more performance
  - Trade-off between both of them
- Increasing hardware and software complexity
- Increasing need for performance
- Application domains
  - Automotive, avionics, space
  - Industrial control
  - Consumer electronics (mobile phone, PDA, video game console, ...)
  - Household appliances (washing machine, coffee machine, ...)
  - ...
- Large variety of needs (performance, reliability, criticallity)

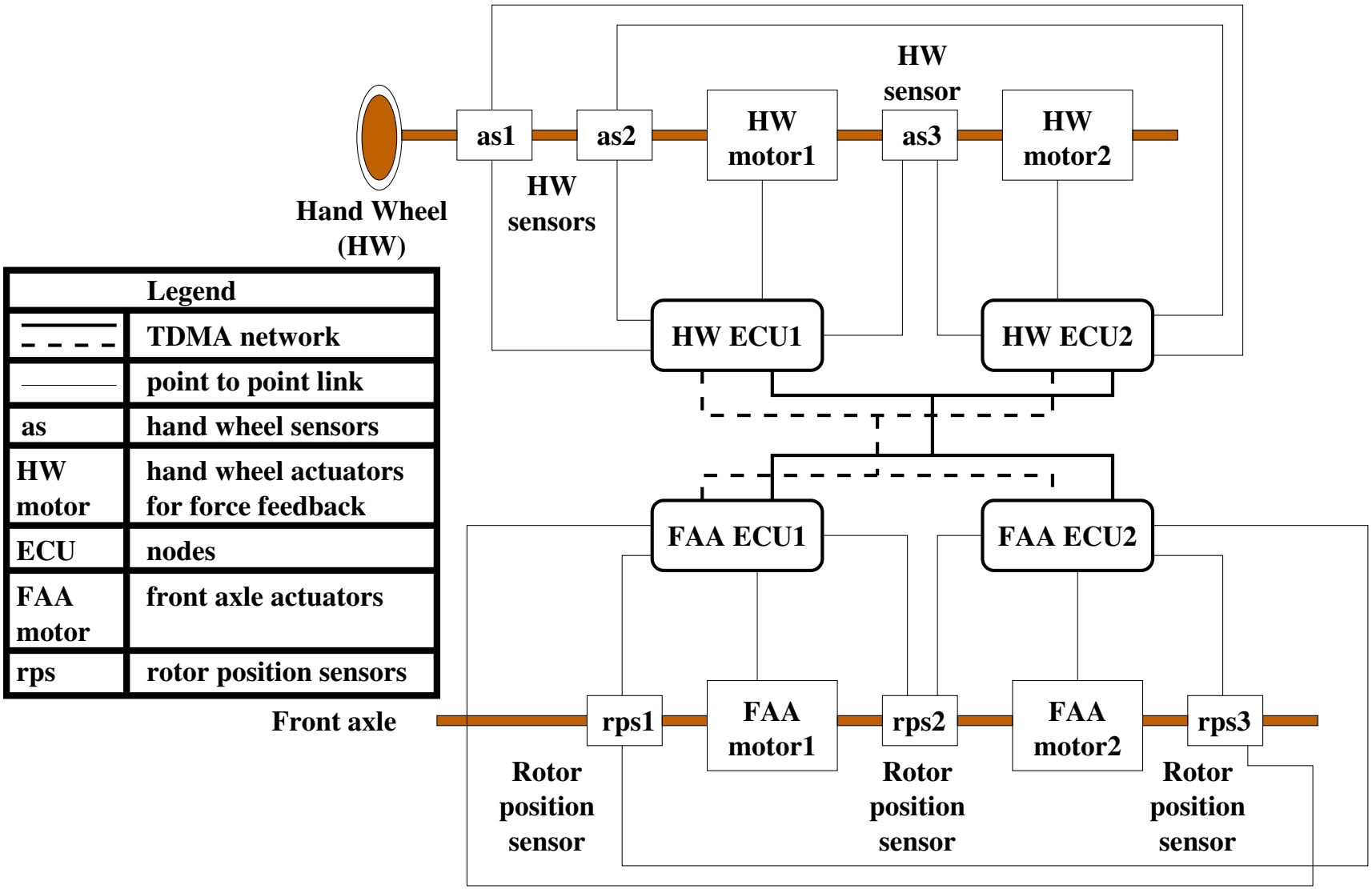
General architecture of an embedded system



Architecture of the controlling part



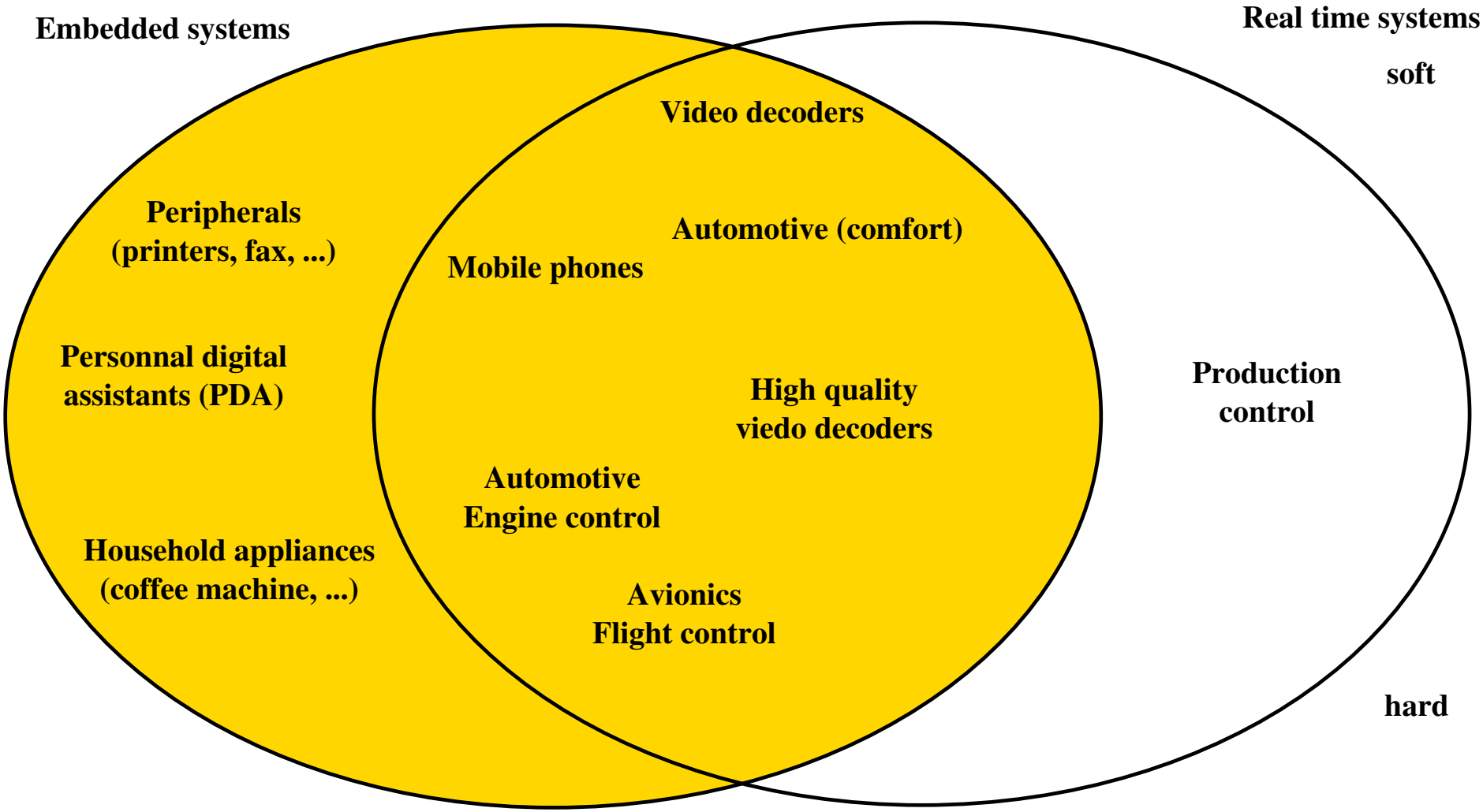
An example: A steer-by-wire operational architecture



Design constraints

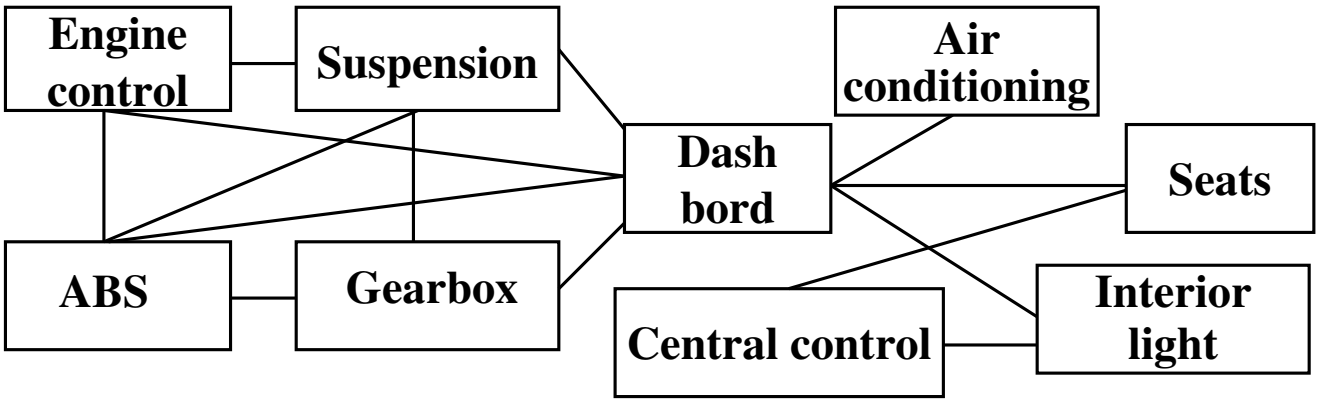
- Real time
  - Real time (physical) is a correctness criterion
  - Need to guarantee timing constraints
- Safety
  - Definition: capability to be confident in the result of a service
  - Obstacles: failures, resulting from faults
  - Means: fault avoidance, fault tolerance
  - Large spectrum of methods depending on the type of faults and the application criticallity (hardware, software)
  - Safety critical systems: certification
- Cost
  - Design cost (delay): time to market
  - Purchase cost: surface of silicium (limited memory, limited energy)
  - Simpler processors (cost and predictability)
  - Ergonomy: weight constraints
- Trade-offs have to be found
  - Energy vs performance
  - Multi-criteria exploration for hardware/software selection

Real time systems vs embedded systems



A bit of history

- Early days: 1 function = 1 execution unit
- Exchanges of data between functions via dedicated links ⇒ deterministic delay

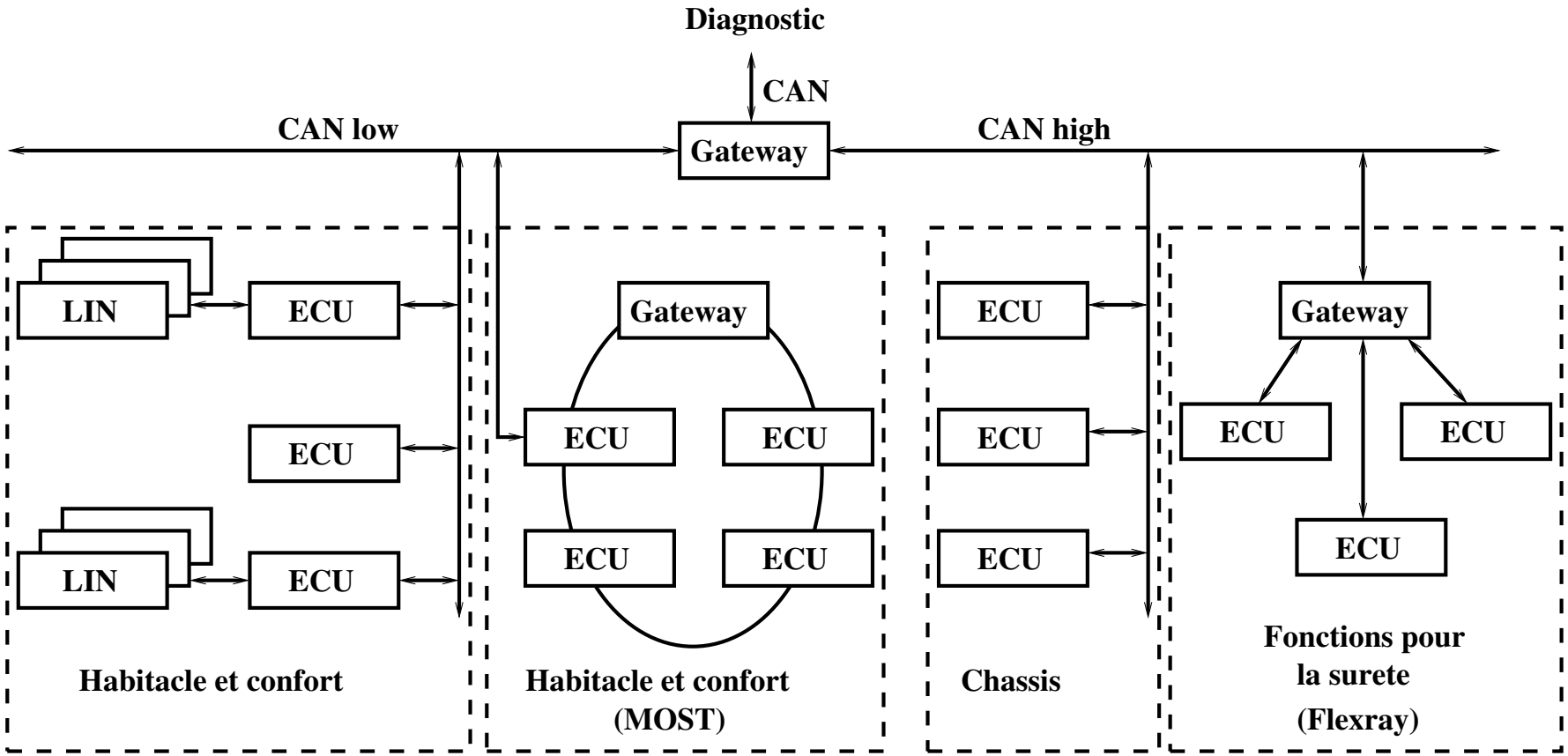


- Huge increase of data exchange leads to unacceptable number of communication links ⇒ share communication means
- Also share execution units

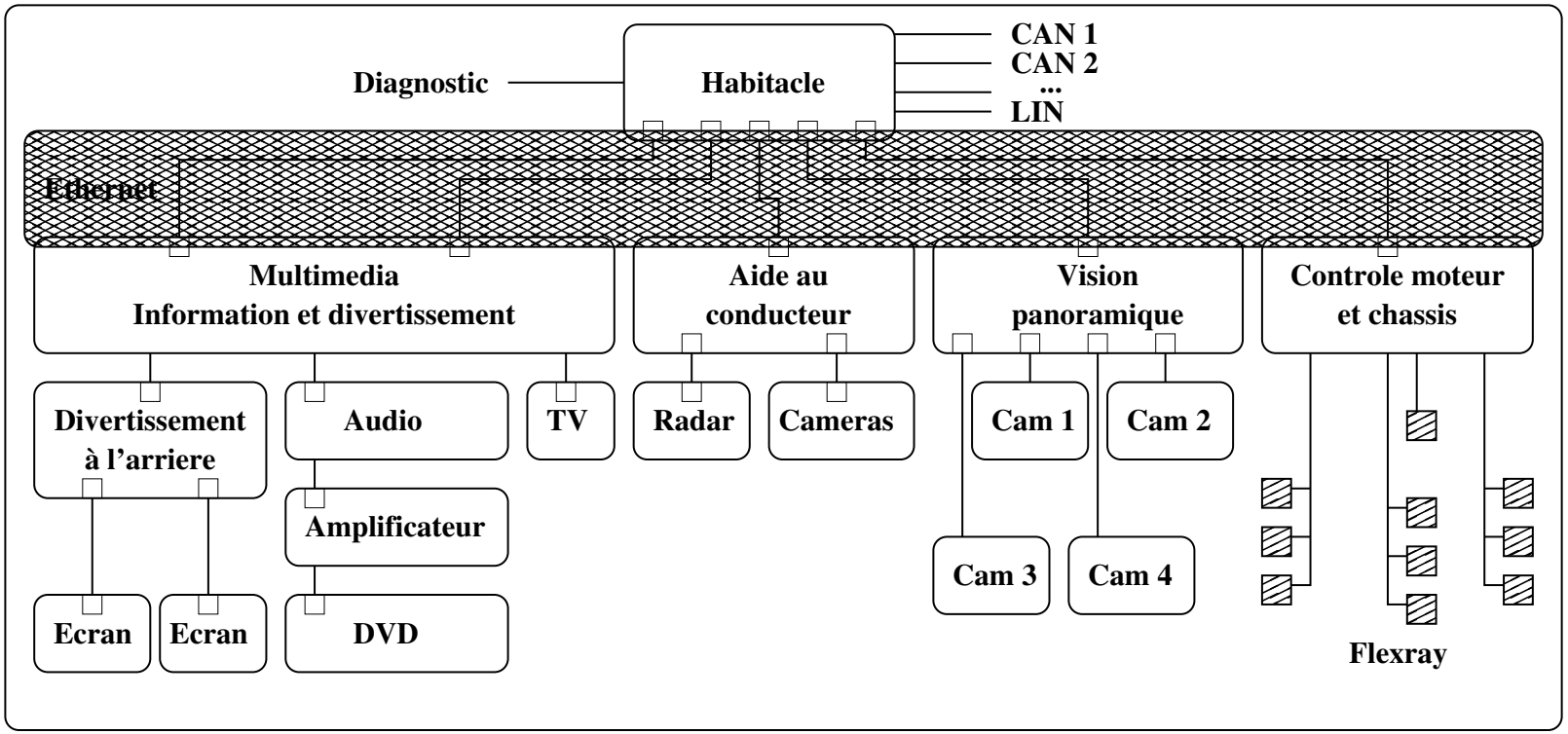
A bit of history

- Replacement of dedicated links by a shared communication medium
- Many different technologies have been proposed
- A very difficult standardization process
- The compromise: a set of different standards in IEC 61158 (Profibus, Worldfip, Foundation Fieldbus, P-Net, Interbus, Profinet, ...)
- Different kinds of communication technologies used
  - Pure fieldbuses: CAN, Profibus, Worldfip, Foundation Fielbus H1, Modbus, P-Net, Interbus, ...
  - Industrial Ethernet solutions: Modbus/TCP, Ethernet/IP, Powerlink, Profinet, EtherCAT, Foundation Fieldbus HSE, ...
  - Emerging wireless solutions: based on wireless standards, such as IEEE 802.11 (WiFi), IEEE 802.15.1 (Bluetooth), IEEE 802.15.4 (Zigbee)
- The choice of a technology depends on many factors
  - The application context
  - The types of flows
  - Historical, economical, political factors
  - ...

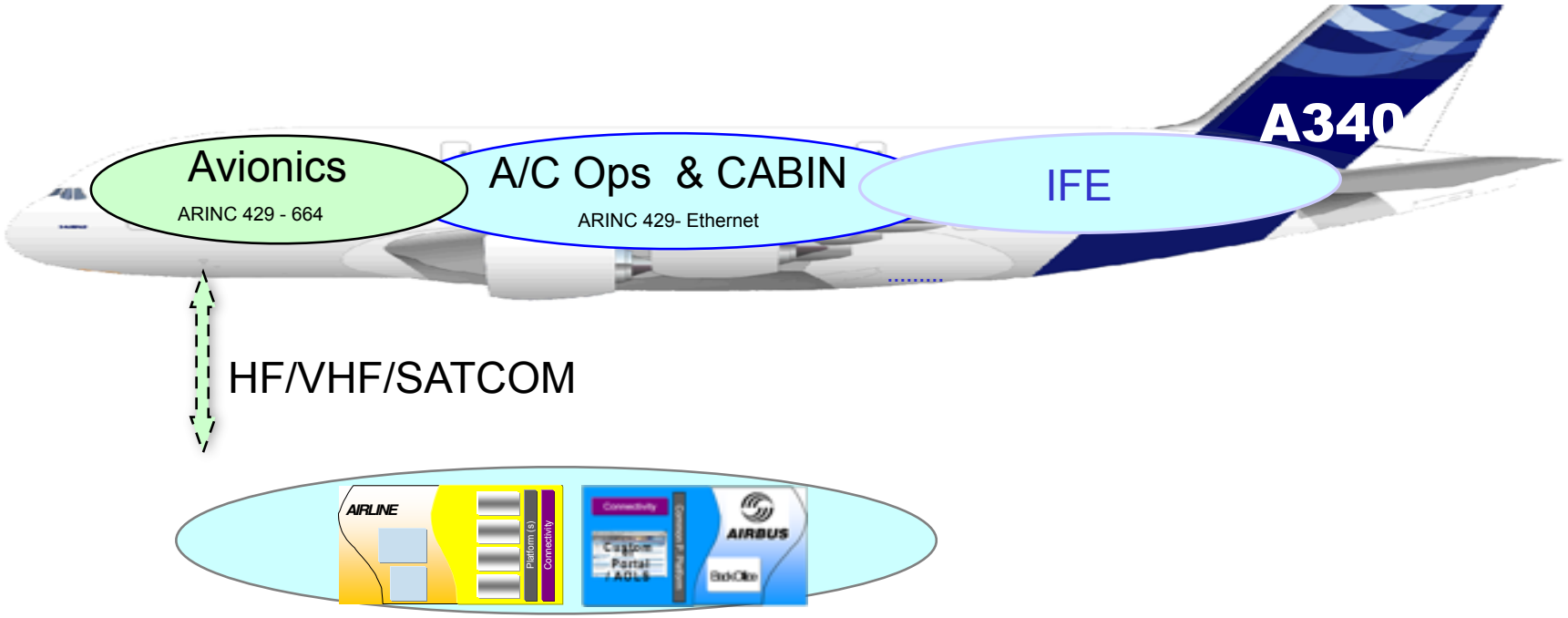
Automotive network architecture in 2006



Vision for future in-car networks



Civilian aircraft communications



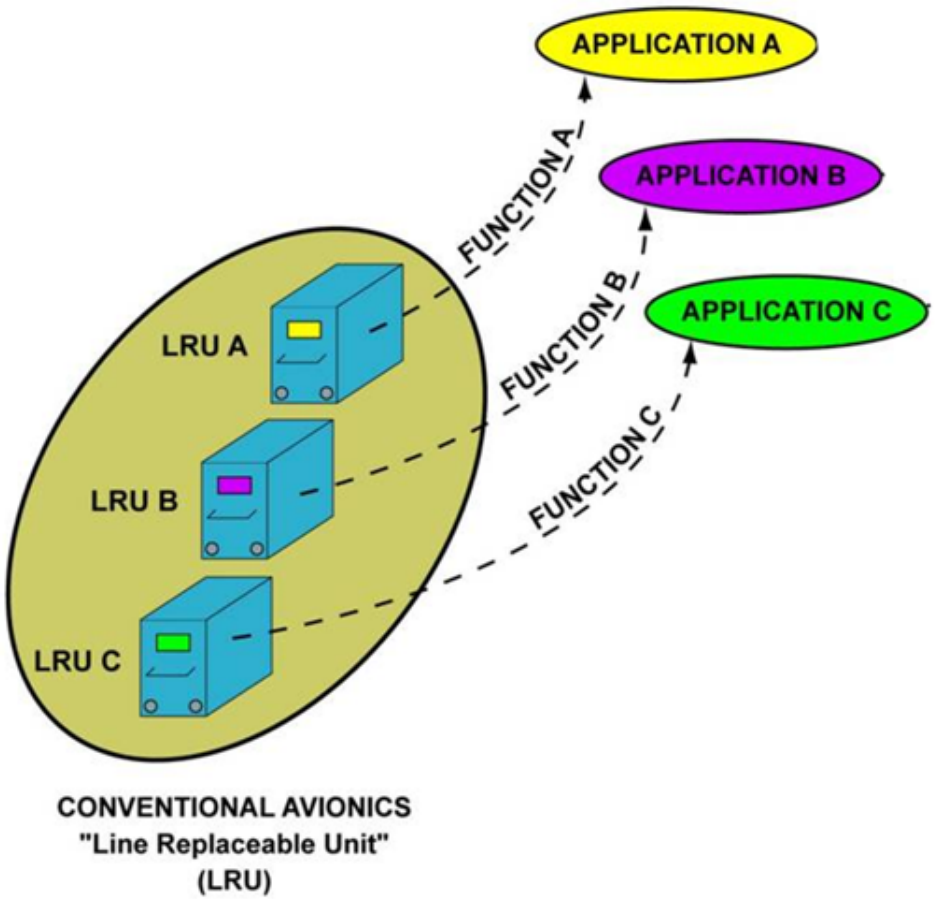
Ground Information System (Airlines, Airbus,…)

Civilian aicraft communications

- The Air/ground communications
  - They provide several means for the communication between aircraft and ground stations for navigation and control: HF, VHF, SATCOM, ...
  - ATC: Air Traffic Control, AOC: Airline Operational Communications, ...
- The Airline passenger communications
  - Part of In-Flight Entertainment (IFE), it includes communications (telephony, e- mail, ...), information (news, weather, web content ...), and interactive services (video games, shopping/e-commerce, surfing the web)
  - Based on Commercial off-the-shelf (COTS) technologies
- The Cabin Communications
  - Based on Ethernet technology open to operational functions (maintenance, ...)
- The Avionics Communications
  - They represent all the information exchanged between the avionics equipment and systems used to the control of the aircraft
  - Based on aeronautical technologies: ARINC 429, AFDX, ...

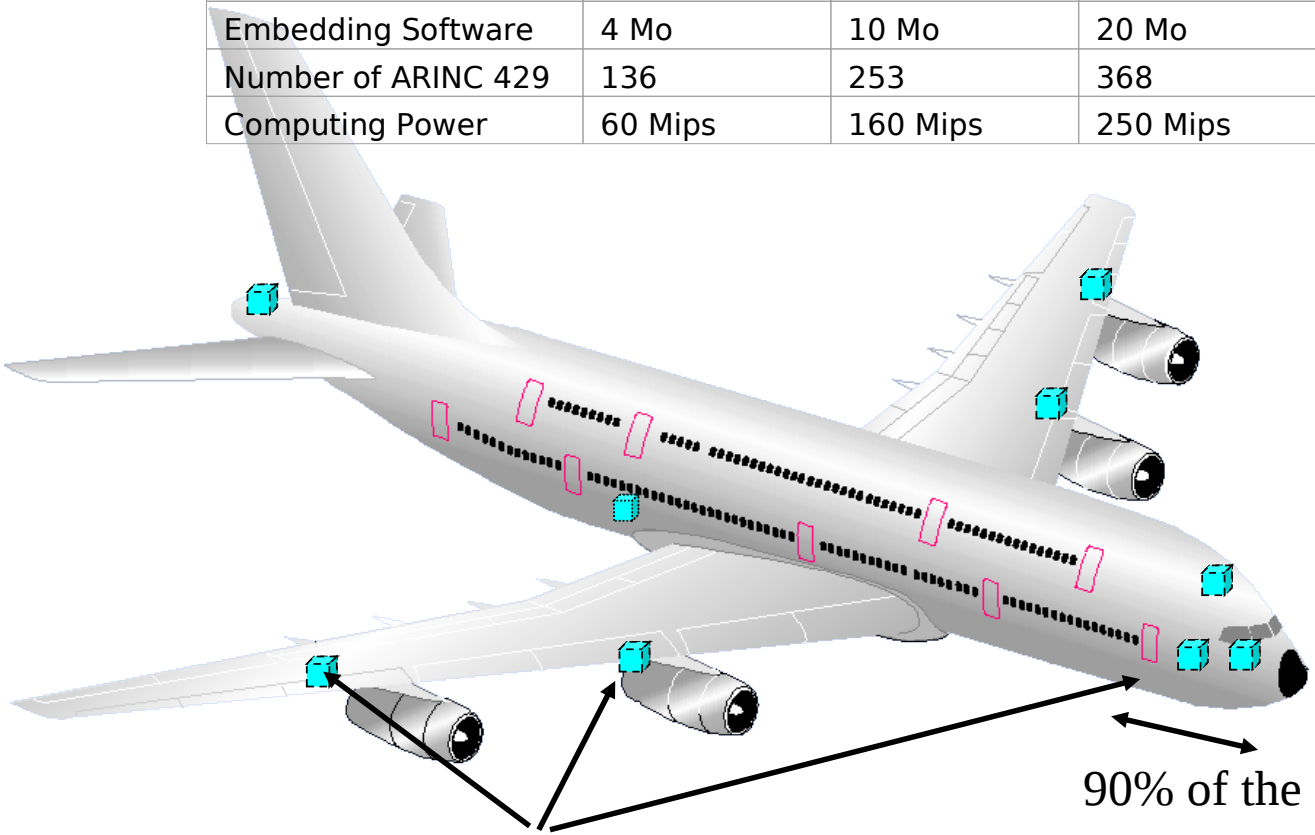
Classical avionics architecture (up to A340)

- Each equipment (LRU) dedicated to a system function (braking, flight computers, ...): sensors, actuators, calculators ...
- natural segregation between the equipments

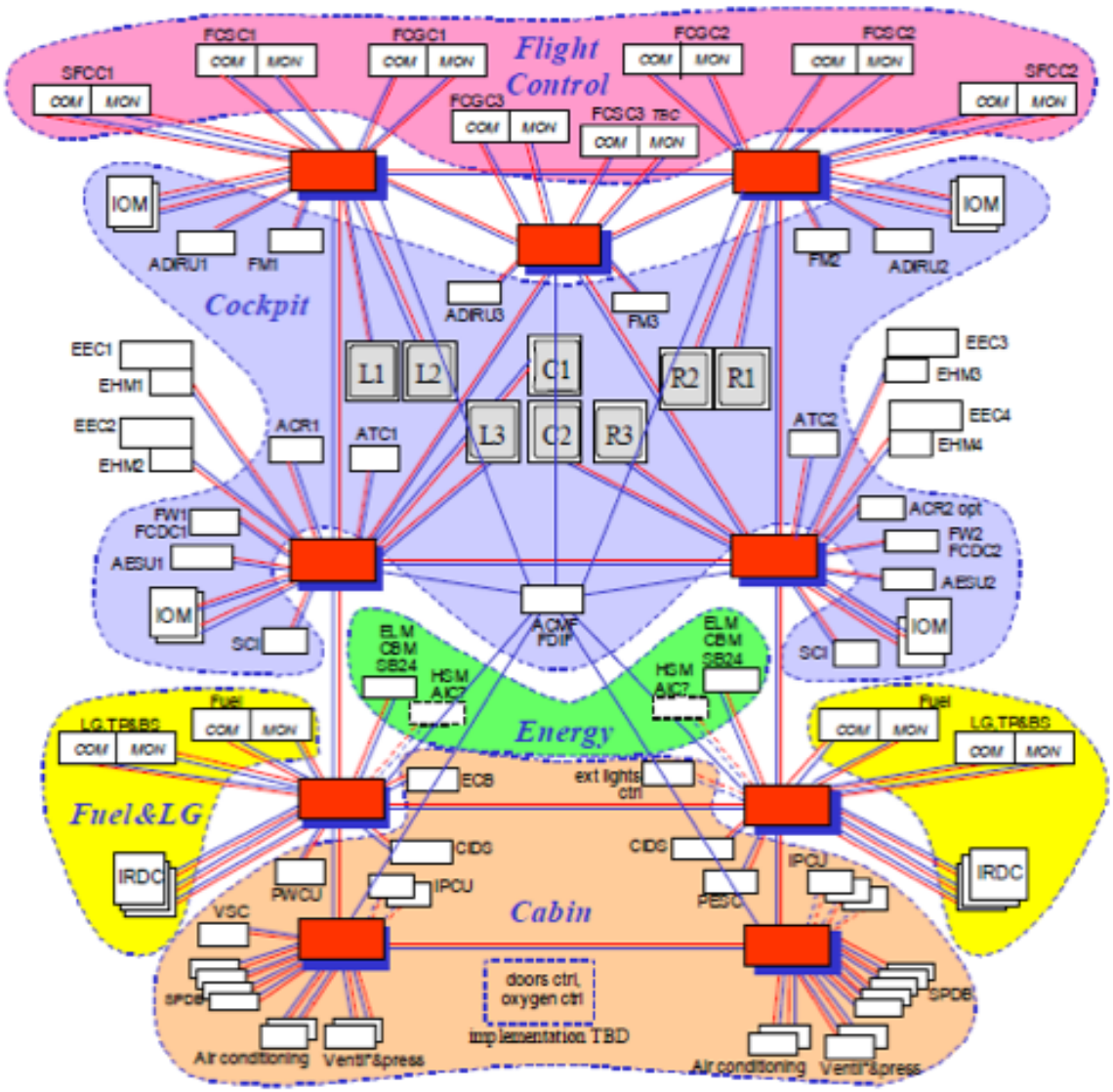


Classic avionics evolution

	A310 (1983)	A320 (1988)	A340 (1993)	A380 (2005)
Avionic Volume	745 litres	760 litres	830 litres	
Number of equipment	77	102	115	147
Embedding Software	4 Mo	10 Mo	20 Mo	100 Mo
Number of ARINC 429	136	253	368	1000 (AFDX)
Computing Power	60 Mips	160 Mips	250 Mips	



An industrial avionics configuration

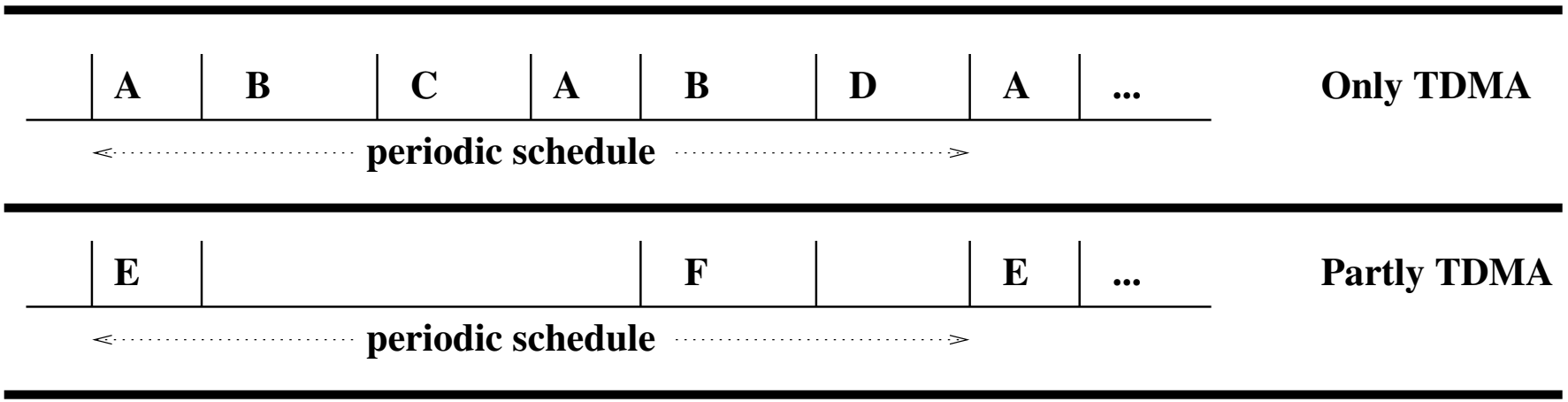


Medium Access Control for real time

- Two main paradigms
  - Time triggered
    - ★ static scheduling
    - ★ Each message is assigned predefined time slots
  - Event triggered
    - ★ Transmissions occur when frames are ready
    - ★ Collisions have to be managed
- No paradigm dominates the other one: depends on flow features
- MAC are based on only one of the two paradigms or both

Dedicated slots per messages

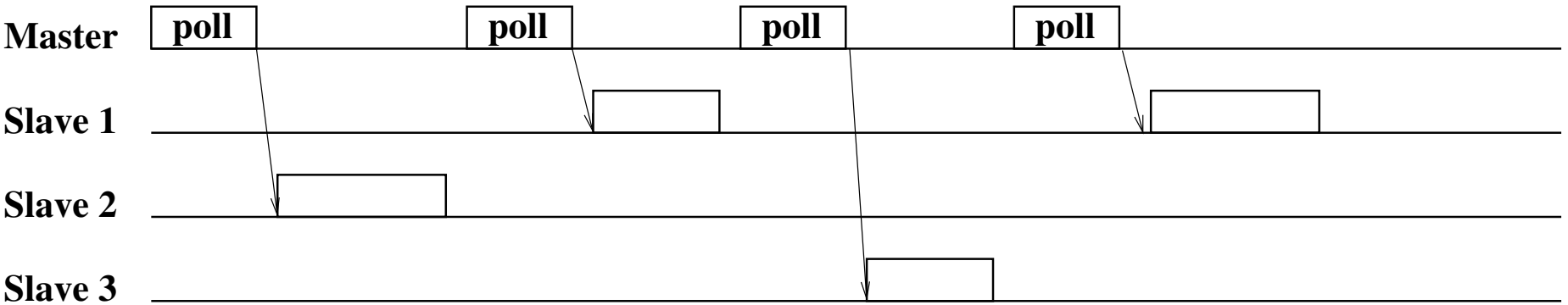
- Time Division Multiple Access (TDMA) $\Rightarrow$ static frame scheduling
- Can allocate the whole resource or only part of it



- Timing behavior fully predictable
- Immediate identification of missing messages
- Very good for periodic messages, poor for the others
- Problematic evolution of the system
- Need for a global synchronization of the system

Polling by a master

- The master node sends a poll frame to a given slave
- The slave answers with the corresponding message



- The master allocates slots to the slaves
- More flexible than previous TDMA solution
- No need for a global synchronization
- Overhead of poll messages
- Problem when master is down

Token passing

- A specific frame (token) is transmitted from stations to stations
- The station which has the token is allowed to transmit
- Mechanisms to provide a fair and periodic access to the token
- Mechanisms to regenerate the token when it is lost.

Virtual token

- Provide a "token-like" behavior without a token frame
- Basic principle
  - A set of n stations with identifiers between 0 and $n-1$
  - A local counter in each station, which takes values between 1 and n
  - A station is allowed to transmit when its local counter equals its identifier
  - Local counters are incremented (modulo n) when there are no transmissions during a specified amount of time
  - Counter incrementations have to be synchronous

Dominance protocol

- Previous solutions: at any time, only one station is allowed to transmit $\Rightarrow$ no collisions
- Dominance protocol: when the medium is idle, any station can start a transmission $\Rightarrow$ collisions might occur
- Non destructive resolution of collisions, based on a priority associated to each frame
- The first field of a frame is its priority (sequence of bits)
- A bit can be dominant (e.g. 0) or recessive (e.g. 1)
- Bit by bit arbitration: two stations transmit at the same time
  - Both transmit dominant or recessive $\Rightarrow$ both continues
  - First one transmits dominant, second one transmits recessive $\Rightarrow$ First one continues, second one stops
- Frame transmissions have to start at the same instant
- Dominance protocol scales better than previous approaches

Summation frame

- Very specific master/slave solution
- Stations are interconnected by a ring
- All the data are exchanged via a single frame.
- It comes to build a shift register on the ring

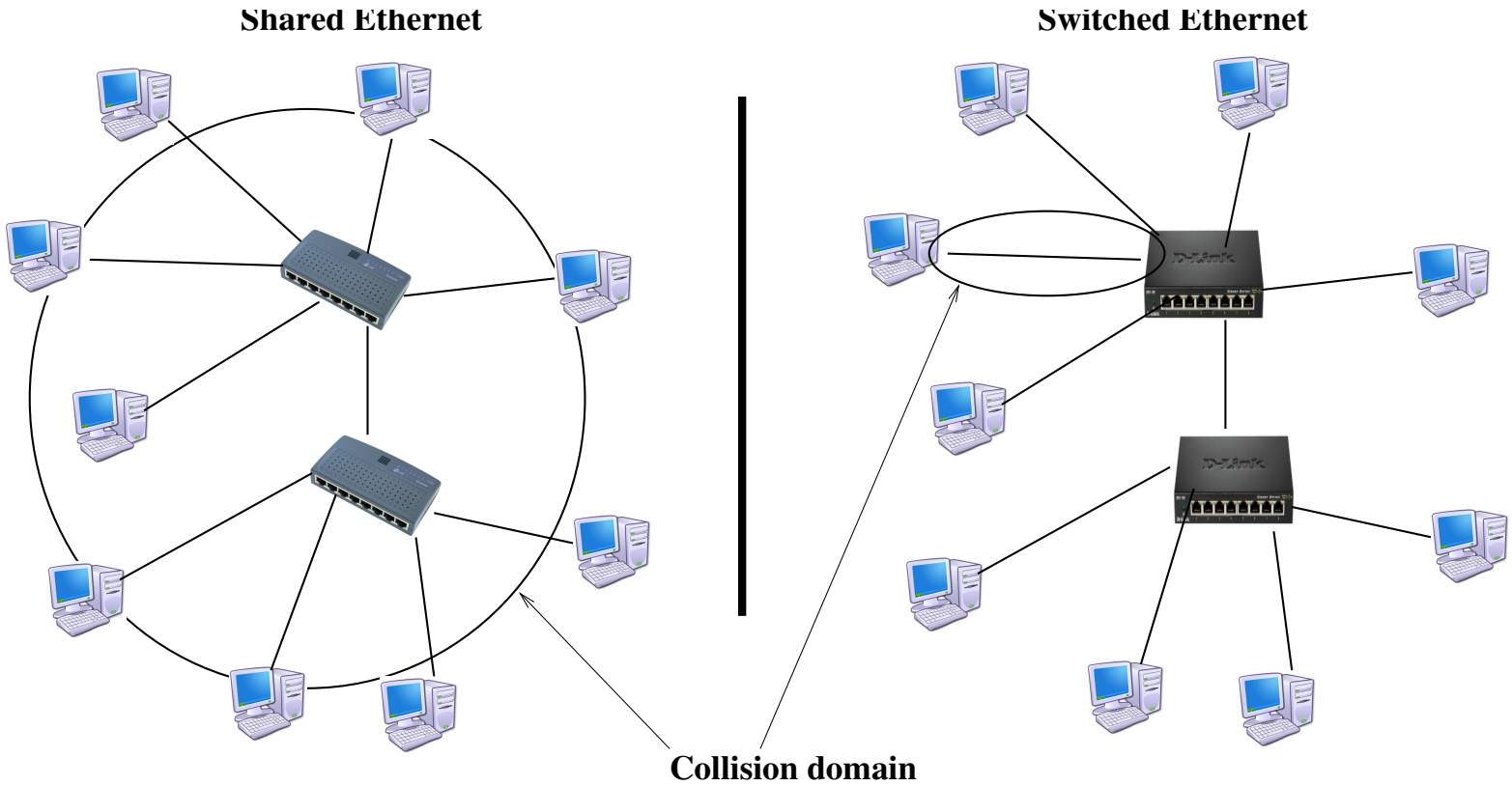
Ethernet and real-time

- Ethernet is appealing for real-time
  - Cheap technology
  - Easy integration with Internet
  - High bandwidth
  - Mature technology with many existing components
- Shared Ethernet is not predictable
  - A set of stations share a single communication channel
  - A frame from a station is received by all the other stations (possibly through hubs)
  - Collisions may occur $\Rightarrow$ CSMA/CD
    - ⋆ CSMA (Carrier Sense Multiple Access): listen to the medium, send only when medium is detected idle
    - ⋆ CD (Collision Detection): if a collision is detected during transmission, transmission is stopped station waits a random delay (exponential backoff) and tries to transmit again
  - At least 64 bytes in order to detect a collision
  - Collision domain: the whole network

Switched Ethernet

- Division of the network with separate collision domains $\Rightarrow$ more than one frame being transmitted at a given time without collision
- A switch:
  - A set of input and output ports with buffers for pending frames
  - A switch fabric which "moves" each frame from its input port to its destination output ports
  - A service discipline for the scheduling of frames in output ports
- Two modes for a switch
  - Store and forward: reception and verification of the whole frame before it is forwarded to its output port(s)
  - Cut through: frame is transfered as soon as destination is decoded
    - ★ Smaller latency
    - ★ Invalid frame might be transferred

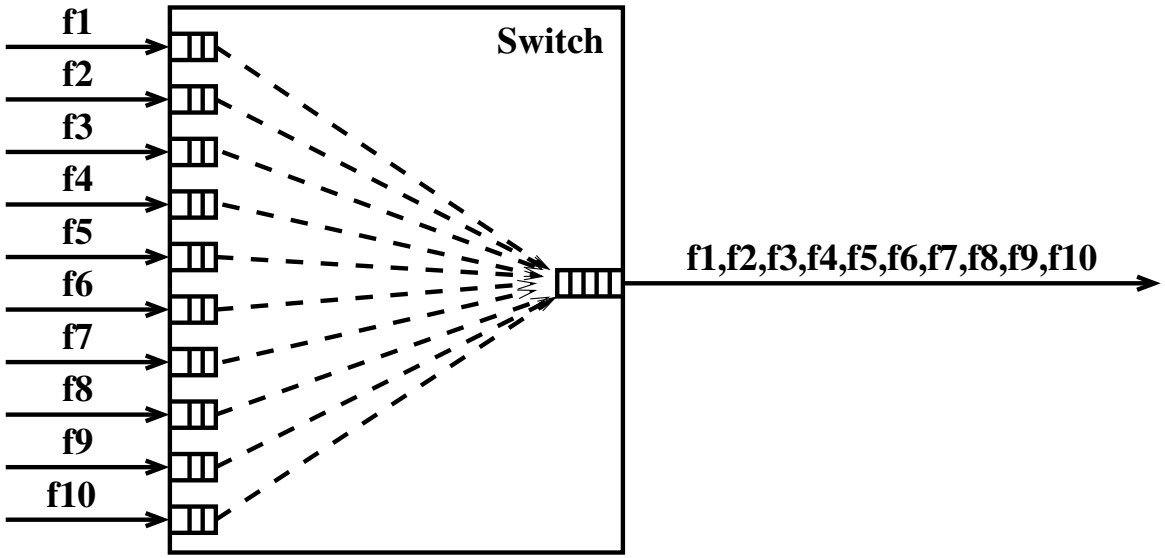
Shared Ethernet Vs switched Ethernet



- Full duplex links $\Rightarrow$ no more collisions

Full duplex switched Ethernet is not deterministic

- Temporary congestion on an output port
  - Increase of the waiting delay of frames in the Tx buffer
  - Frame loss by overflow of the Tx buffer
- An illustrative example



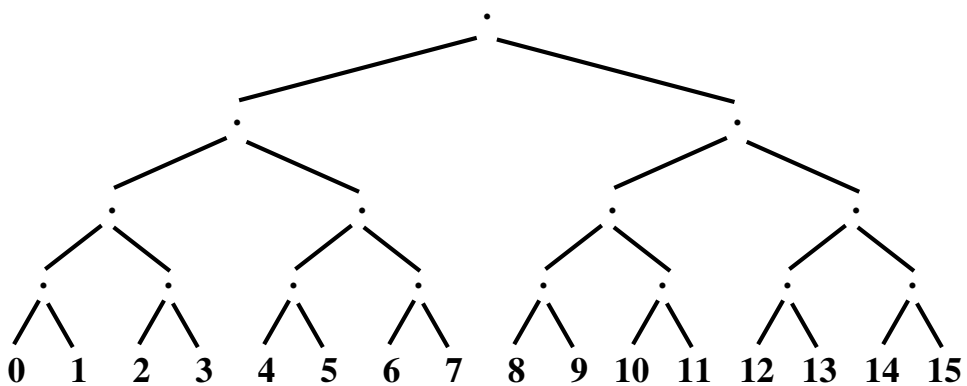
  - Five frames at the same time $\Rightarrow$ one frame waits until the transmission of the four other ones
  - More than five frames at the same time $\Rightarrow$ at least one frame is lost
- Addition of dedicated mechanisms to classical full duplex switched Ethernet in order to guarantee the determinism of transmissions

Real time Ethernet

- Many solutions have been proposed
  - Use Ethernet as it is
  - Use a control mechanism over Ethernet in order to guarantee properties for the real time traffic
  - Modify Ethernet MAC
- In the context of automation and factory automation
  - Many standards such as Profinet, Ethercat, Powerlink, ...
- In the context of avionics
  - Avionics Full DupleX switched Ethernet (ARINC 664) is the de facto standard
- In the context of automotive
  - Ethernet-AVB is the most promising solution
- In the context of space
  - TTEthernet has been used by the NASA
  - Other switched Ethernet solutions are considered
- Short presentation of some real-time switched Ethernet solutions

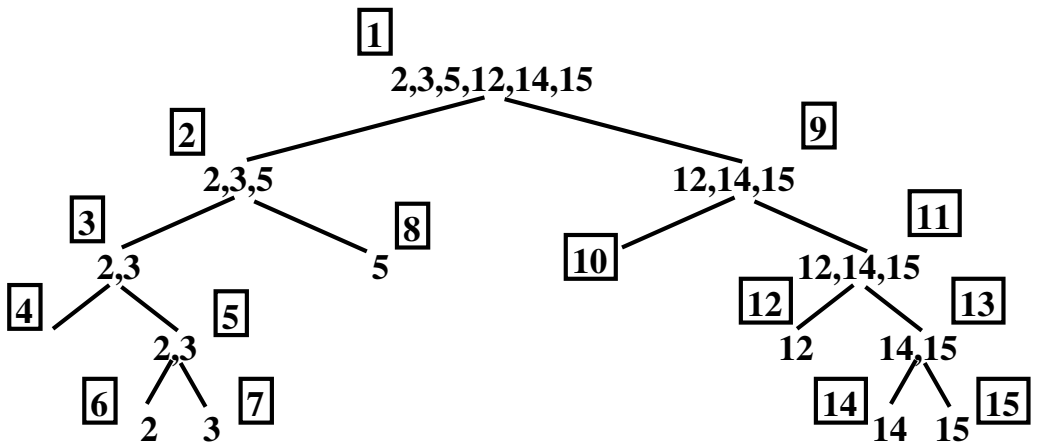
CSMA/DCR: Carrier Sense Multiple Access with Deterministic Collision Resolution

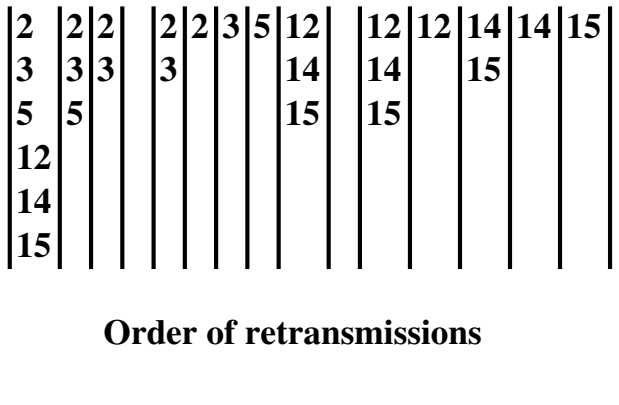
- Technology used for the Charles de Gaulle (french aircraft carrier)
- Modification of the Ethernet MAC
  - Does nothing in order to avoid collisions
  - The algorithme for the retransmission of frames is deterministic
- General description of the MAC
  - Allocation to each flow transmitted on the network
  - No collision $\Rightarrow$ classical CSMA/CD
  - Collision $\rightarrow$ the indexes are used in order to schedule the transmissions of all the pending frames
  - The retransmission algorithm is based on a virtual tree including all the indexes



CSMA/DCR

- Description of the retransmission algorithm
  - Visit the nodes of the graph in a depth-first manner
  - Transmit all the pending frames with indexes in the sub-graph of the current node
    - ★ Transmission of 0 or 1 frame ⇒ no collision ⇒ go up in the graph
    - ★ Transmission of at least 2 frames ⇒ collision ⇒ go down in the graph
- Example: pending frames with indexes 2, 3, 5, 12, 14 and 15





- The overall time for retansmission is upper bounded

RETHER: Real-time ETHERnet

- Based on a circulating token
- Can be implemented on standard Ethernet components
- No real time traffic $\Rightarrow$ the non real time traffic is not delayed
  - Basic CSMA/CD mode when there is no real time traffic
  - RETHER mode as soon as there is at least a real time traffic
- From CSMA/CD mode to RETHER mode
  - Real time request from an application $\Rightarrow$ the corresponding station (initiator) broadcats a *switch_to_rether* message
  - Every station which receives the message empties its buffer, switches to RETHER mode and sends an acknowledge
  - The initiator receives all the acknowledge messages $\Rightarrow$ it creates the token and sends it to the next station
  - More than one initiator $\Rightarrow$ the one with the smallest identifier wins (does not acknowledge the *switch_to_rether* messages of the others
- In the Rether mode
  - two sets of nodes
    - ★ RT_set: stations with at least one real time traffic
    - ★ NRT_set: all the stations

RETHER: Real-time ETHERnet

- In the Rether mode
  - MTRT (Maximum Token Rotation Time): duration for one cycle of the token
  - One cycle of the token: visit, first all the stations in the RT_set, then as many stations as possible in the NRT_set
  - A time_to_deadline field in the token
    - ★ An NRT station receives the token ⇒ it checks whether it has enough time for its transmission till the deadline

      Yes: it transmits
      it updates time_to_deadline
      it sends the token to the next NRT station

      No: it registers as first NRT station in the next cycle
      it sends the token to the first RT station
  - Example: RT_set = {1,3,6}, NRT_set = {1,2,3,4,5,6,7,8,9,10,11}

First cycle		Second cycle		...
RT	NRT	RT	NRT	...
1,3,6	1,2,3,4,5,6,7	1,3,6	7,8,9,10,11,1,2	...

  - MTHT (Maximum Token Holding Time): maximum duration a station can keep the token

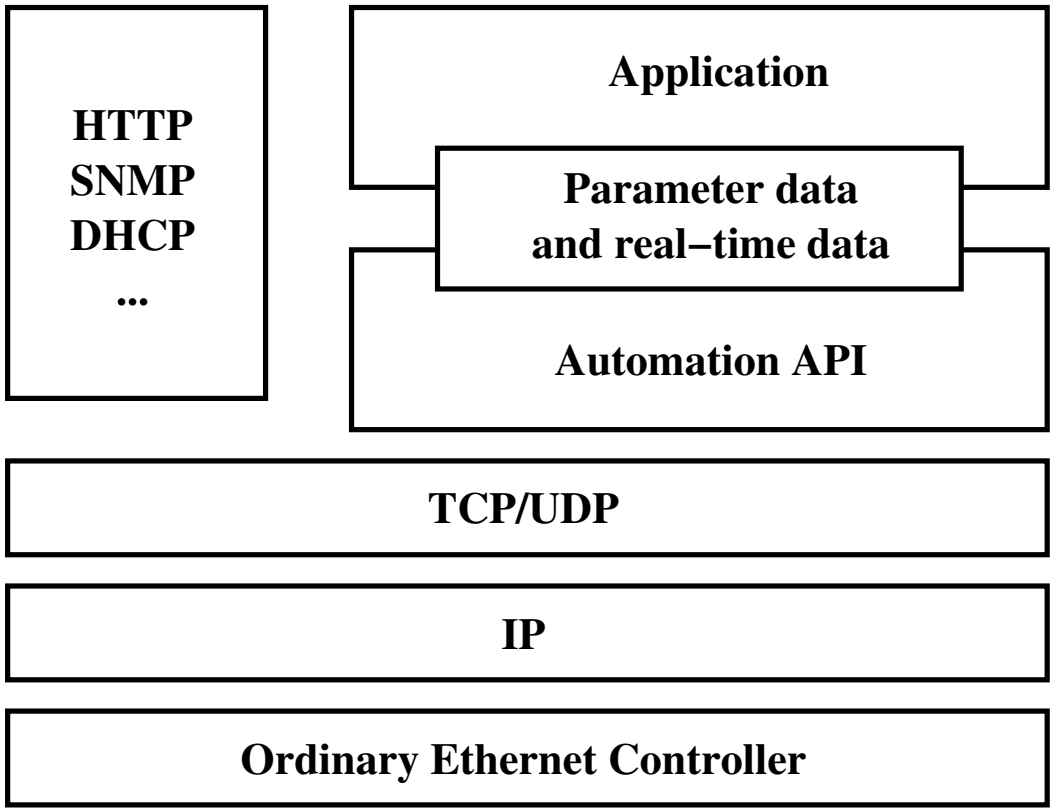
RETHER: Real-time ETHERnet

- In the Rether mode
  - A station can be added in RT_set if there is still enough time for non real time traffic
  - A node can remove itself from RT_set
  - The token is regenerated when it is lost
    - ⋆ A station $S2$ sends the token to the next station $S3$ $\Rightarrow$ it also sends it to the previous one $S1$
    - ⋆ $S1$ does not receive the token within a given time $\Rightarrow$ it regenerates the token and sends it to $S3$
- From RETHER mode to CSMA/CD mode
  - As soon as the RT_set becomes empty
  - The last station which removes from RT_set broadcasts a *switch_to_csma* message
  - A station receives the message $\Rightarrow$ it switches to CSMA mode
  - A NRT station does not receive the token within a given time $\Rightarrow$ it switches to CSMA mode

Industrial Ethernet technologies

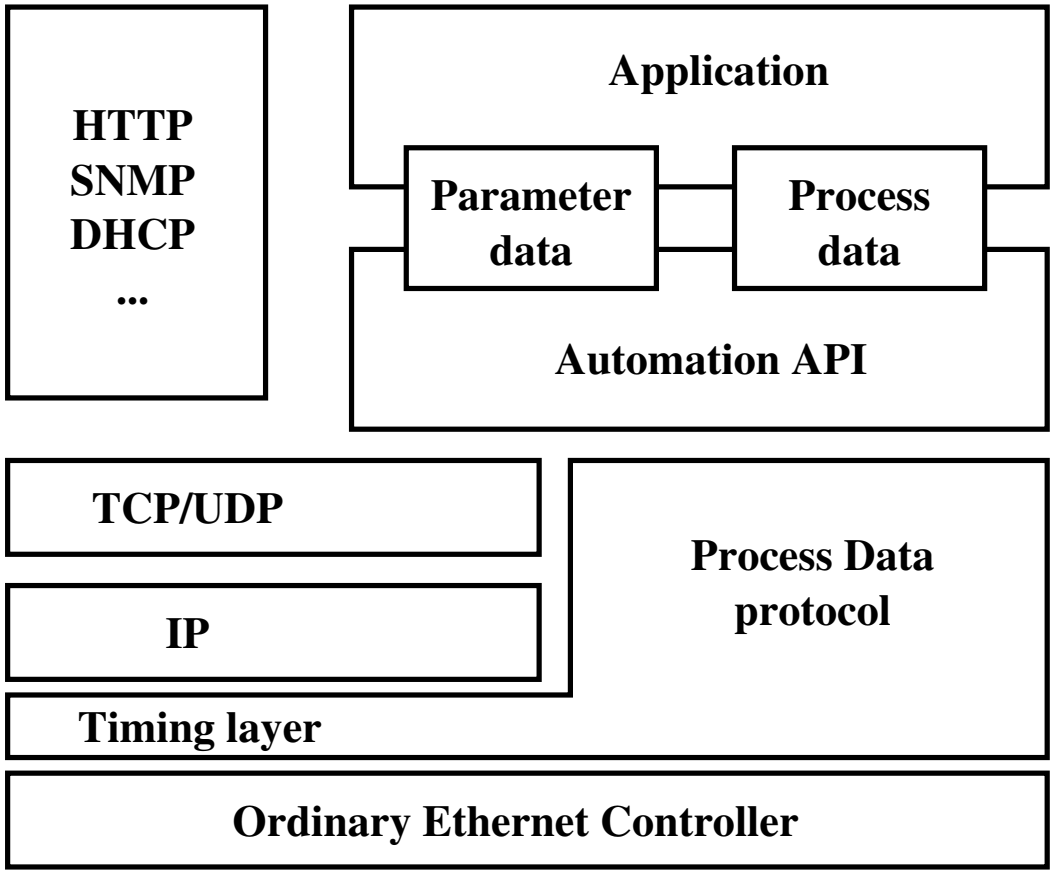
- Very different solutions, depending on the timing constraints (1 ms, 10 ms, 100 ms)
- The solutions can be classified
- Presentation of a small subset of the different solutions

Class A: the best effort approach



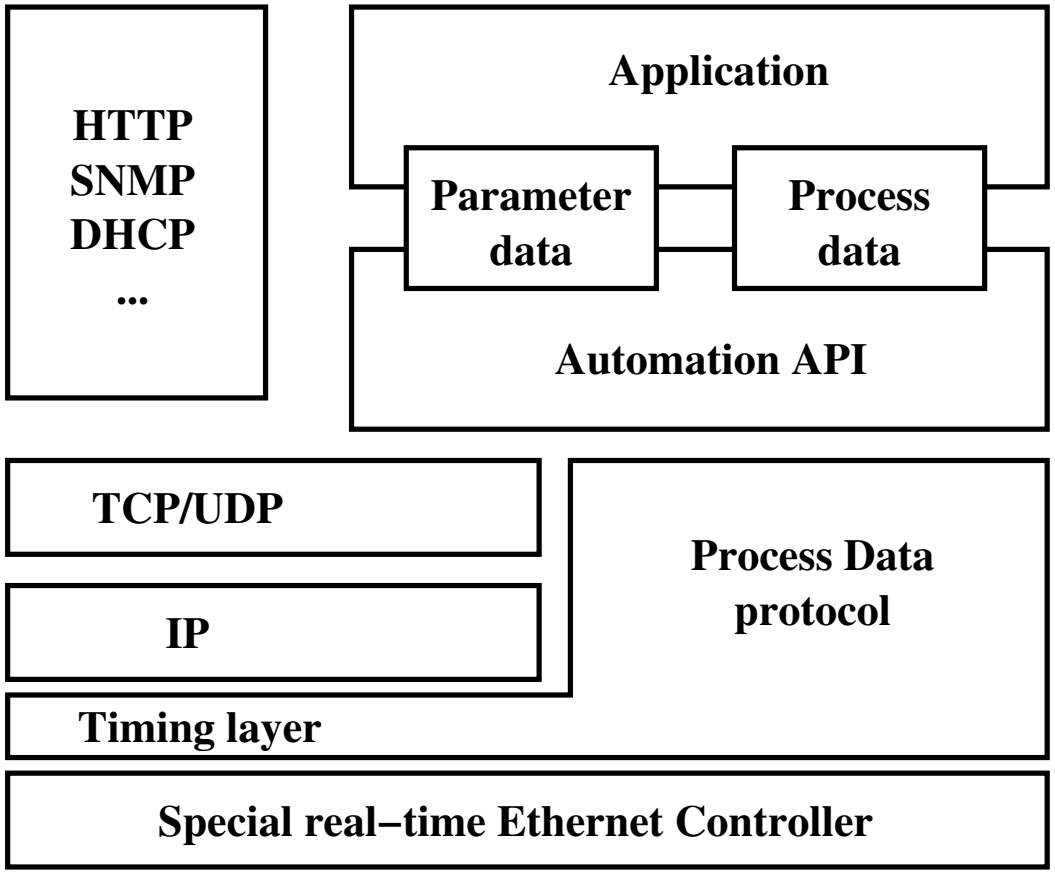
- Completely TCP/UDP/IP based
- Ordinary Ethernet controllers and switches
- Modbus/TCP, Ethernet/IP, PROFINET-CBA

Class B: the soft real-time approach



- Process Data: parallel channel to TCP/UDP/IP
- TCP/UDP/IP timing controlled by process data driver
- Ordinary Ethernet controllers and switches (or hubs)
- Ethernet Powerlink, PROFINET RT

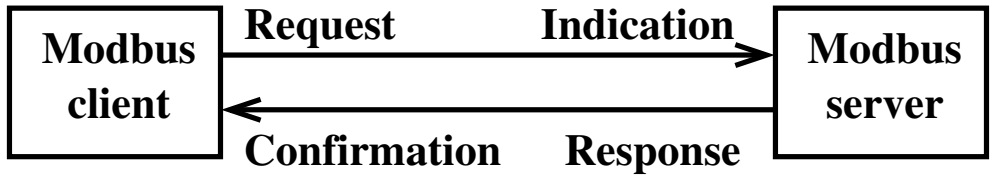
Class C: the hard real-time approach



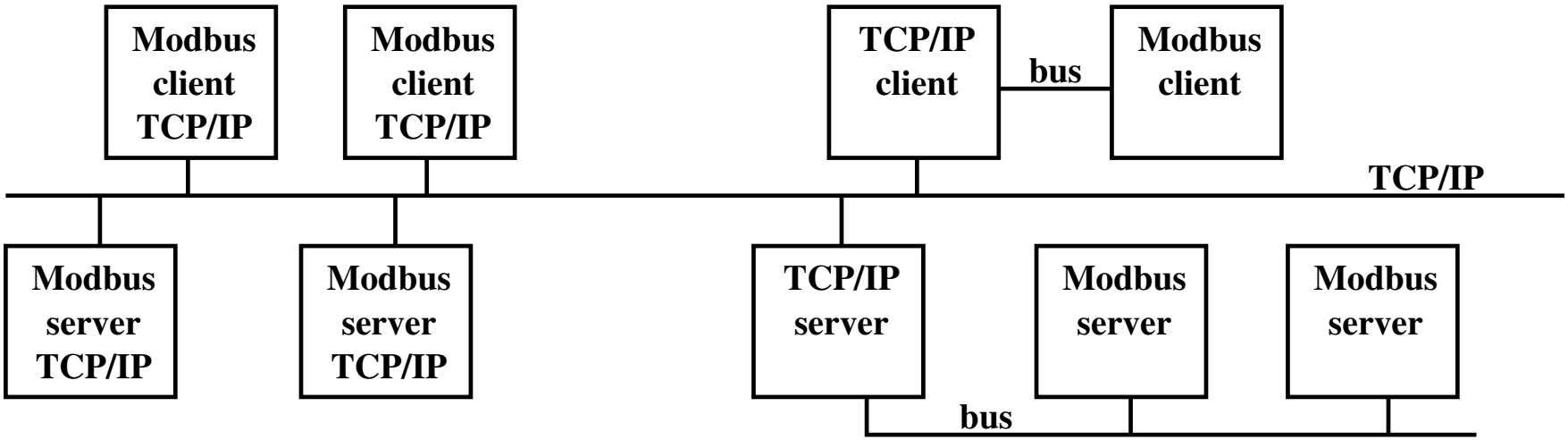
- Process Data: parallel channel to TCP/UDP/IP
- TCP/UDP/IP timing controlled by process data driver
- Special Ethernet controllers or switches
- PROFINET IRT, SERCOS III, EtherCAT

Modbus/TCP (best-effort approach)

- Schneider Electric approach: modbus on TCP/IP
- Few services, simple to implement
- Widely used, many products available
- Client-server approach

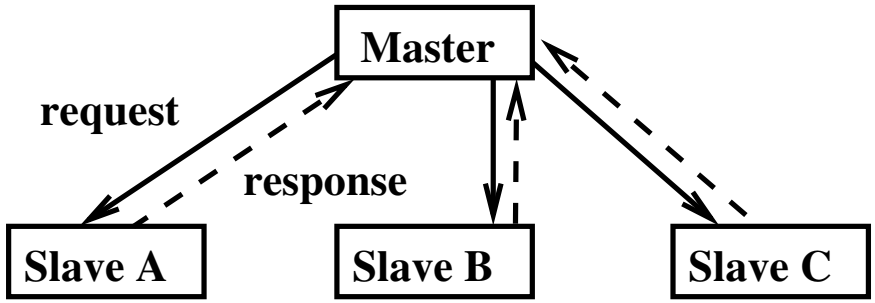


- General architecture

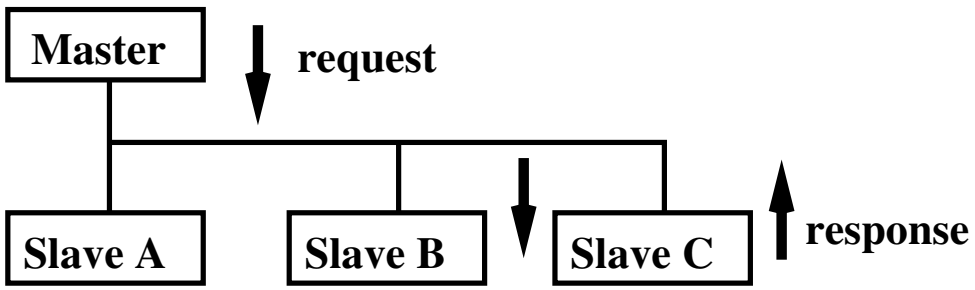


The MODBUS protocol

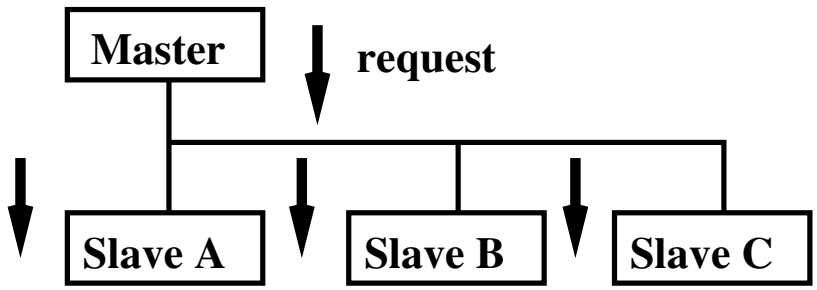
- First used in automation in 1979 by Modicon
- Very simple master/slave fieldbus



- The master sends a request to a slave and the slave answers
- An address for each slave, between 1 and 64
- Two kinds of transmission



Master to one slave



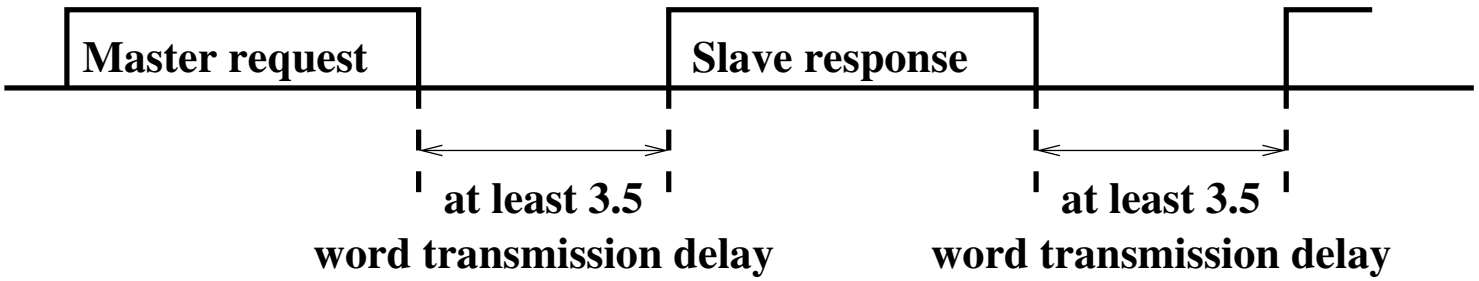
Master to all slaves

The MODBUS protocol

- The request frame format
  - Slave number (one byte): number of the addressed slave
  - Function code (one byte): type of request
    - 19 function codes are defined
    - 0x03: read registers, 0x10: write registers, ...
  - Parameters of the request (up to 256 bytes): the parameters of the request (e.g. the registers to be read)
  - CRC (two bytes): detection of errors of transmission
- The response frame format
  - Slave number (one byte): number of the addressed slave
  - Function code (one byte): type of request (e.g. read registers), the most significant bit is one when there is an error
  - Answer to the request (up to 256 bytes) or exception code (one byte)
  - CRC (two bytes): detection of errors of transmission
- Each byte begins with a start bit and ends with a stop bit

The MODBUS protocol

- Scheduling of frames



- a Typical message exchange
  - From master to slave

04	03	00 02	00 01	25 CA
Dest	Read Reg	No 1st Reg	Nb of Reg	CRC

  - From slave to master with error

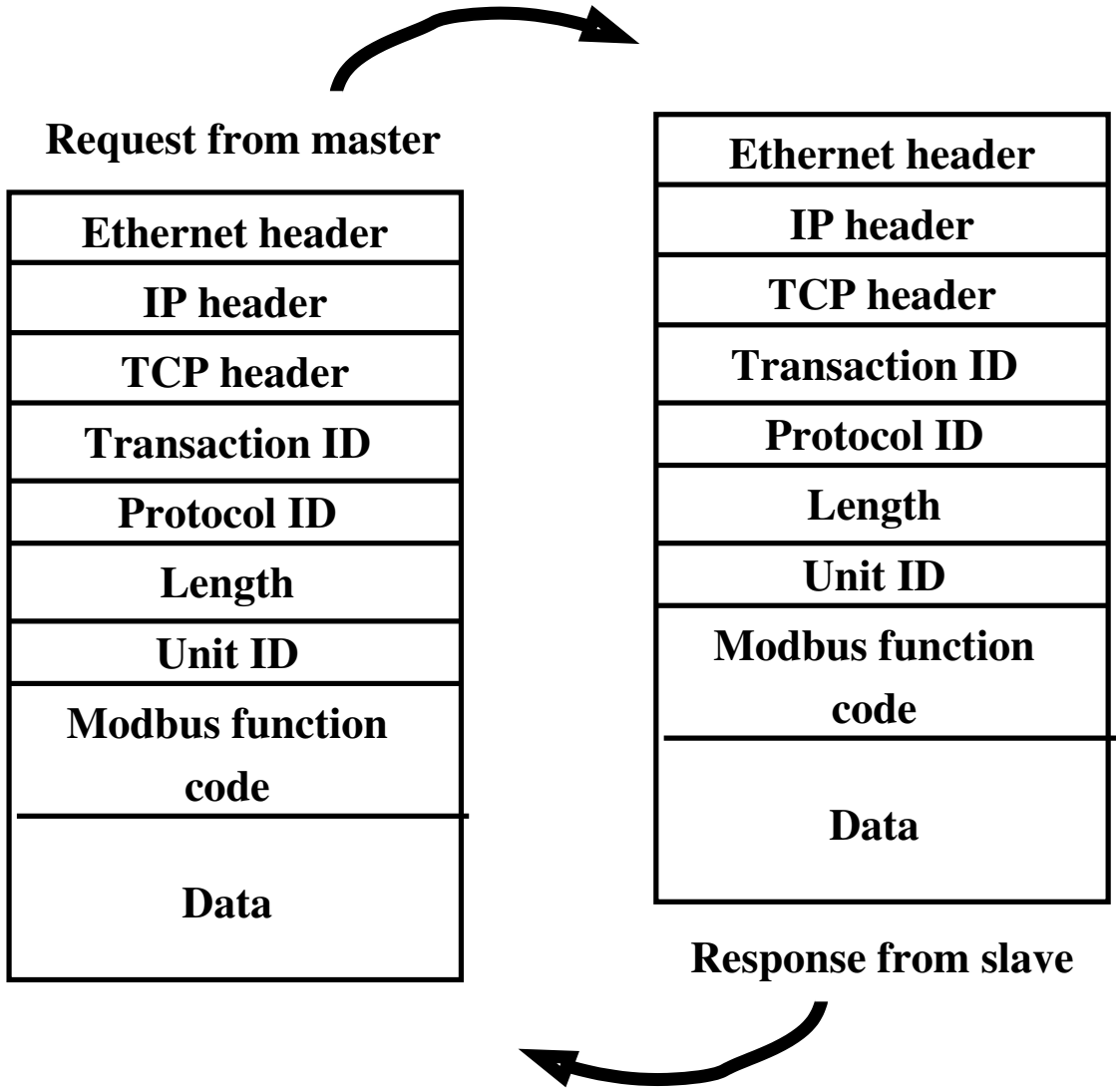
04	83	02	01 31
Dest	Error	No Reg	CRC

  - From slave to master without error

04	03	02	02 58	B8 DE
Dest	Read Reg	Nb bytes	Data	CRC

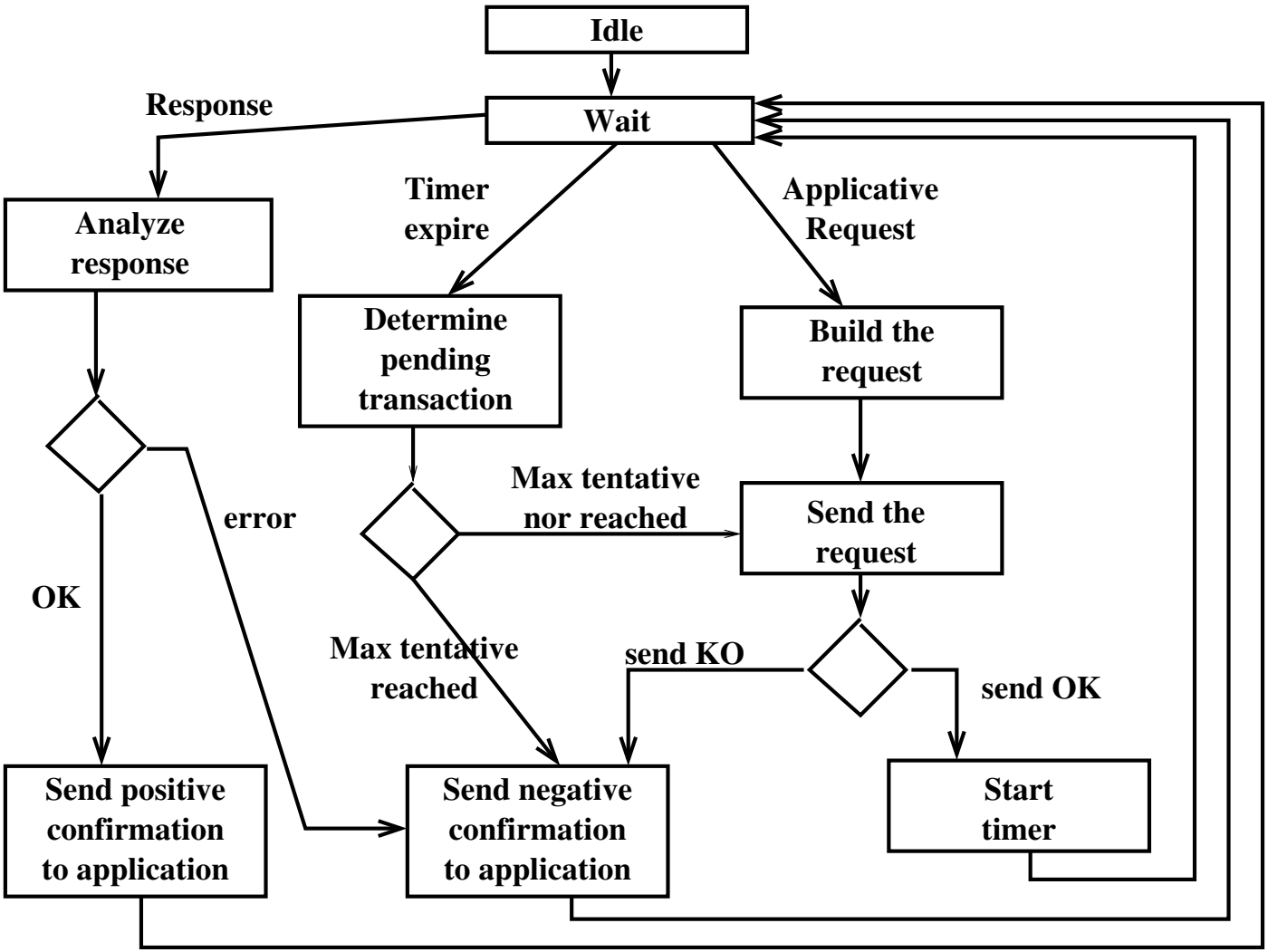
Modbus/TCP (best-effort approach)

- Frame format



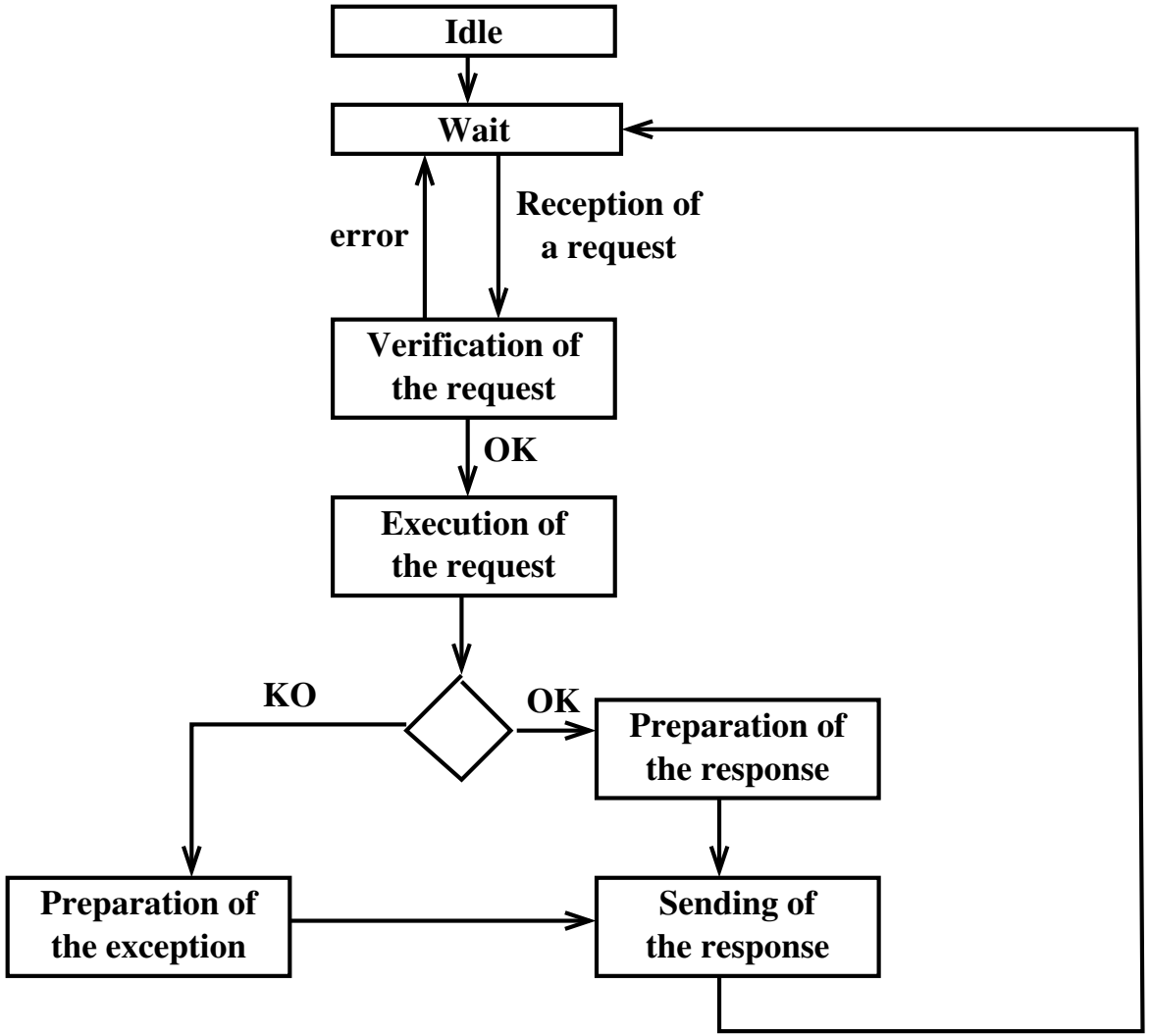
Modbus/TCP (best-effort approach)

- Diagram of Modbus client



Modbus/TCP (best-effort approach)

- Diagram of Modbus server

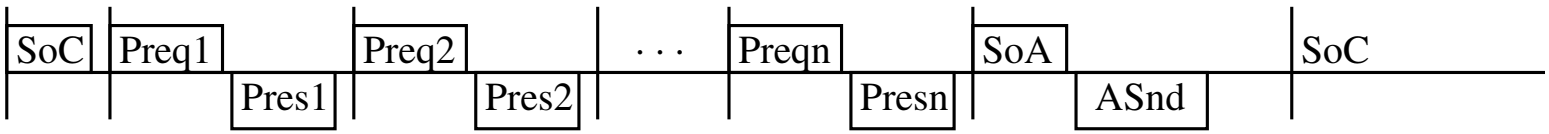


Ethernet Powerlink (soft real time approach)

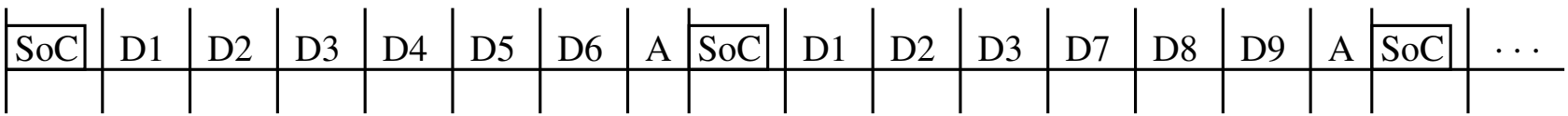
- Open protocol from the Ethernet Powerlink Standardization group
- Extension of Ethernet using polling and time slots
- Critical data are transmitted during very short isochronous cycles
- Precise synchronization between the different nodes
- Less critical data are transmitted during the asynchronous phase
- Basic cycle with a given duration, which includes
  - A start of cycle frame (SoC) for the synchronization of the nodes, emitted by the managing node (MN)
  - The isochronous phase where the MN sends Poll Request frames and each addressed node answer with a Poll Response frame
  - The asynchronous phase where the MN sends a Start of Asynchronous (SoA) frame and the addressed node answer with a ASnd frame, where he can send ad-hoc data
- Possibility to multiplex several critical data in the same slot

Ethernet Powerlink (soft real time approach)

- Cycle overview

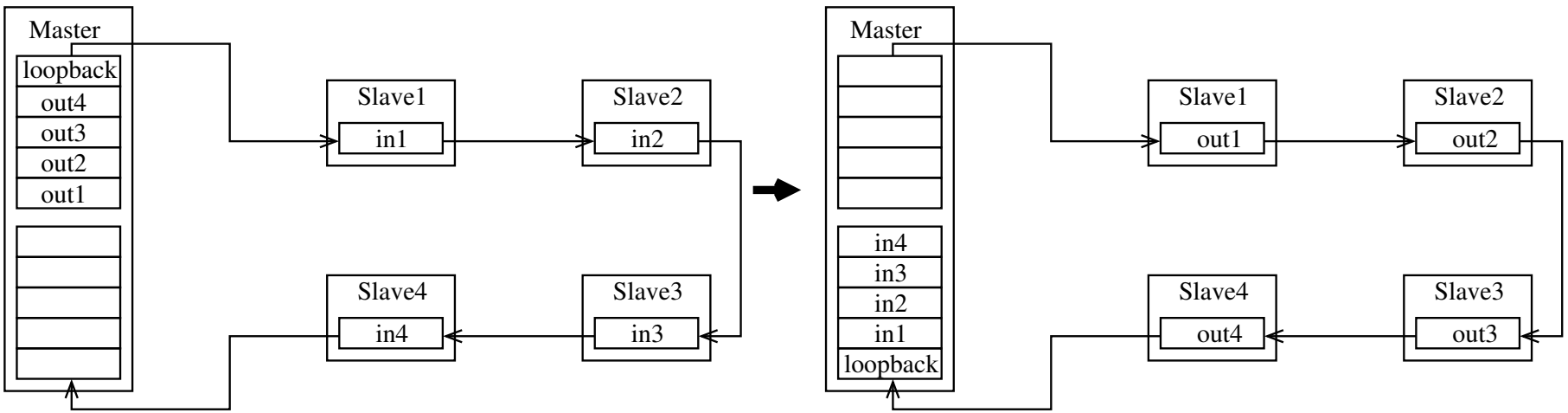


- Data multiplexing



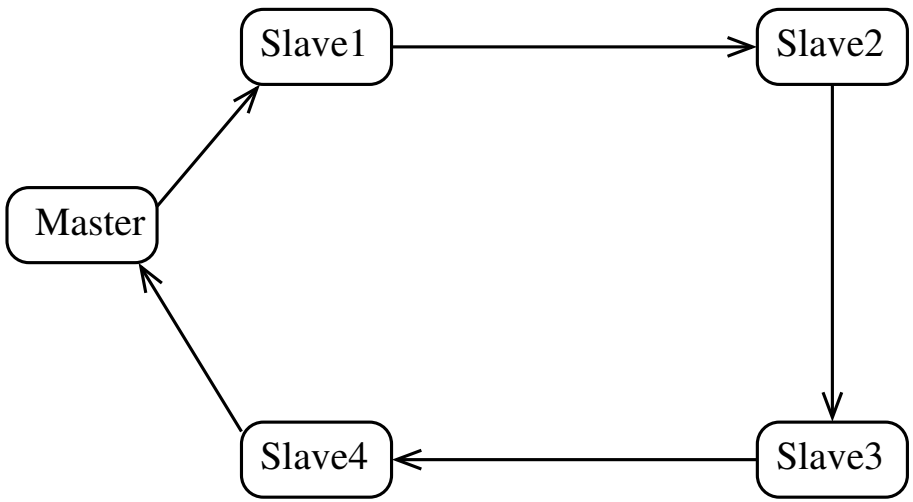
Ethercat (hard real time approach)

- Modification of Ethernet MAC
- Application of the Interbus principle in the context of Ethernet
  - Interbus: Summation frame with all the data
    - ★ From the master to the slaves
    - ★ From the slaves to the master



Ethercat (hard real time approach)

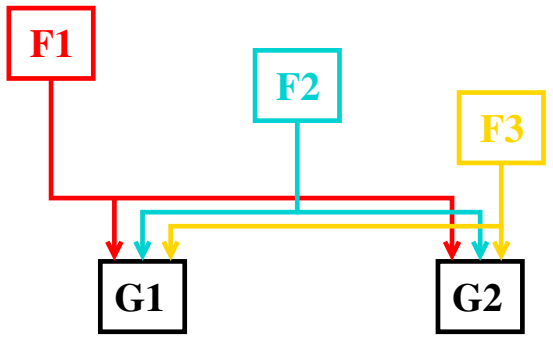
- Master/slave architecture, in a ring topology
- The master sends a single Ethernet frame (telegram)
- Each slave processes the frame on the fly
  - The processing is done by hardware
  - Each station can extract data from the frame
  - Each station can write data in the frame



telegram

Classical avionics (up to Airbus A340)

- A network of equipments



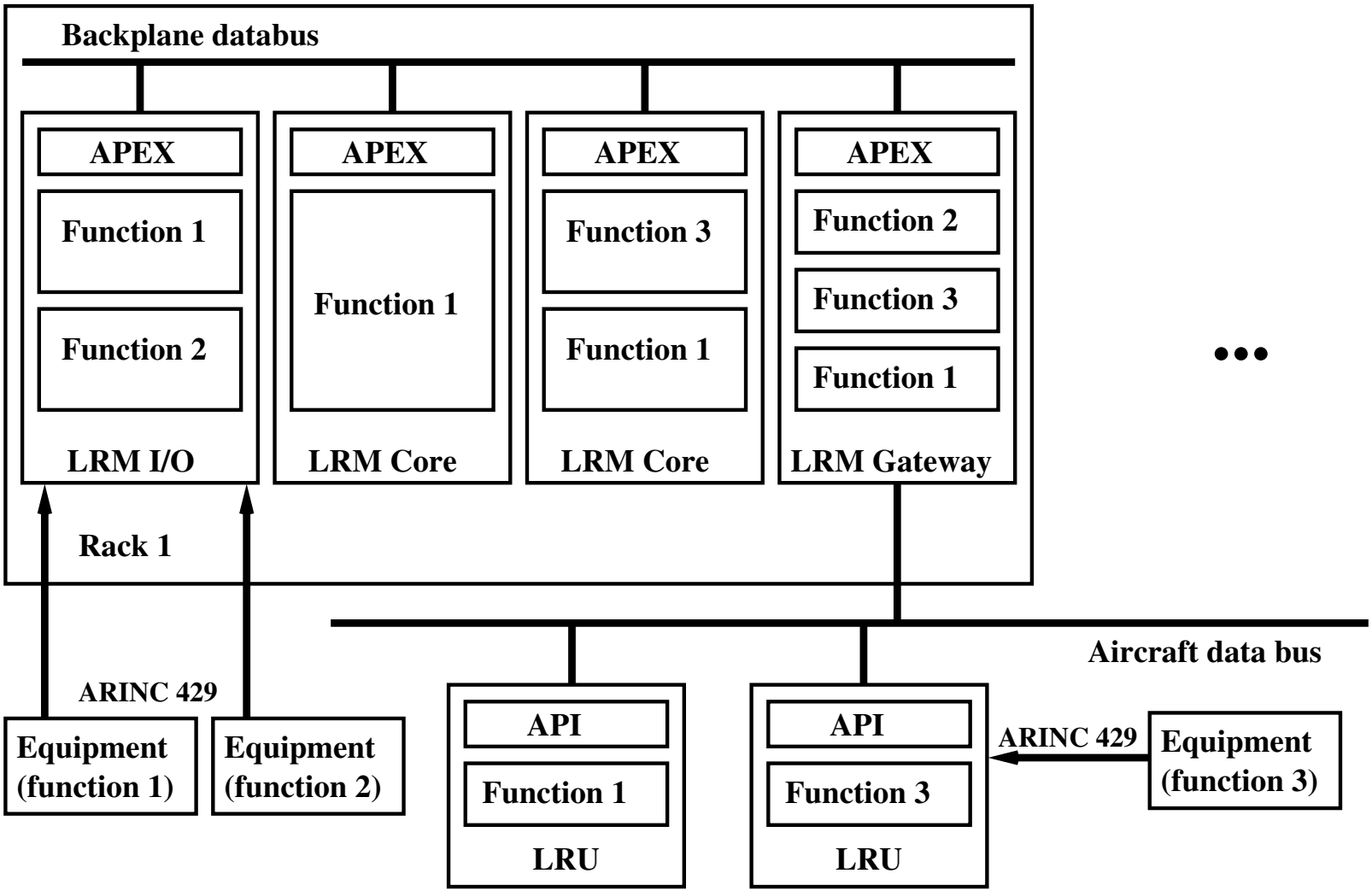
- Each data is transmitted from its source to every equipment that needs the data
- One dedicated line for each data
- Repetitive transmission of the data on each line
- Each line: a mono-emitter ARINC 429 data bus
- Each data individually identified by a label
- Low bit rate: 12 Kbits/s to 100 Kbits/s
- At most 20 receivers per line

The ARINC 429 main characteristics

- Unidirectional data bus standard
- Transmission of messages over two twisted pairs
- The ARINC 429 data word

  - P (parity bit): so that there is an odd number of "1" in the 32-bit word
  - SSM (Sign/Status Matrix): contains equipment condition, operational mode or validity of data contents, depending on the format of the data
  - DATA: the data, with a number of different formats
    - ★ Binary Coded Decimal (BCD)
    - ★ Two's complement binary notation (BNR)
    - ★ ...
  - SDI (Source Destination Identifier): different possibilities
    - ★ Identification of the receiver for which the data is destined
    - ★ Identification of the source of the transmission if the label can be sent on different data buses
  - LABEL: identification of the data and its associated parameters

IMA architecture (ARINC 651)

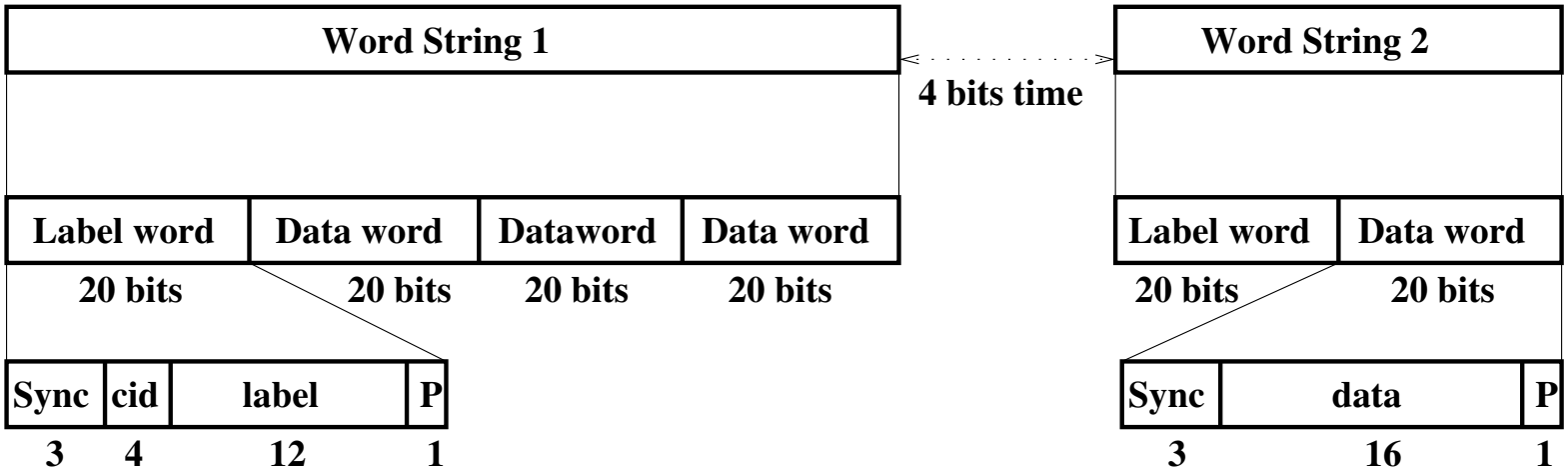


IMA characteristics

- Hardware components
  - LRM (Line Replaceable modules): resources in common cabinets
    - ★ Core modules for the execution of the applications
    - ★ Input/output modules for communications with non-IMA equipments
    - ★ Gateway modules for communications between cabinets
  - LRU (Line Replaceable Unit): existing non-IMA equipments
  - Backplane databus: ARINC 659
- Sharing of execution resources
  - An avionics subsystem: a partition with an assigned time window to execute its application on a shared module
  - Guarantee isolation between subsystems $\Rightarrow$ robust partitioning concept
    - ★ Spatial isolation: limitation and protection of the address space of each partition, e.g. by a MMU
    - ★ Temporal isolation: static allocation of a time slice for the execution of each partition, based on WCET
  - Communications between partitions via ports (APEX: APplication EXecutive)
    - ★ Sampling ports: only the last value of data is stored
    - ★ Queueing ports: all the values of data are stored
  - Logical channel: multicast link between ports, independent of the communication technology

ARINC 629 general characteristics

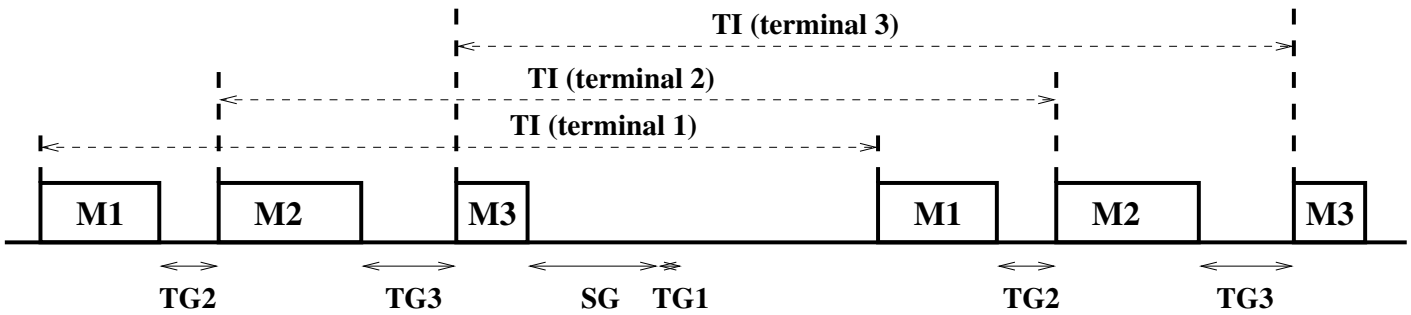
- Transmission of messages at a rate of 2 to 8 Mbits/s
- General structure of a message



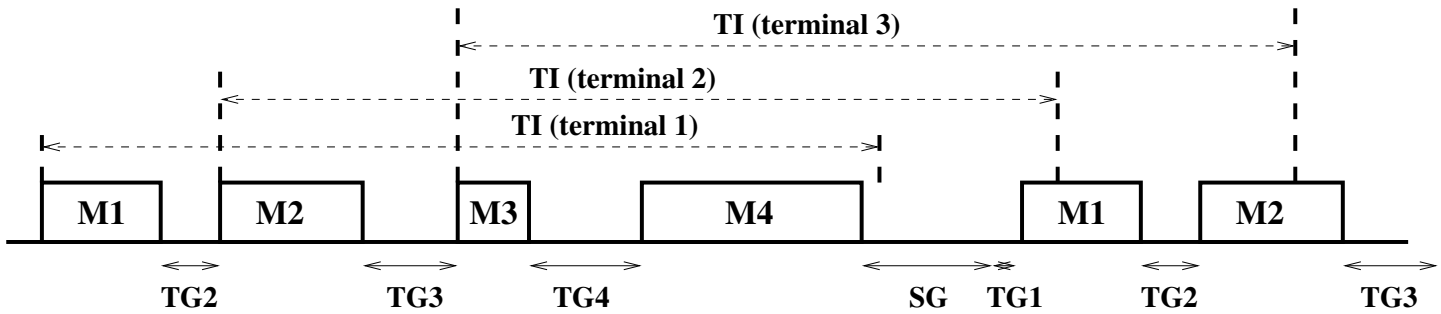
  - Up to 31 word strings in a message
  - One word string: one label word and up to 256 data words
  - Sync: 3 bits for synchronization
  - P: 1 parity bit
  - Cid (Channel Identification): differentiate signals from different equipments with the same label
- Two incompatible alternatives data-link level protocols (CSMA-CA)
  - The basic protocol
  - The combined protocol

ARINC 629 Basic Protocol (BP)

- TGi: Terminal Gap for which terminal i must wait after bus activity before starting its own transmission
- TI: Transmit Interval for which each terminal must wait between its own successive transmissions
- SG: Synchronization Gap used for synchronizing all the terminals for the beginning of a new cycle
- Periodic mode when TI is large enough

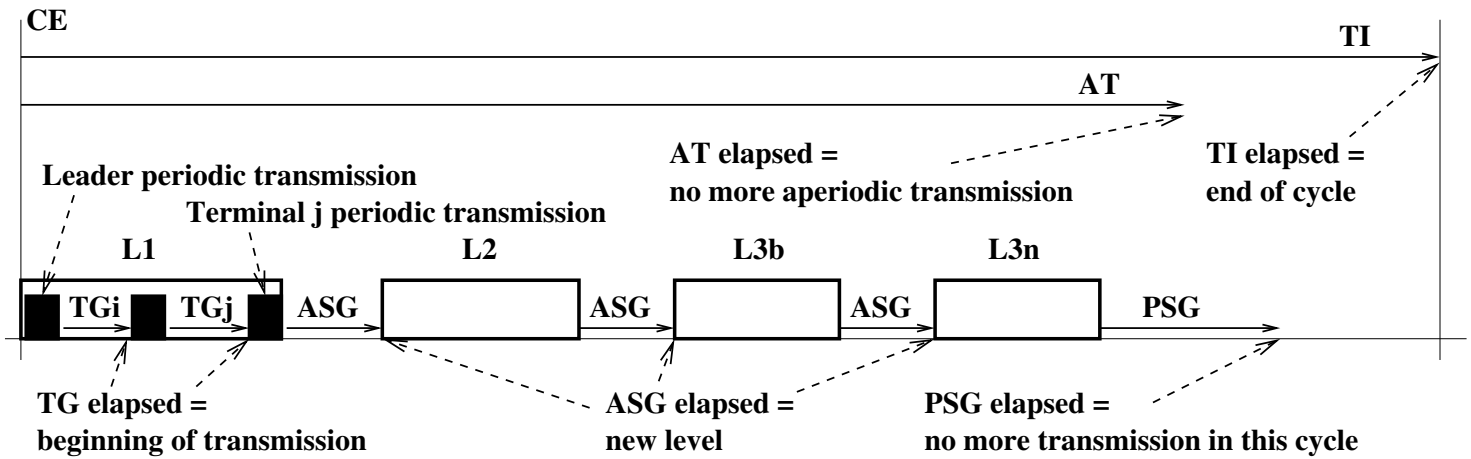


- Aperiodic mode when TI is too small



ARINC 629 Combined Protocol (CP)

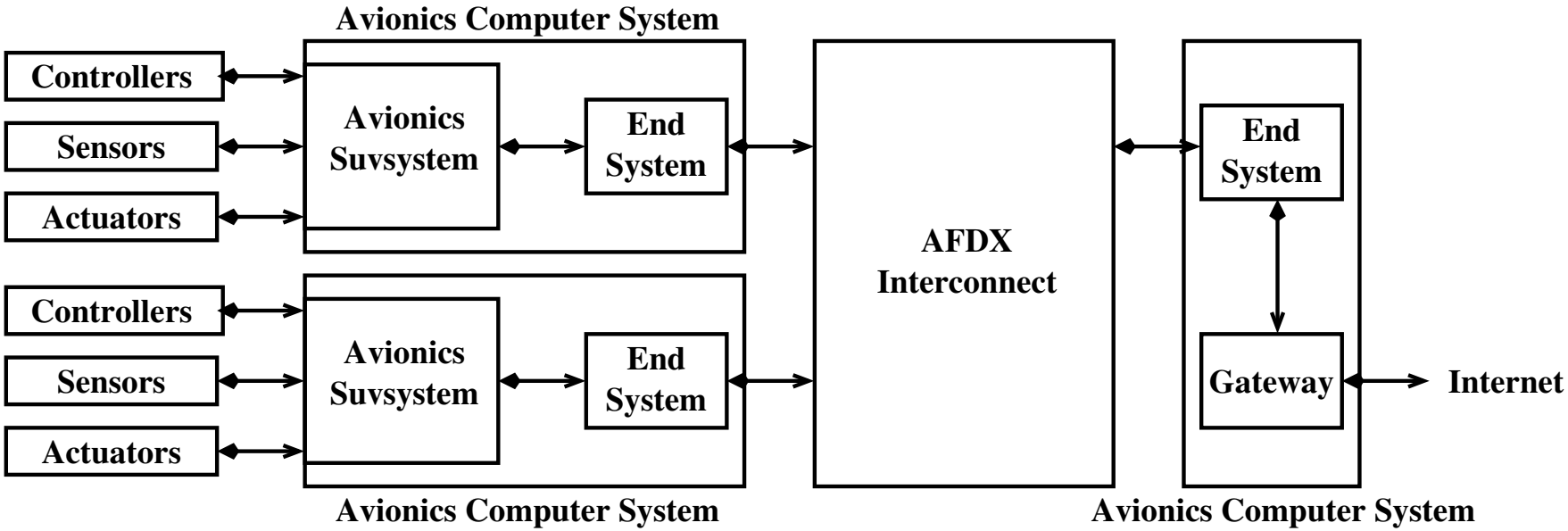
- Combination of periodic and aperiodic data transmissions
- Level 1 (L1): periodic transmissions
- Level 2 (L2): shorter (higher priority) aperiodic transmissions
- Level 3 (L3): longer (lower priority) aperiodic transmissions
- TI only applied to the first periodic transmission $\Rightarrow$ Concatenation event (CE) to cancel all unelapsed TI timers
- Periodic Synchronization Gap (PSG) for cycle synchronization
- Synchronization Gap (ASG) for transition between levels
- Aperiodic Time-out (AT) to avoid cycle overflow
- At least level 1 and level 2 messages in a cycle
- Level 3 messages can be delayed



The reasons for the AFDX

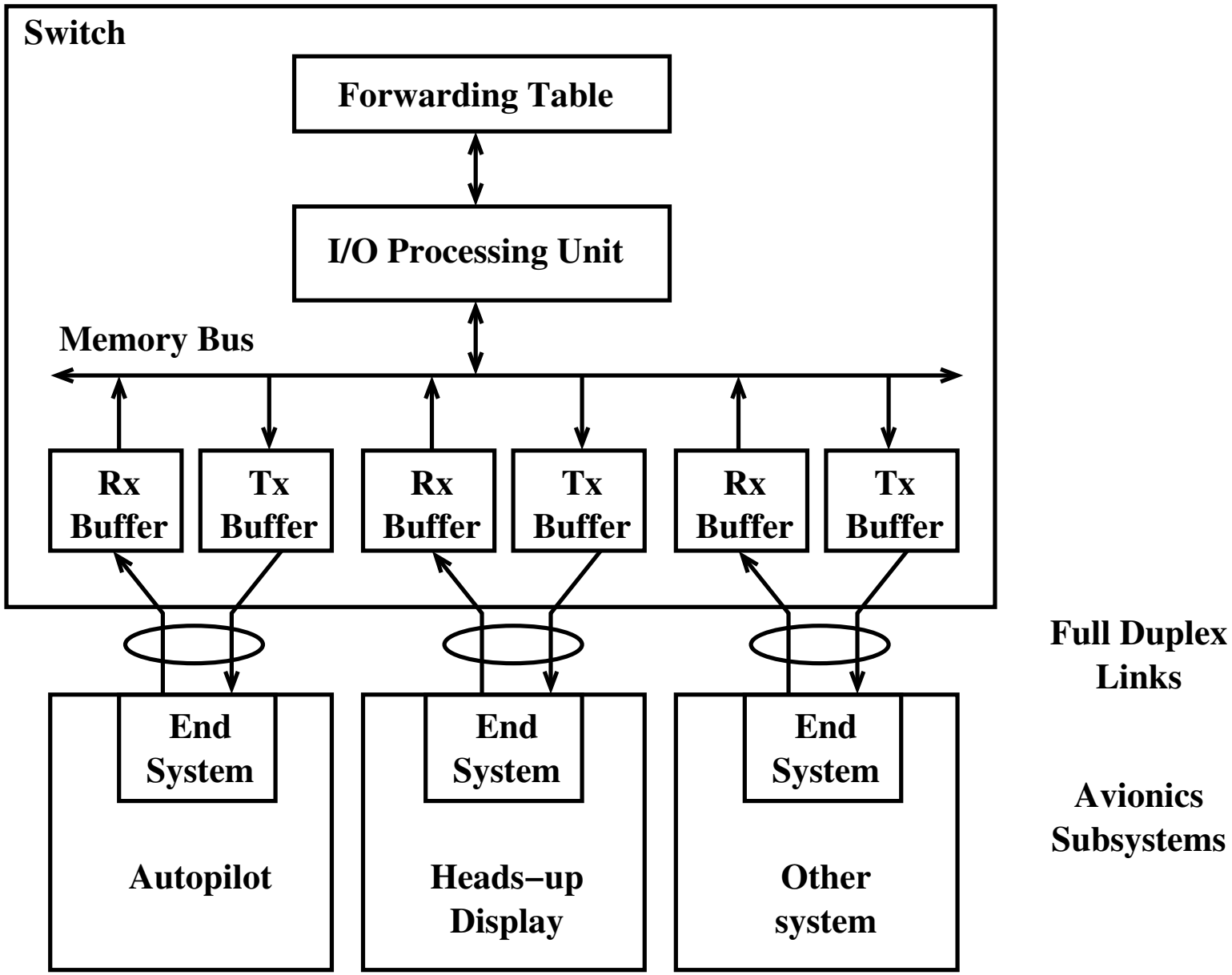
- ARINC 429 is no more sufficient: too many buses are needed
- ARINC 629 is too expensive
- Why Ethernet technology ?
  - High throughput offered to the connected units (100 Mbits/s)
  - High connectivity offered by the network structure
  - A mature industrial standard
  - Low connection cost
- But Ethernet CSMA/CD is not deterministic
  - Potential collision on the physical medium
  - Binary Exponential Back off retransmission algorithm
- The solution adopted for ARINC 664
  - switched Ethernet technology: units directly connected by point-to-point links to Ethernet switches $\Rightarrow$ possible collision domain = single link between two elements
  - Full duplex links $\Rightarrow$ no more collisions
- The problem is shifted to the switch level

AFDX general architecture



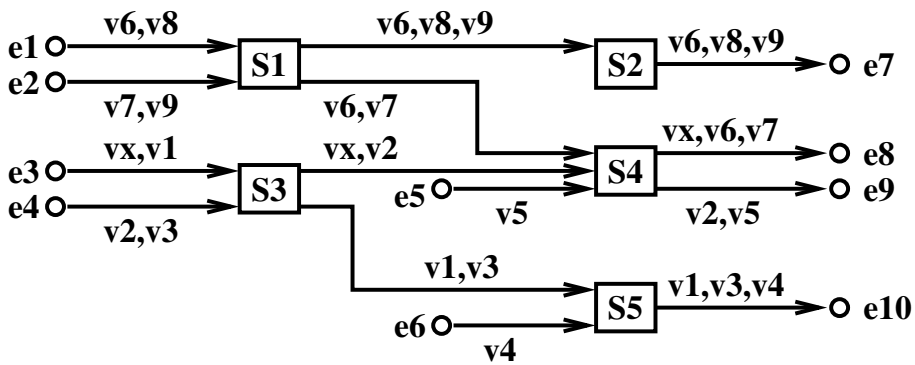
- Avionics Subsystem: Flight control computer, Global Positioning System, tire pressure monitoring system, ...
- End System: interface between the Avionics Subsystems and the AFDX, guarantees a secure and reliable data interchange
- AFDX Interconnect: network of switches with full duplex links
- Gateway: communication path between the Avionics Subsystems and the external IP network $\Rightarrow$ data loading, logging

Full duplex switched Ethernet example

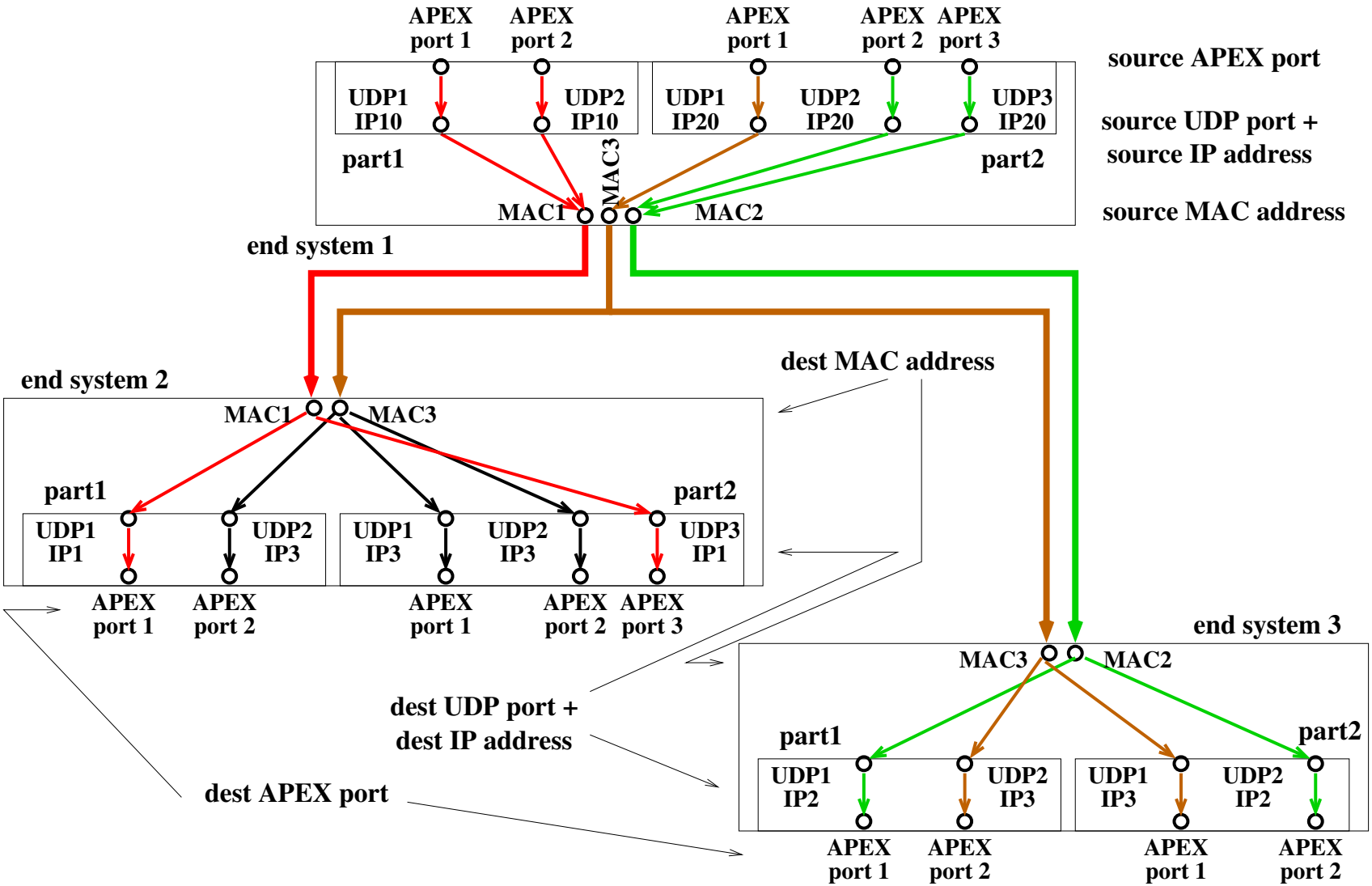


The AFDX key characteristics

- Static full duplex Switched Ethernet: no CSMA/CD, no spanning tree, static 802.1D tables
- One FIFO buffer per output port
- Traffic characterization based on the Virtual Link (VL) concept
  - Static definition of the flows which enter the network
  - Multicast path with deterministic routing
  - Mono transmitter assumption
  - Sporadic flows (BAG: Bandwidth Allocation Gap)
    - ★ Discrete values: 1, 2, 4, 8, 16, 32, 64, 128 ms
    - ★ Traffic shaping on each emitting end system
  - Minimum (S_{min}) and maximum (S_{max}) frame lengths for each VL
  - BAG and S_{max} define the maximum bandwidth allocated to a VL
    - ★ Example: $BAG = 16$ ms and $S_{max} = 128$ bytes $\Rightarrow$ 64000 bits/s
  - Scheduling of VLs on end systems $\Rightarrow$ (bounded) jitter

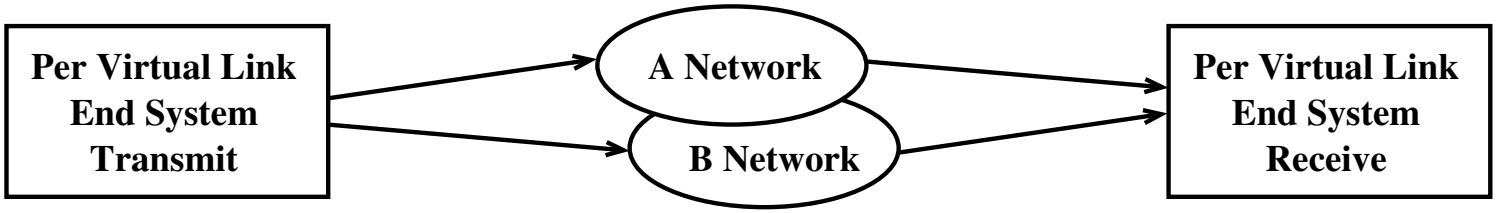


AFDX VL integration

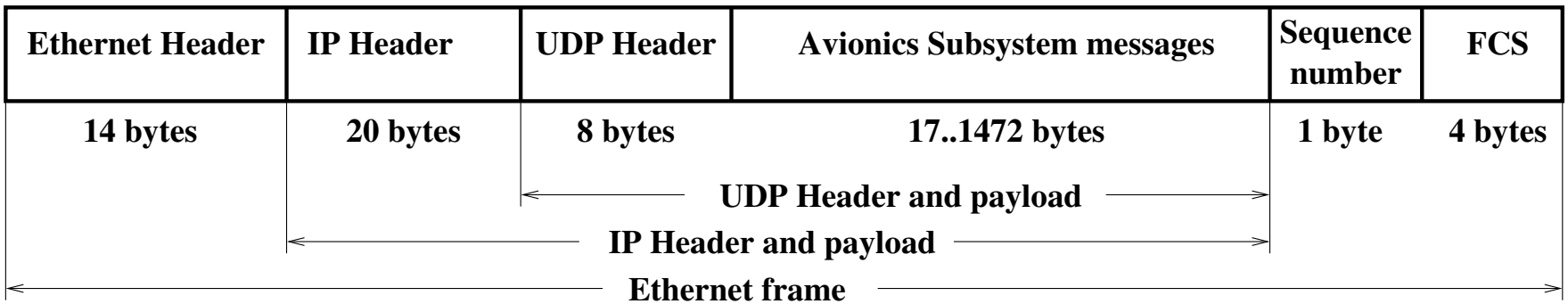


The redundancy management in the AFDX

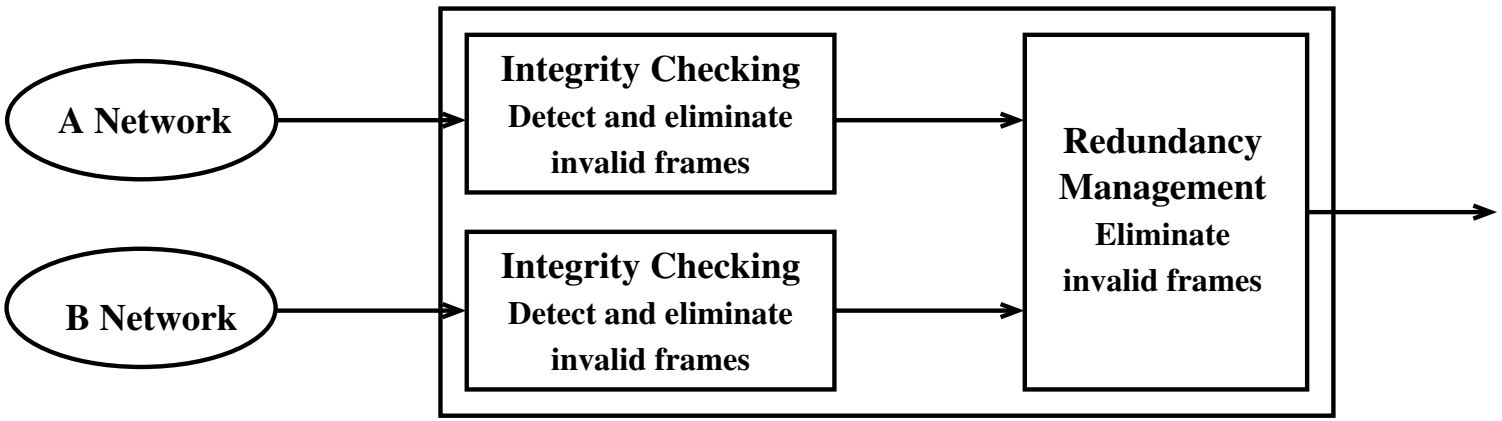
- Frames transmitted on two independent networks



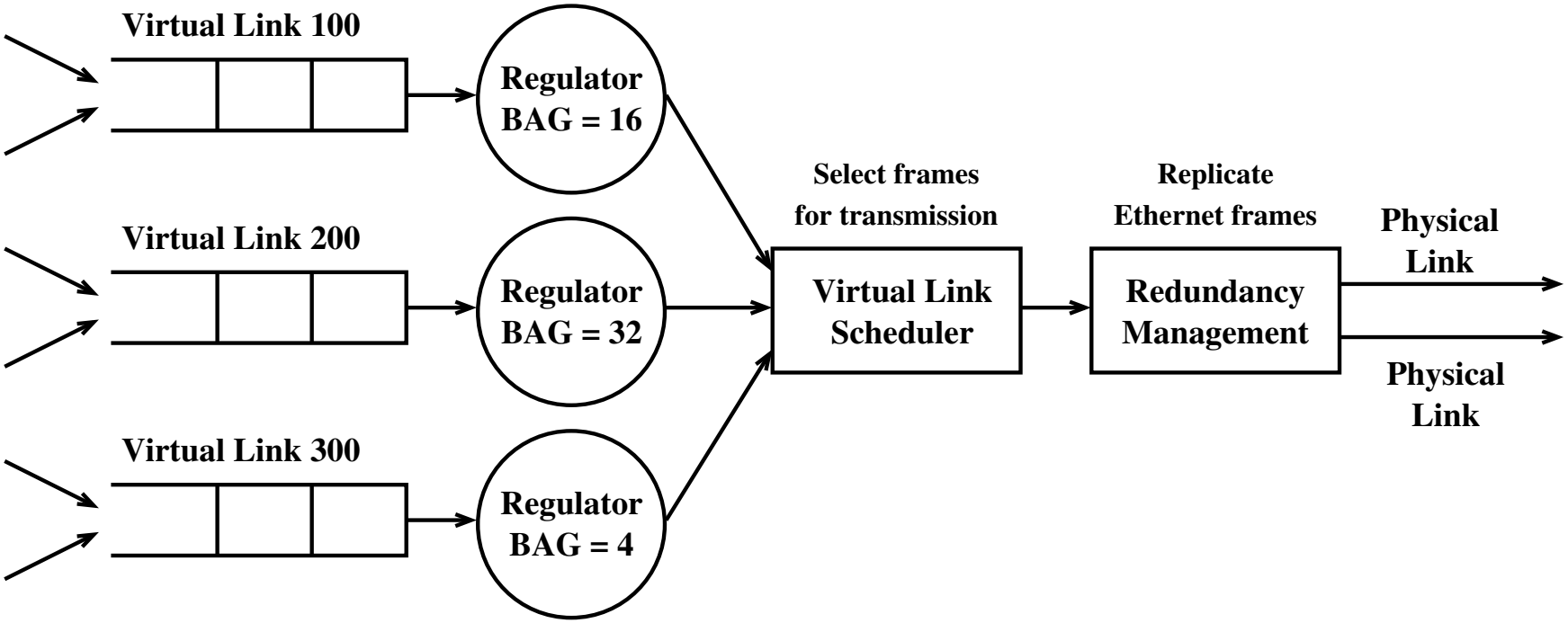
- Identification of replicas via a sequence number

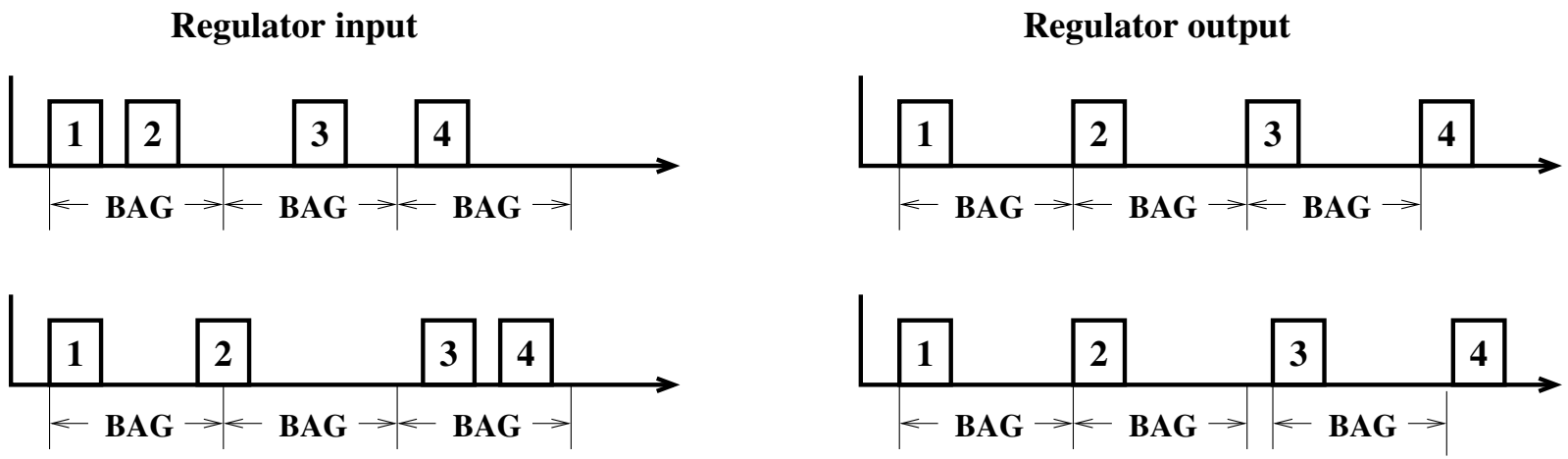


- Elimination of the second received frame

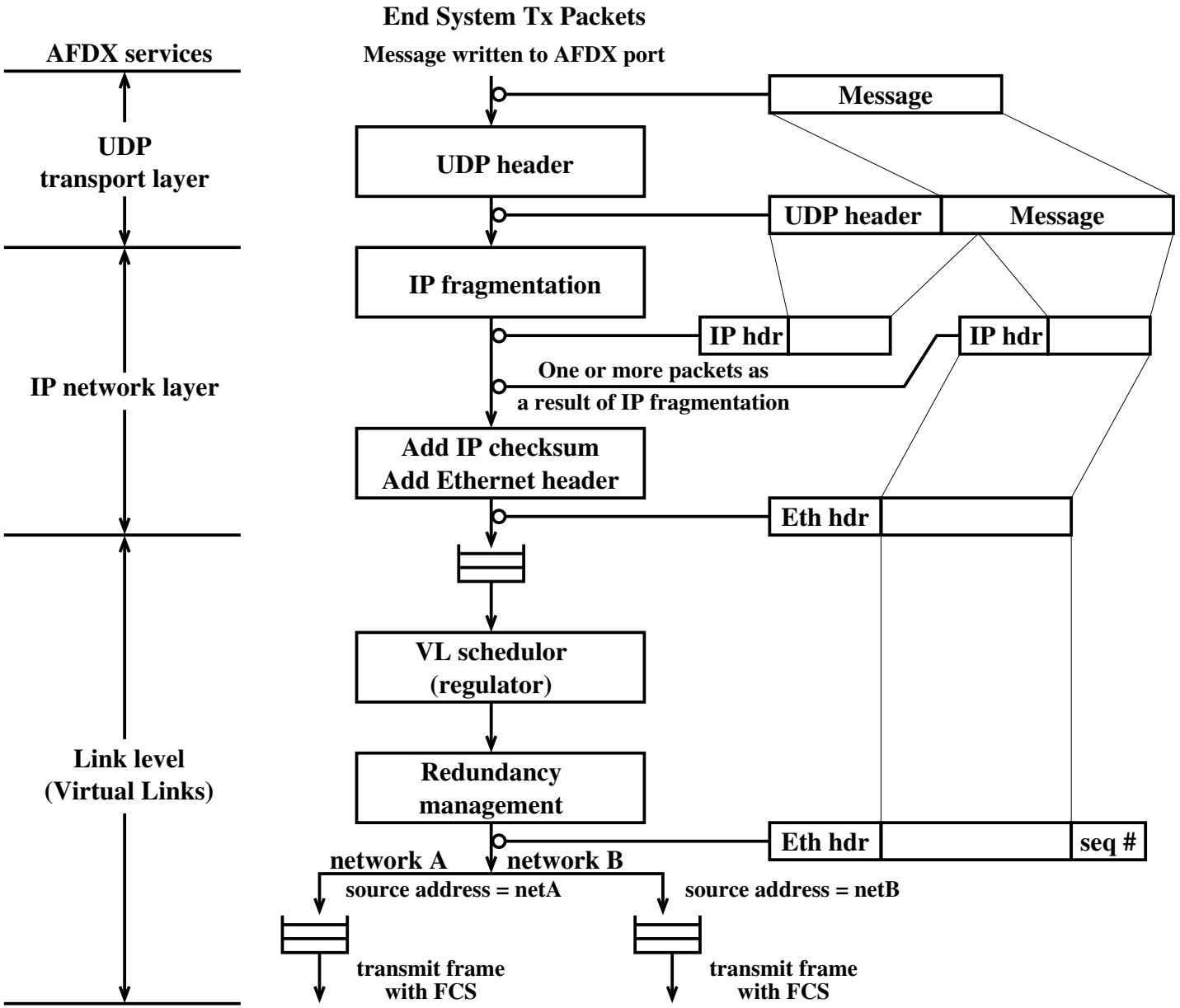


The packet regulation of Virtual Links

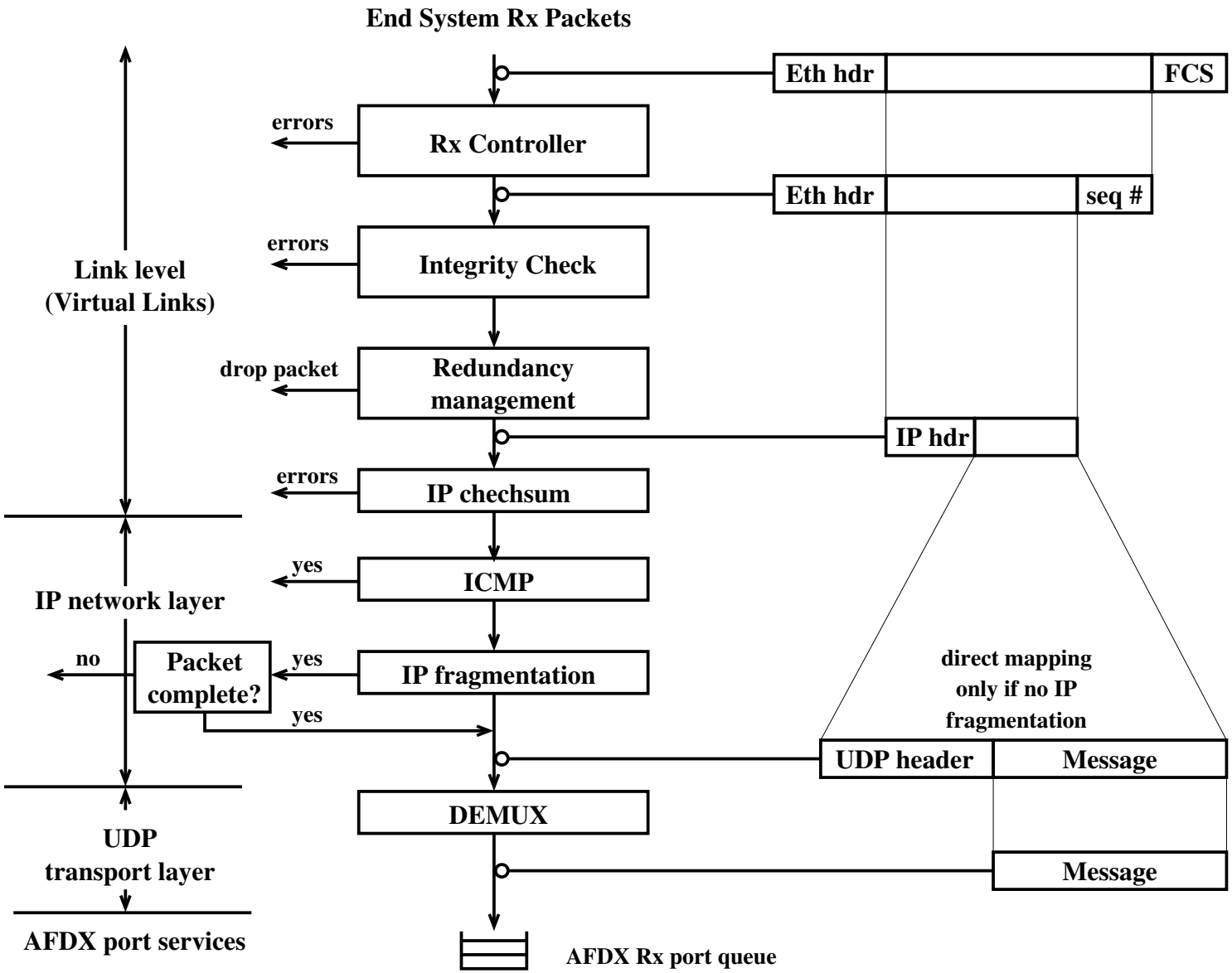




The AFDX transmit protocol stack

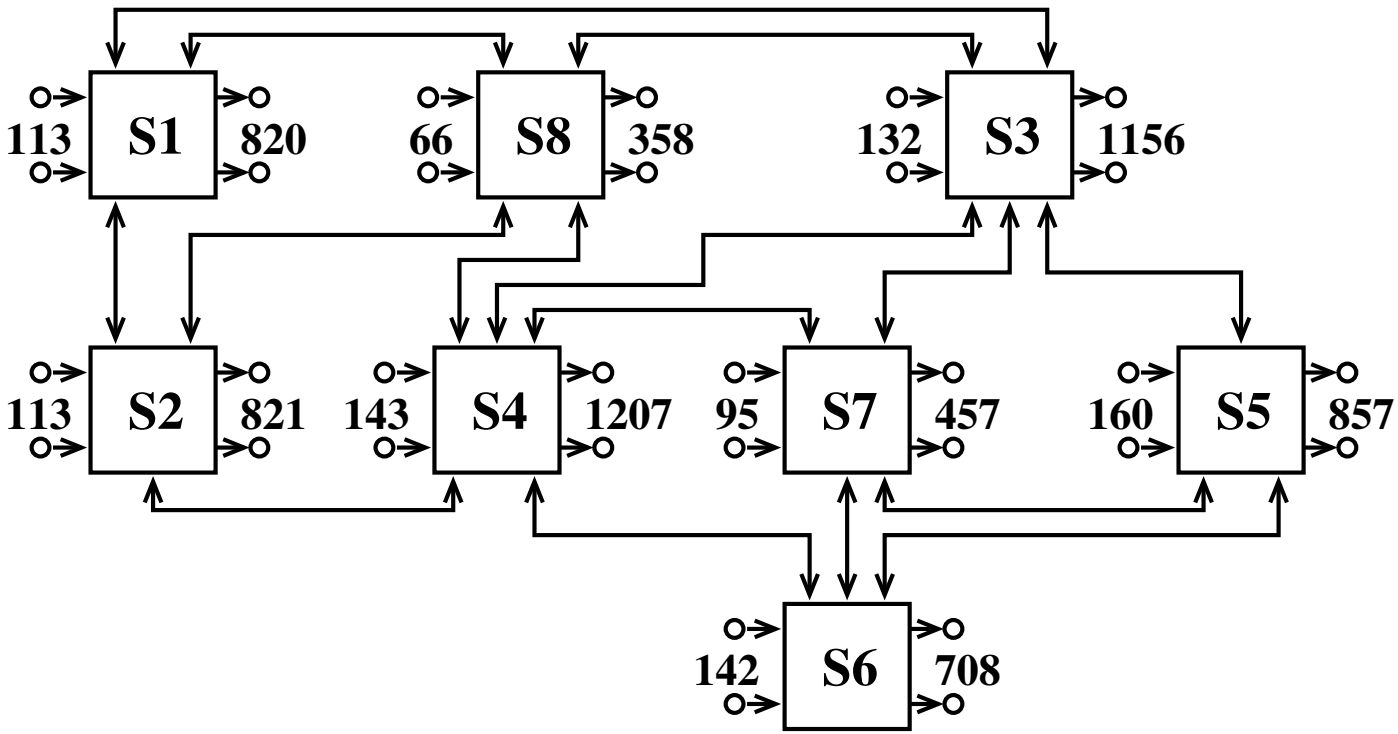


The AFDX receive protocol stack



An industrial AFDX configuration

- Two redundant networks including each 8 switches
- Roughly 120 end systems
- Roughly 1000 VL and 6400 paths



An industrial AFDX configuration

- VLs between switches

S \ D	1	2	3	4	5	6	7	8
1		71	78					34
2	72			77				34
3	90			212	35		42	52
4		97	134			37	35	48
5			80			72	64	
6				82	61		52	
7			52	47	59	67		
8	51	45	43	52				

An industrial AFDX configuration

- BAG and maximum frame length

Bag (ms)	Number of VL
2	20
4	40
8	78
16	142
32	229
64	220
128	255

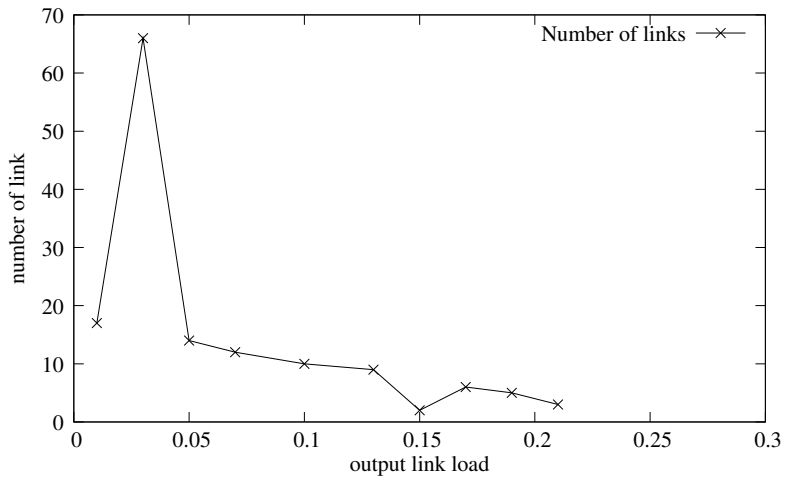
S_{max} (bytes)	Number of VL
0-150	561
151-300	202
301-600	114
601-900	57
901-1200	12
1201-1500	35
$>$ 1500	3

An industrial AFDX configuration

- Length of VL paths

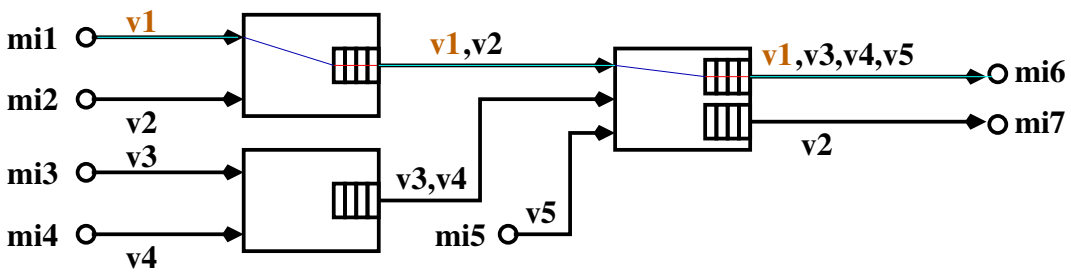
Number of crossed switches	Number of paths
1	1797
2	2787
3	1537
4	291

- Load of the links



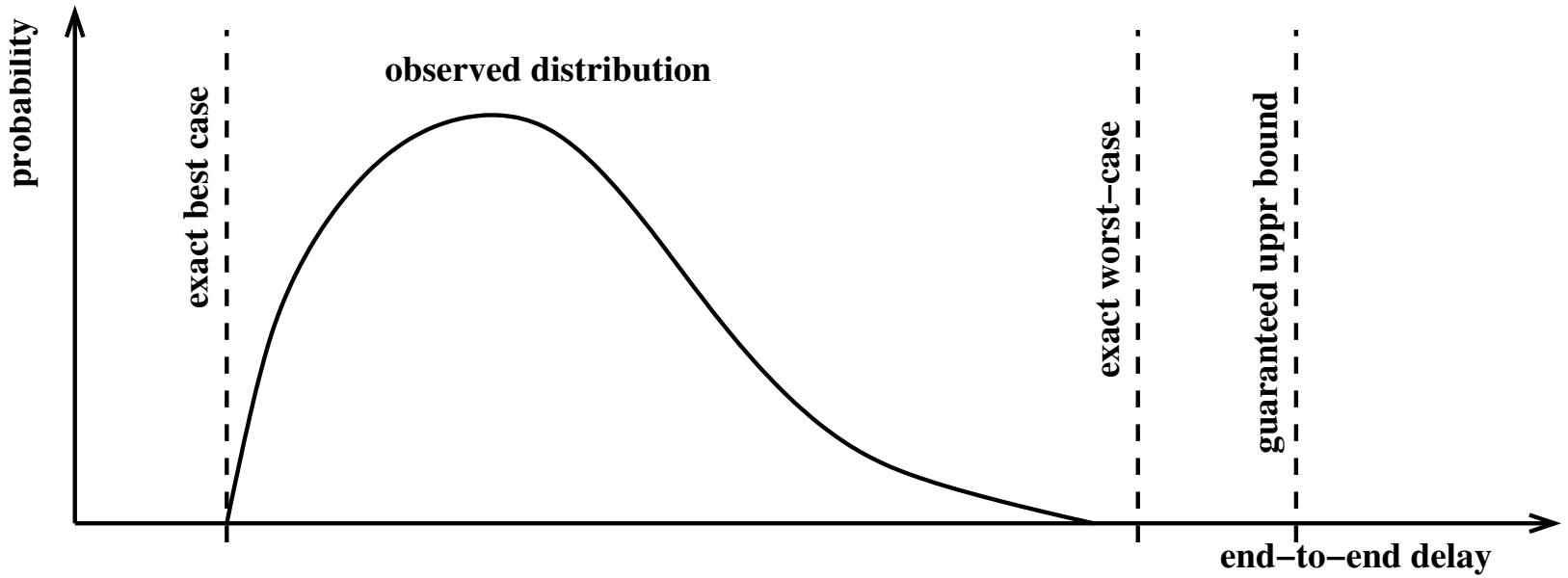
End-to-end delay on an AFDX network

- Three parts in the delay
  - Transmission times on links (known): link bandwidth, frame size
  - Switching latency (known)
  - Waiting time in FIFO queues (unknown)



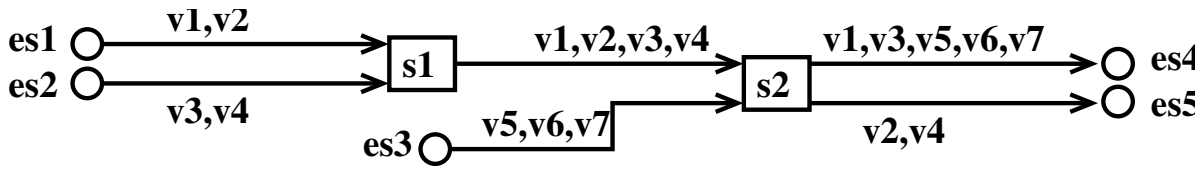
- Best-case delay: no waiting time in FIFO queues
- Worst-case delay: maximum overall waiting time in FIFO queues
- Distribution of the delay: distribution of the waiting time in FIFO queues

Characterisation of the end-to-end delay of a flow



- Best-case delay
  - Most of the time easy to obtain
- Worst-case delay
  - Most of the time difficult to obtain
  - Mandatory to guarantee that the deadlines are met
  - Certification for the A380: network calculus $\Rightarrow$ sure upper bound
- Distribution of the delay
  - Shows the real behaviour of the network
  - Shows where the delay is between the best case and the worst case

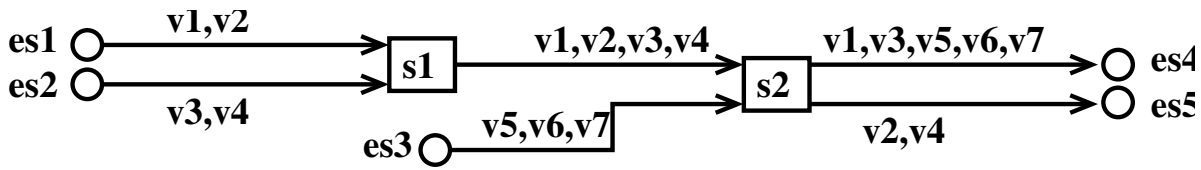
Worst-case end-to-end delay of a flow

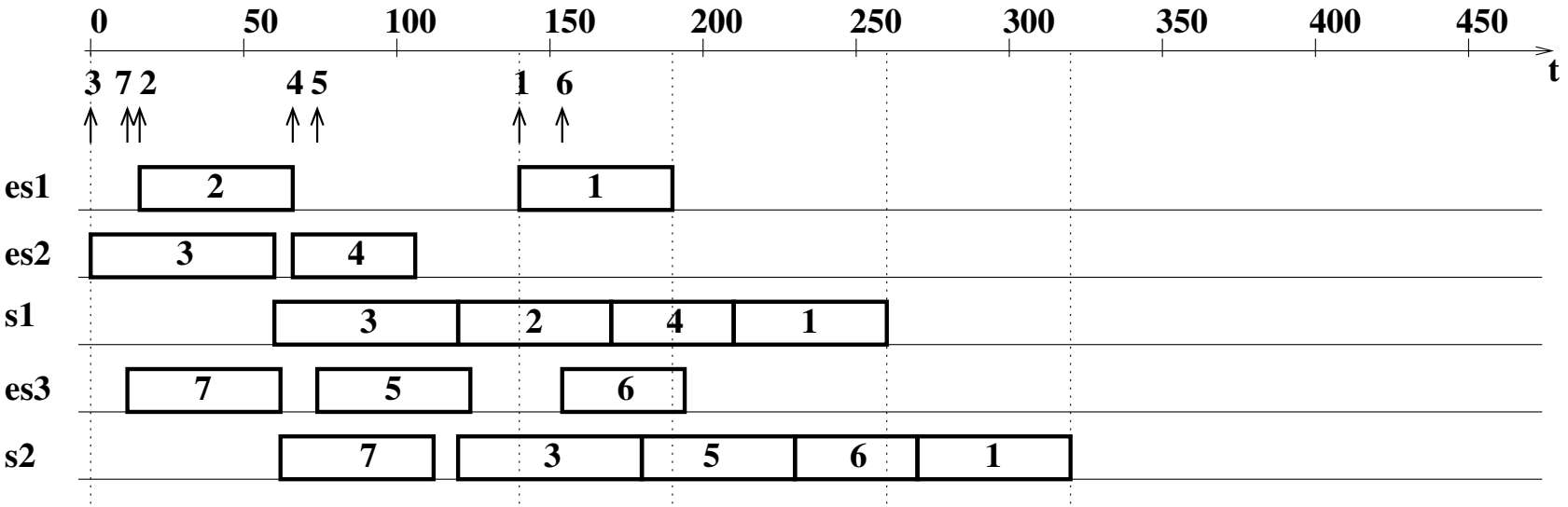


	$BAG(v_i)$	$s_{min}(v_i)$	$s_{max}(v_i)$	Tx
v_1	4 *ms*	625 bytes	625 bytes	50 μs
v_2	4 *ms*	625 bytes	625 bytes	50 μs
v_3	4 *ms*	750 bytes	750 bytes	60 μs
v_4	4 *ms*	500 bytes	500 bytes	40 μs
v_5	4 *ms*	625 bytes	625 bytes	50 μs
v_6	4 *ms*	500 bytes	500 bytes	40 μs
v_7	4 *ms*	625 bytes	625 bytes	50 μs

- Worst-case delay of $v_1 \Rightarrow$ identifie a worst-case scenario for v_1

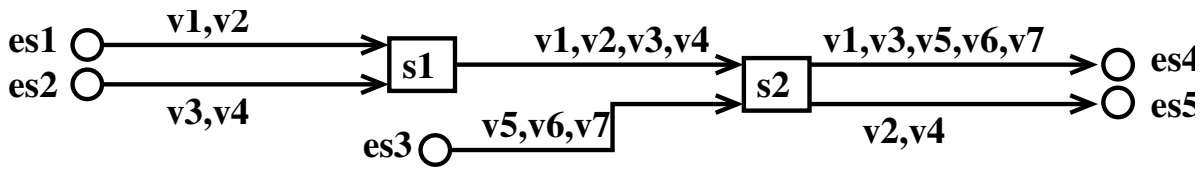
Arbitrary scenario for v_1

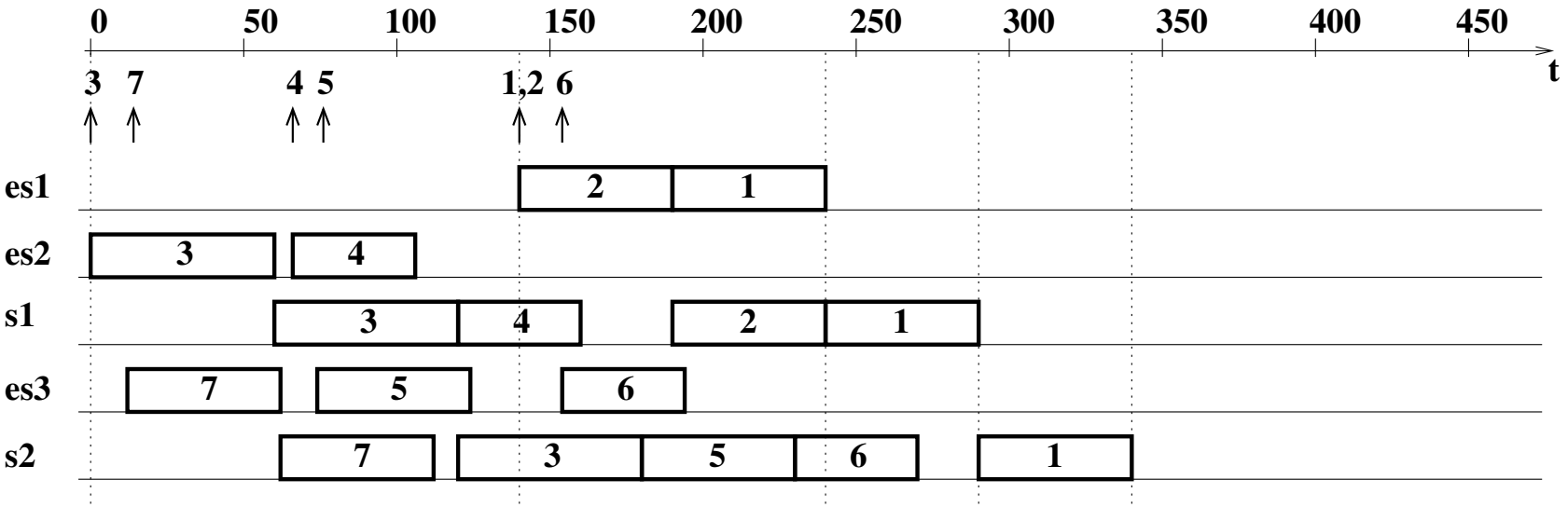




- End-to-end delay for v_1: 180 μs

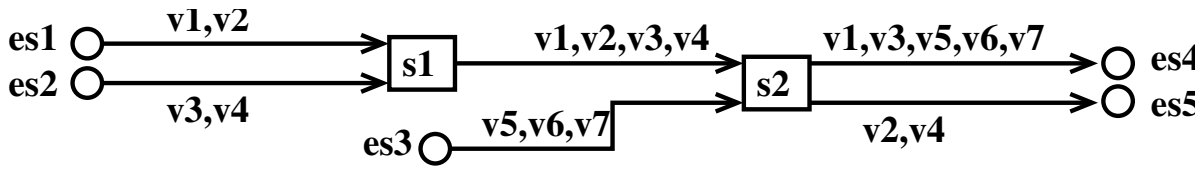
Maximizing the delay in source end system

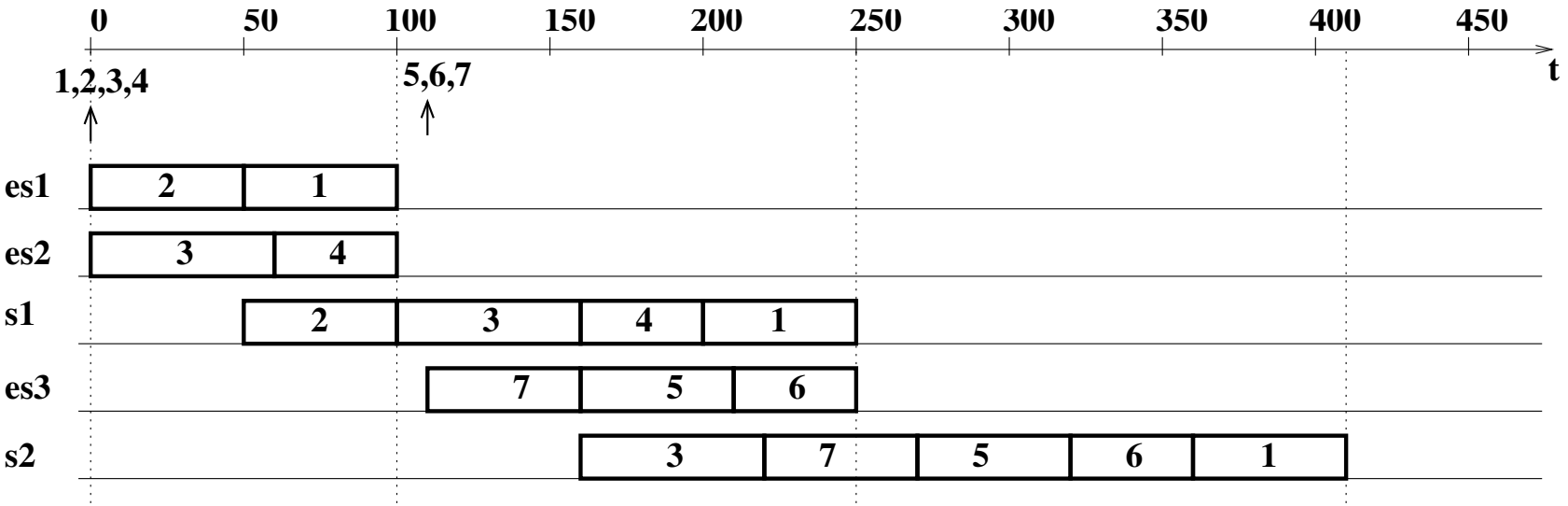




- End-to-end delay for v_1: 200 μs

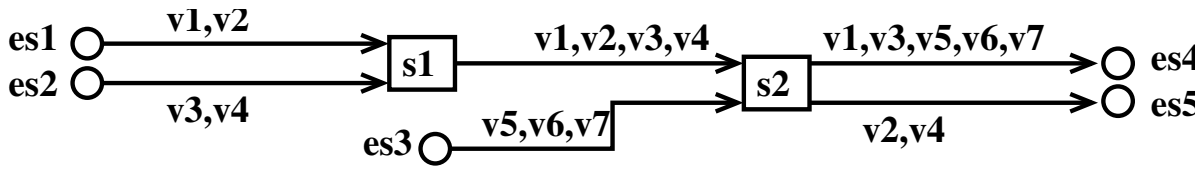
Critical instant in switch output ports: one candidate

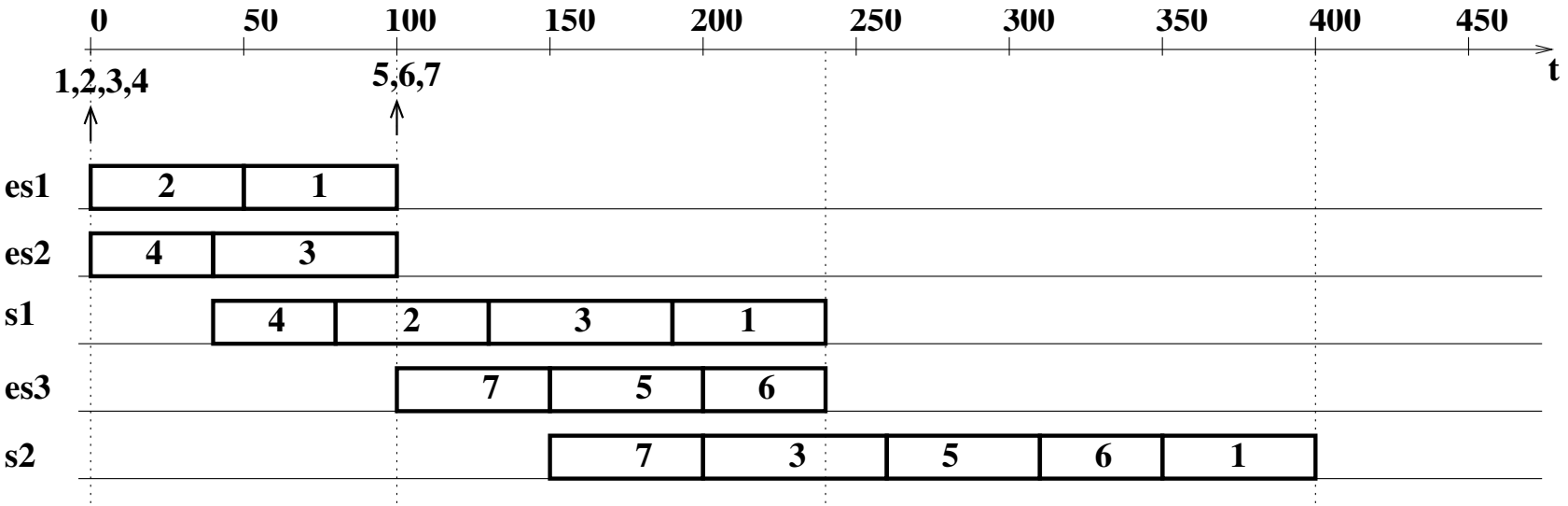




- End-to-end delay for v_1: 410 μs

Critical instant in switch output ports: another candidate



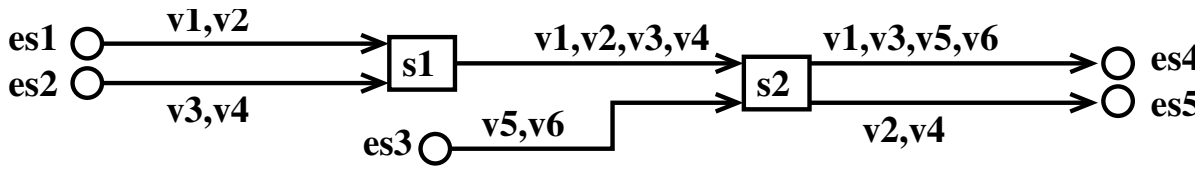


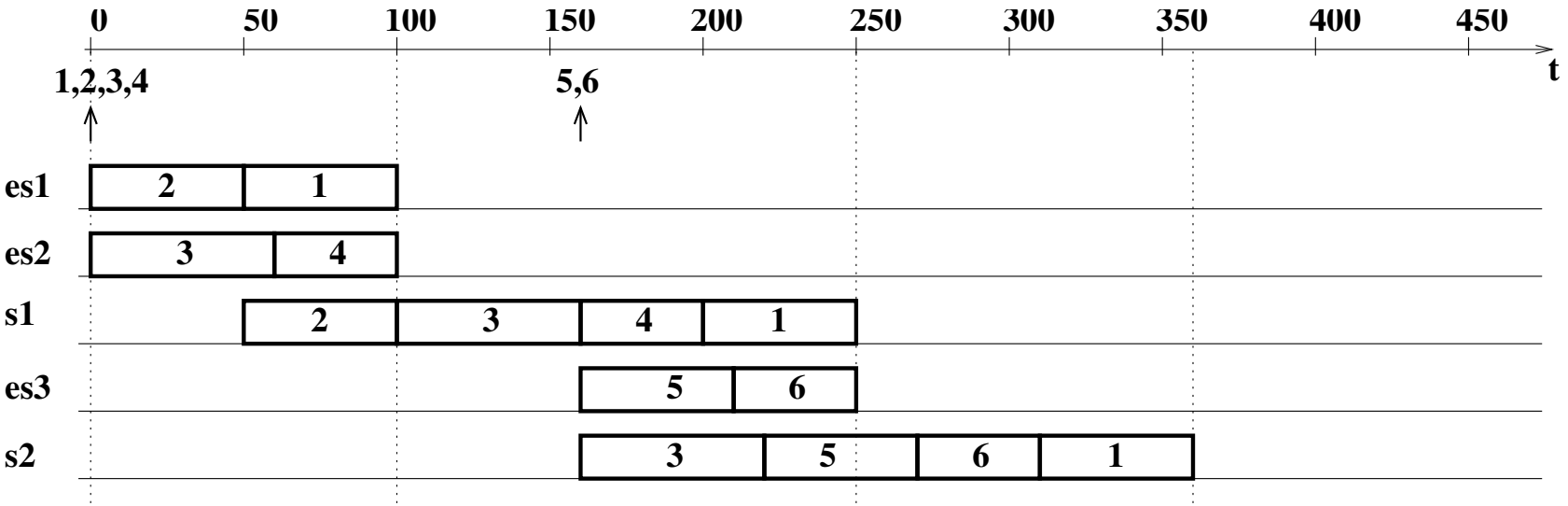
- End-to-end delay for v_1: 400 μs

Are candidate scenarios comparable

- Difference between the candidate scenarios: transmission order of the frames
- Impact of a frame f' on the worst-case delay
  - Transmission time of f' (depends on the number of bytes of the frame): a higher transmission time can lead to a higher impact on one output port
  - Number of output ports shared with the frame under study
- Worst-case scenario for the presented example
  - frames are sorted, first by decreasing size, then by increasing number of shared output ports
- In some cases it does not lead to the worst-case scenario (next slide configuration)

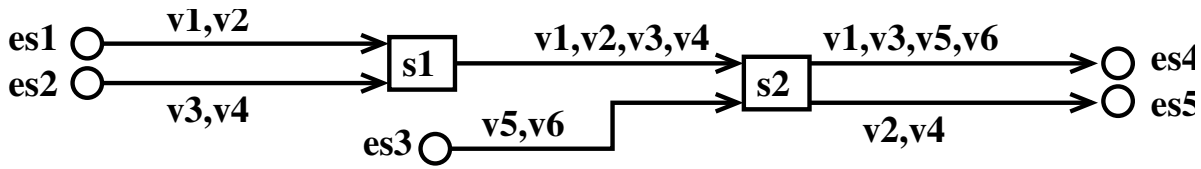
Critical instant in switch output ports: one candidate

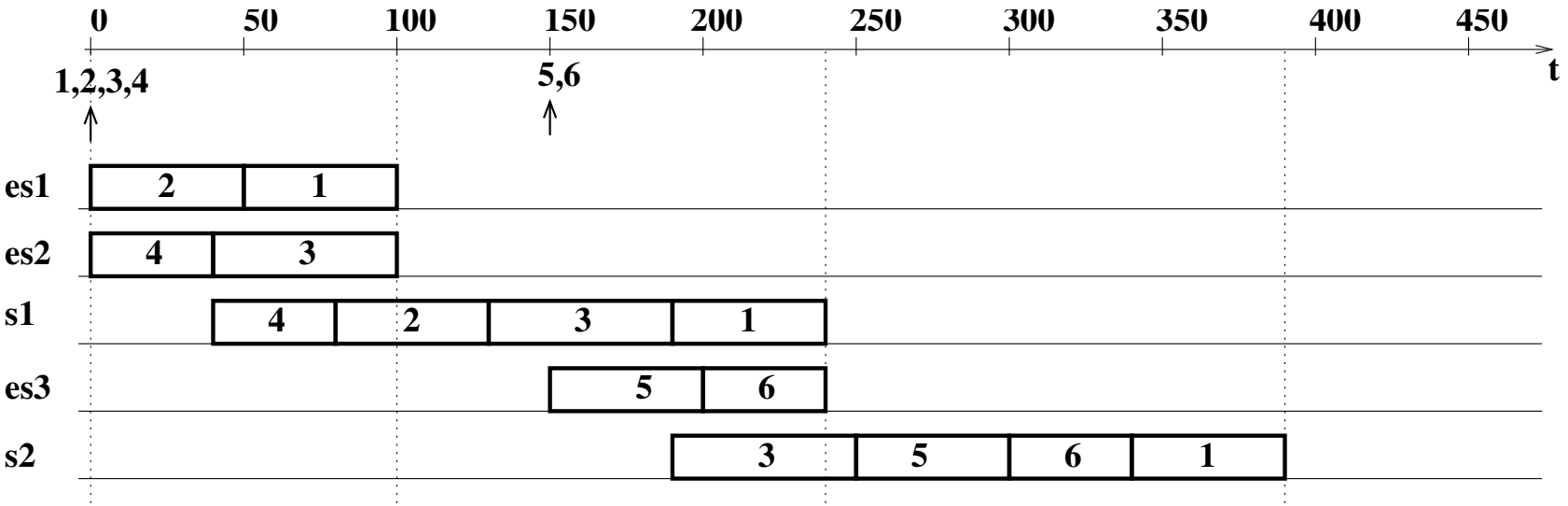




- End-to-end delay for v_1: 360 μs

Critical instant in switch output ports: another candidate

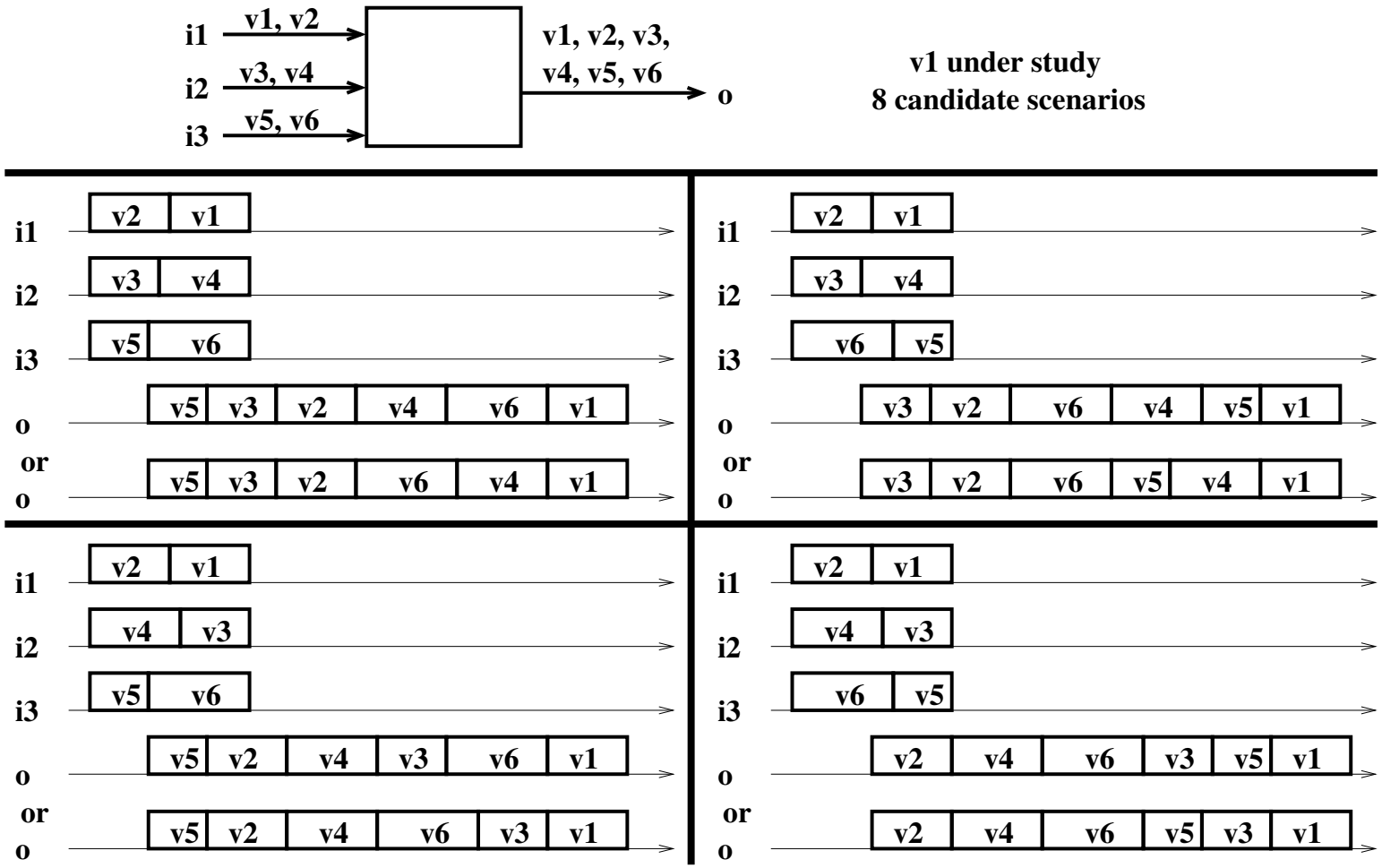




- End-to-end delay for v_1: 390 μs

Exhaustive search for the worst-case delay

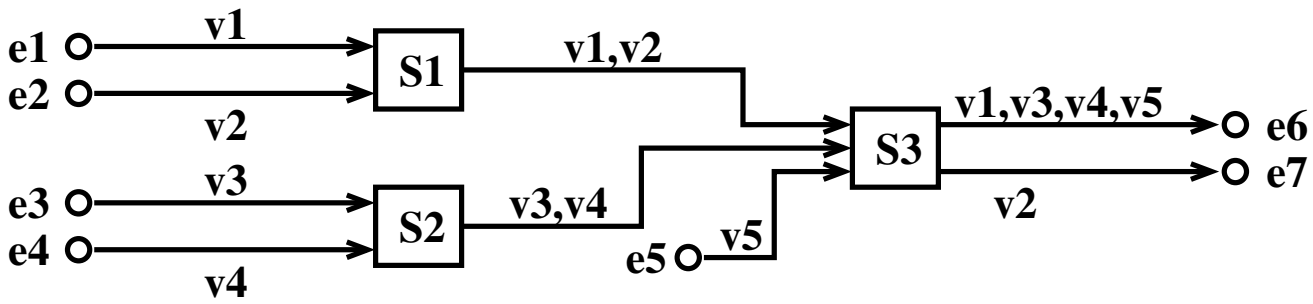
- Build all possible sequences and simulate



- Two implementations: ad hoc tool in Java, Uppaal
- Remaining combinatorial explosion issue

Exercise

- Determine the worst-case delay for the five flows of the following configuration



- All the VLs have the same BAG (4000 μs) and S_{max} (4000 bits)

Sure upper bound of the end-to-end delay of a given flow

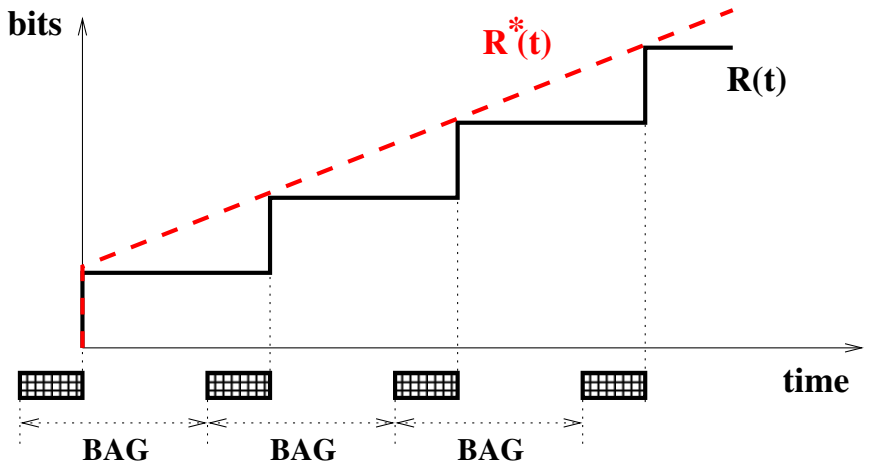
- Basic idea: upper bound the impact of each competing flow
- Two approaches have been proposed
  - Network calculus
  - Trajectories
- Industrial tool for the certification of the AFDX: Confgen developped by Rockwell-Collins

Network calculus: Modeling of the input traffic

- Definition of a traffic envelope R^* of the traffic function R

$$\forall s \leq t,\ R(t) - R(s) \leq R^*(t-s)$$

- A traffic envelope can be defined for each VL

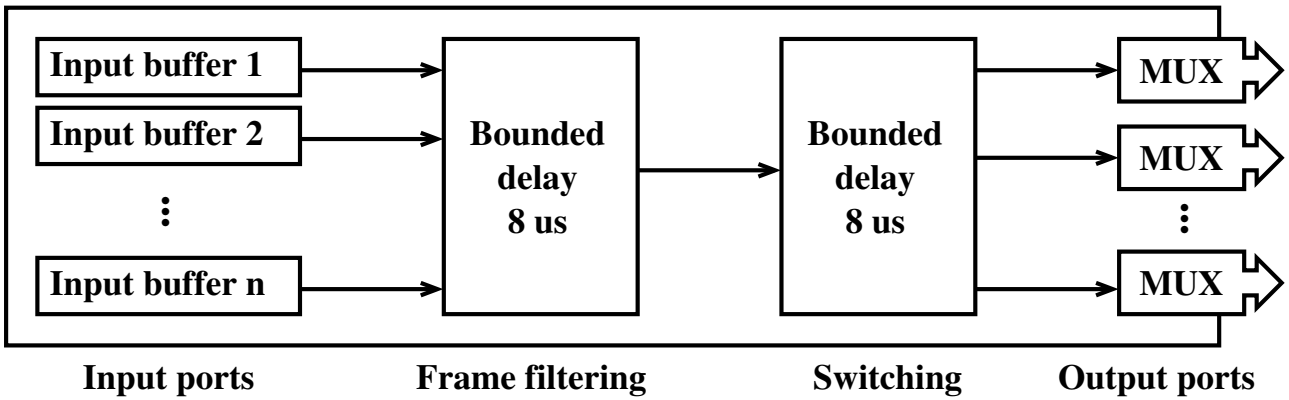


- Traffic envelope of a VL

$$R^*(t) = \gamma_{\frac{S_{max}}{BAG}, S_{max}} = S_{max} + \frac{S_{max}}{BAG} \times t$$

Network calculus: modeling of the architectural elements

- Use of standard elementary building blocks
  - Multiplexers
  - Demultiplexers
  - Buffers
- Modeling of an AFDX switch

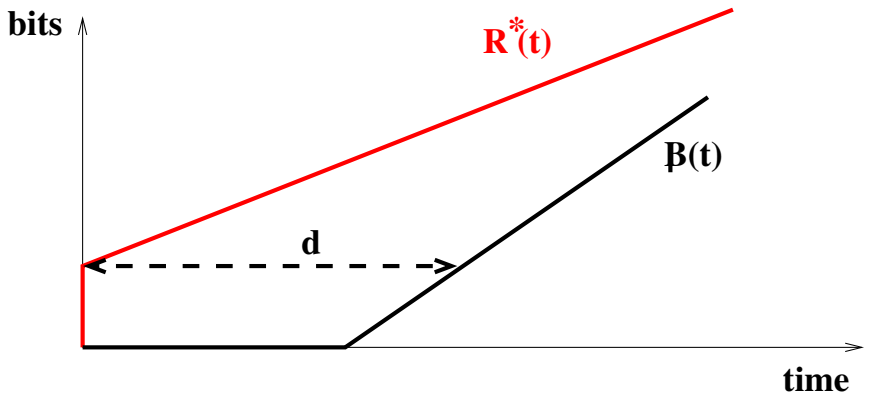


- Service curve $\beta_{R,T}$ of an output port: technological delay T and rate of the output link R

$$\beta_{100,16}(t) = max\,(0, 100 \times (t - 16))$$

Network calculus: end to end delay calculus

- Based on network calculus theorems
  - Upper bound on the delay d in one node

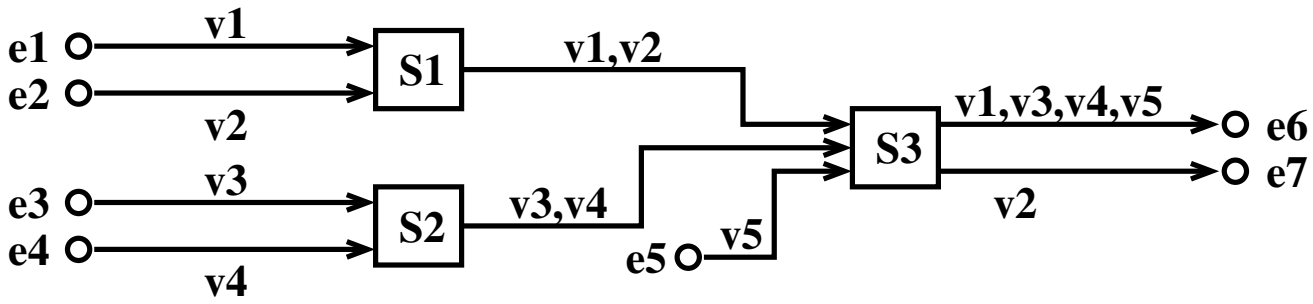


$$\forall t, d(t) \leq h(R^*, \beta)$$

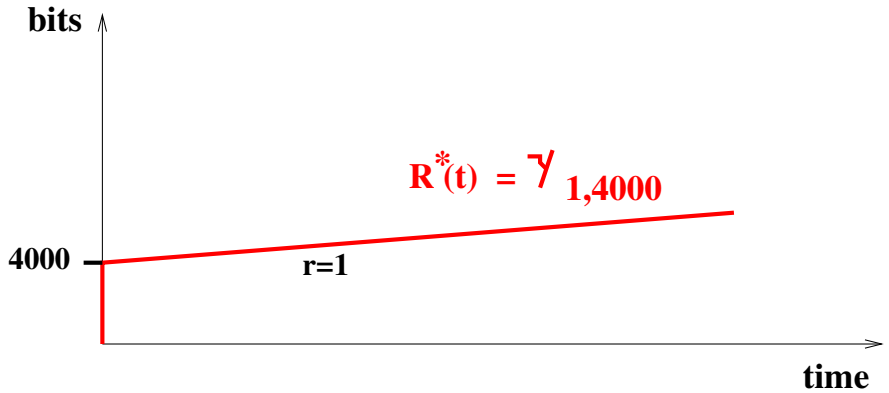
$$h(R^*, \beta) = \sup_{t \geq 0} \left(\inf \{ d \geq 0 \ : \ R^*(t) \leq \beta(t+d) \} \right)$$

- Overall arrival curve at an output port = sum of the arrival curves of the competing flows
- Output curve $\Rightarrow$ integrate the jitter (upper bound on the delay in the node)
- Data-flow delay calculus: the arrival curves of flows are propagated

Network calculus: illustration on a sample example



- All the VLs have the same BAG (4000 μs) and S_{max} (4000 bits)
- Arrival curve of each VL in its first switch

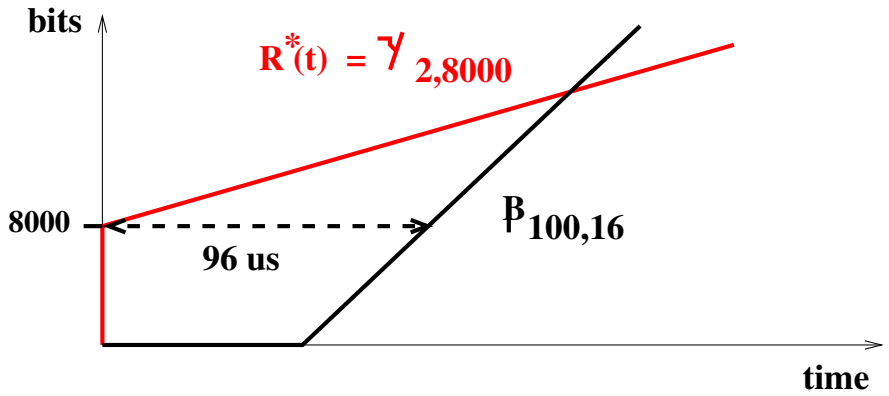


- Cumulative arrival curve in switches $S1$ and $S2$

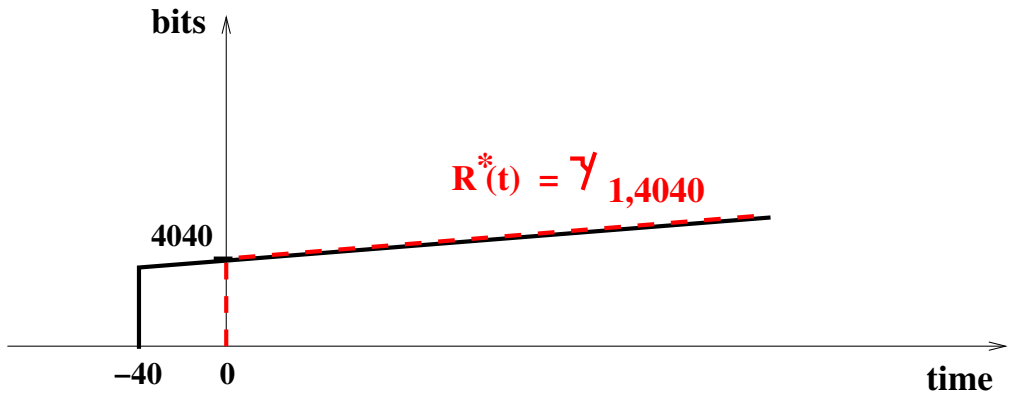
$$\gamma_{1,4000} + \gamma_{1,4000} = \gamma_{2,8000}$$

Network calculus: illustration on a sample example

- Upper bound on the delay of $v1$, $v2$, $v3$ and $v4$ in their first crossed switch: 96 μs
  - Technological latency: 16 μs
  - Maximum waiting time in the output buffer: 40 μs
  - Transmission time: 40 μs



- Output curve of $v1$, $v2$, $v3$ and $v4$ after their first crossed switch



Network calculus: illustration on a sample example

- Cumulative arrival curve in the upper output port of switch $S3$

$$\gamma_{1,4040} + \gamma_{1,4040} + \gamma_{1,4040} + \gamma_{1,4000} = \gamma_{4,16120}$$

- Upper bound on the delay of $v1$, $v3$, $v4$ and $v5$ in $S3$: 177.2 μs
- Cumulative arrival curve in the lower output port of switch $S3$

$$\gamma_{1,4040}$$

- Upper bound on the delay of $v2$ in $S3$: 56.4 μs
- Upper bound on the end-to-end delay of $v1$, $v3$ and $v4$

$$40 + 96 + 177.2 = 313.2\ \mu s$$

- Upper bound on the end-to-end delay of $v2$

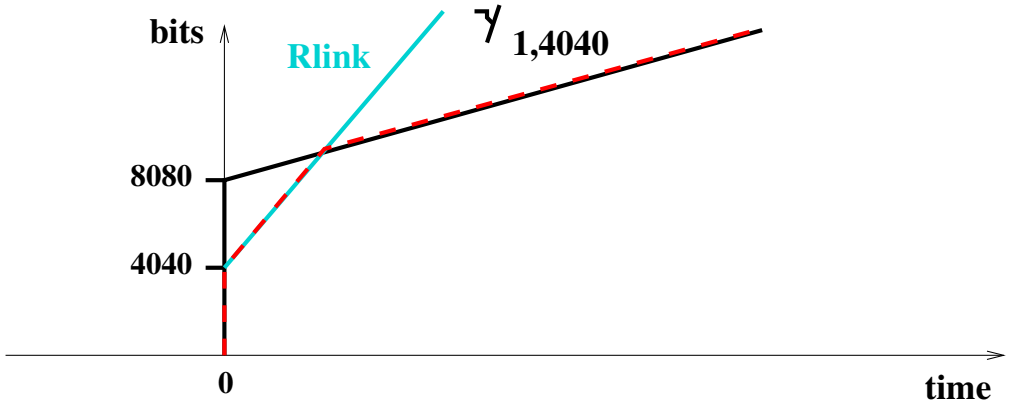
$$40 + 96 + 56.4 = 192.4\ \mu s$$

- Upper bound on the end-to-end delay of $v5$

$$40 + 177.2 = 217.2\ \mu s$$

Network calculus: flow serialization

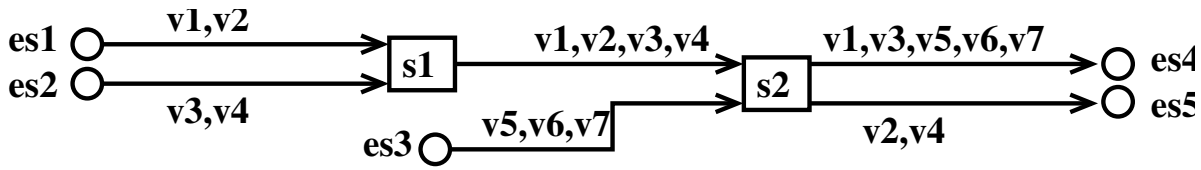
- Two frames cannot arrive simultaneously from the same link
  - Let's consider the upper output port of switch $S3$
  - The VLs $v3$ and $v4$ arrive from the same link
  - The maximum burst from this link is the maximum frame size among $v3$ and $v4$
  - The second frame is received at the rate of the link
- Optimized cumulative arrival curve from the link shared by $v3$ and $v4$
  - Reduction of the burst



  - The upper bounds on the end to end delays of $v1$, $v3$, $v4$ and $v5$ are reduced
    - ★ 273.6 μs for $v1$, $v3$ and $v4$
    - ★ 177.6 μs for $v5$

Exercise

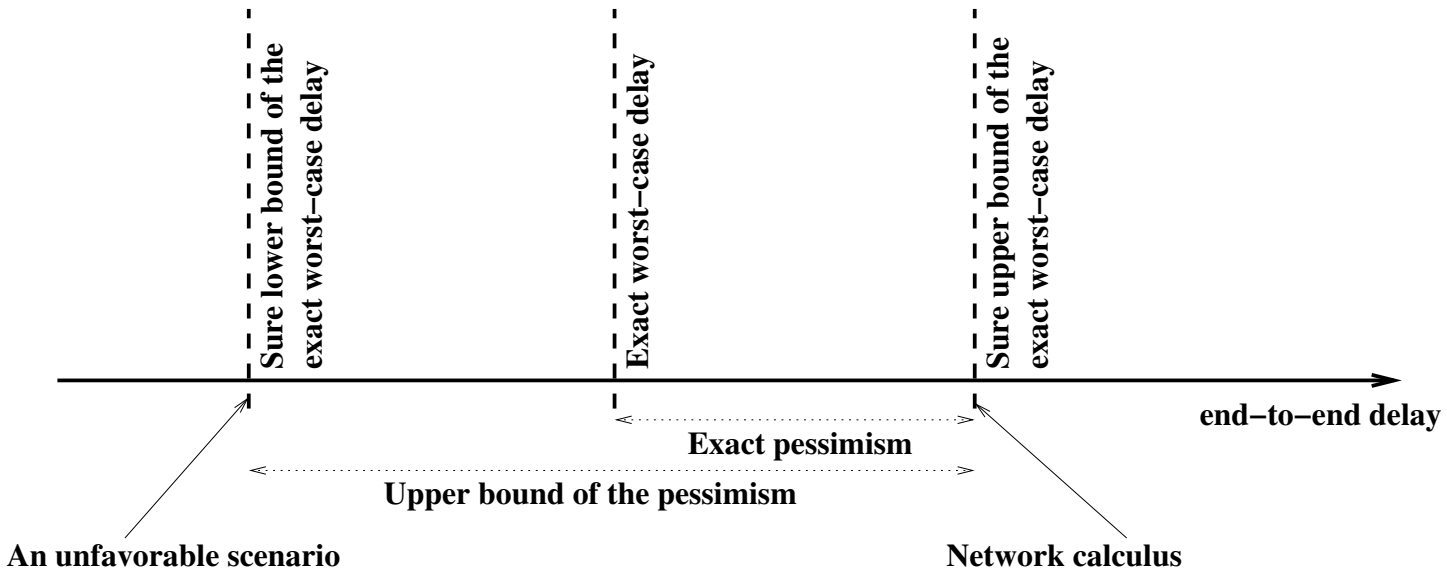
- Compute an upper bound on the delay of all the VLs of the following configuration



	$BAG(v_i)$	$s_{min}(v_i)$	$s_{max}(v_i)$	Tx
v_1	4 *ms*	625 bytes	625 bytes	50 μs
v_2	4 *ms*	625 bytes	625 bytes	50 μs
v_3	4 *ms*	750 bytes	750 bytes	60 μs
v_4	4 *ms*	500 bytes	500 bytes	40 μs
v_5	4 *ms*	625 bytes	625 bytes	50 μs
v_6	4 *ms*	500 bytes	500 bytes	40 μs
v_7	4 *ms*	625 bytes	625 bytes	50 μs

Pessimism of the obtained upper bounds

- Principle of the evaluation



- No assumptions on end system scheduling
- Upper bound on the pessimism of the network calculus approach on an industrial configuration
  - Between 0.8 % and 63 %
  - On average 13.5 %

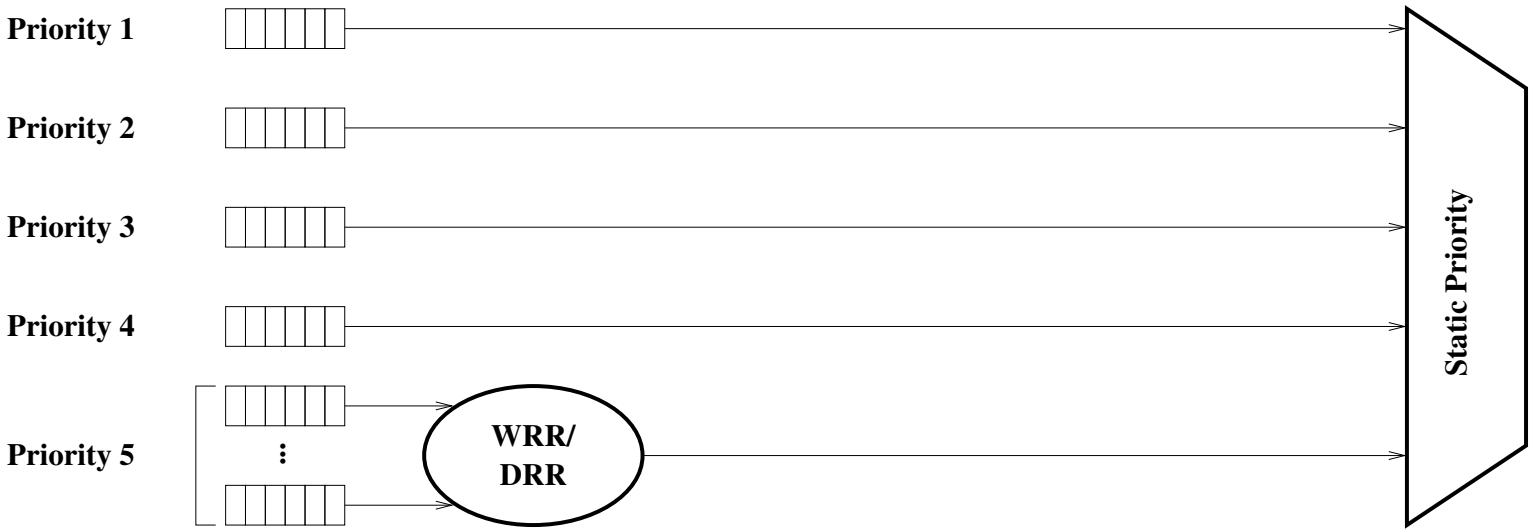
Theoretical traffic Vs effective traffic

- One VL: more or less a reservation of bandwidth made by a supplier
- Measures on a real configuration shows that
  - Some VLs are transmitted once during a whole flight
  - Some VLs are transmitted only in one mode (parking, take-off, landing, ...)
  - Actual period of a VL is on average 10 times larger than its BAG
- Worst-case analysis is based on theoretical traffic with no assumption on end system scheduling
  - Network and end systems are certified separately
  - A VL is a contract $\Rightarrow$ the VL is assumed to be used at its full capacity
  - Every function is assumed to be active in all the modes
- A very small part of the bandwidth is practically used

$\Rightarrow$ room for less/not critical flows, provided avionics flow constraints are guaranteed

Towards a QoS-AFDX

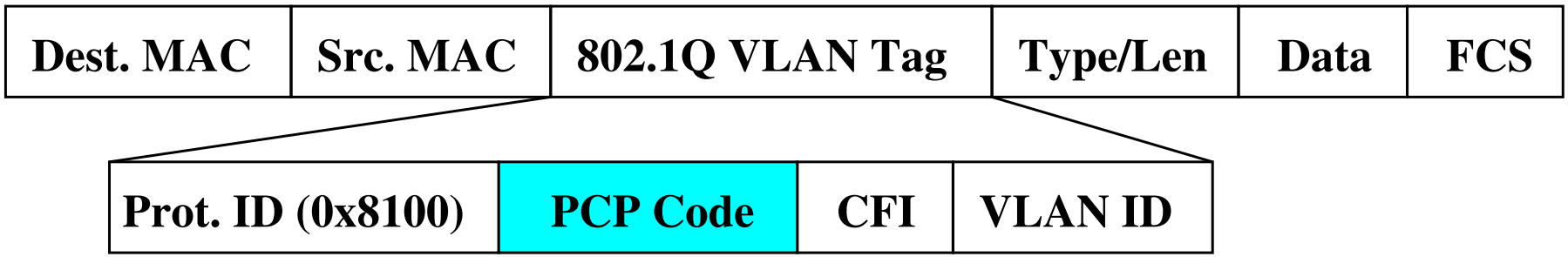
- Service discipline at switch output port shouldn't be too far from AFDX
- One envisioned solution



- Some other solutions are considered, e.g. the integration of a Busrst Limiting Shaper (BLS)
- Up to now, solutions with a global synchronization (e.g. AVB,TSN) are not considered

Strict priority algorithm

- No resource reservation
- Frames tagged with a 3 bit Priority Code Point value

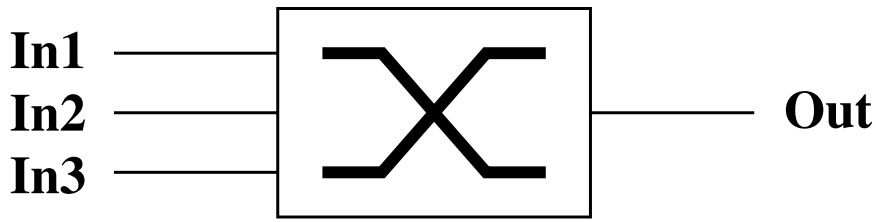


- A switch output port has a given number of queues (traffic classes)
- Mapping: PCP codes to traffc classes (queues), for example

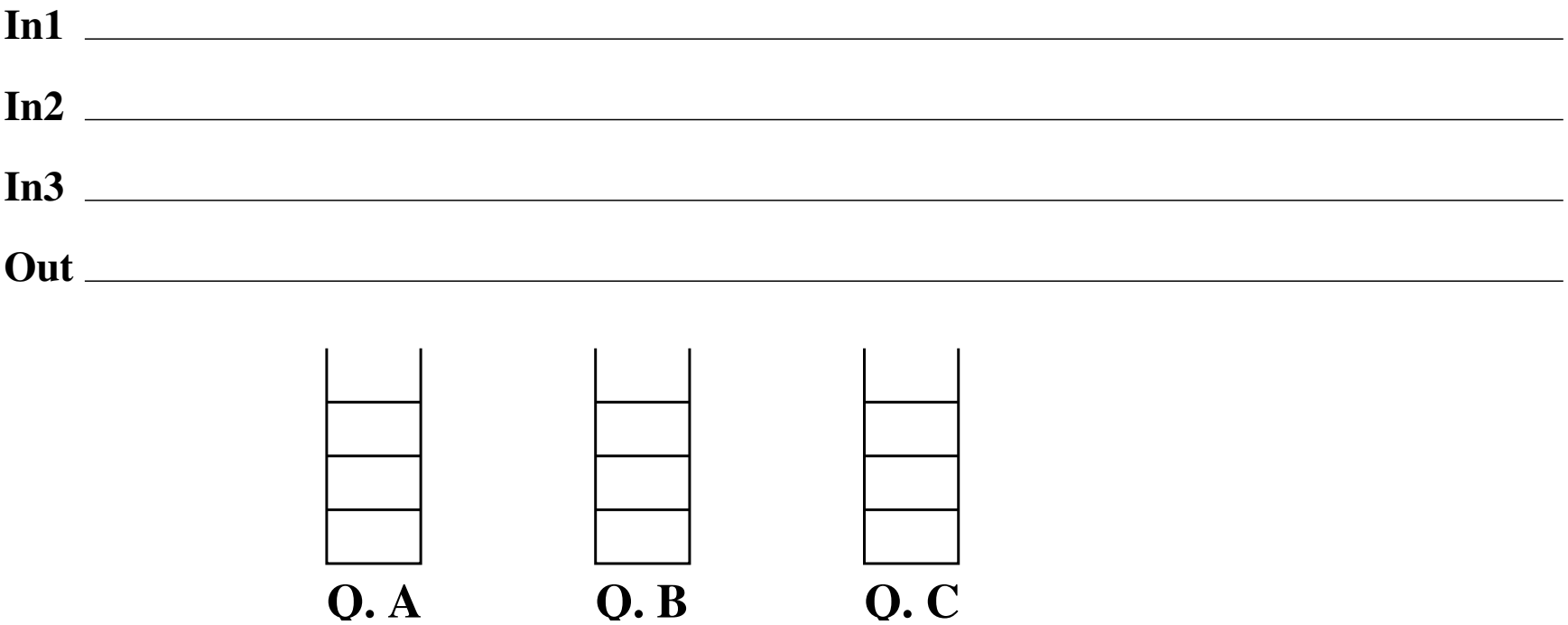
PCP code in frame	0	1	2	3	4	5	6	7
Traffic class number	0	0	0	0	1	1	2	2

Strict priority algorithm

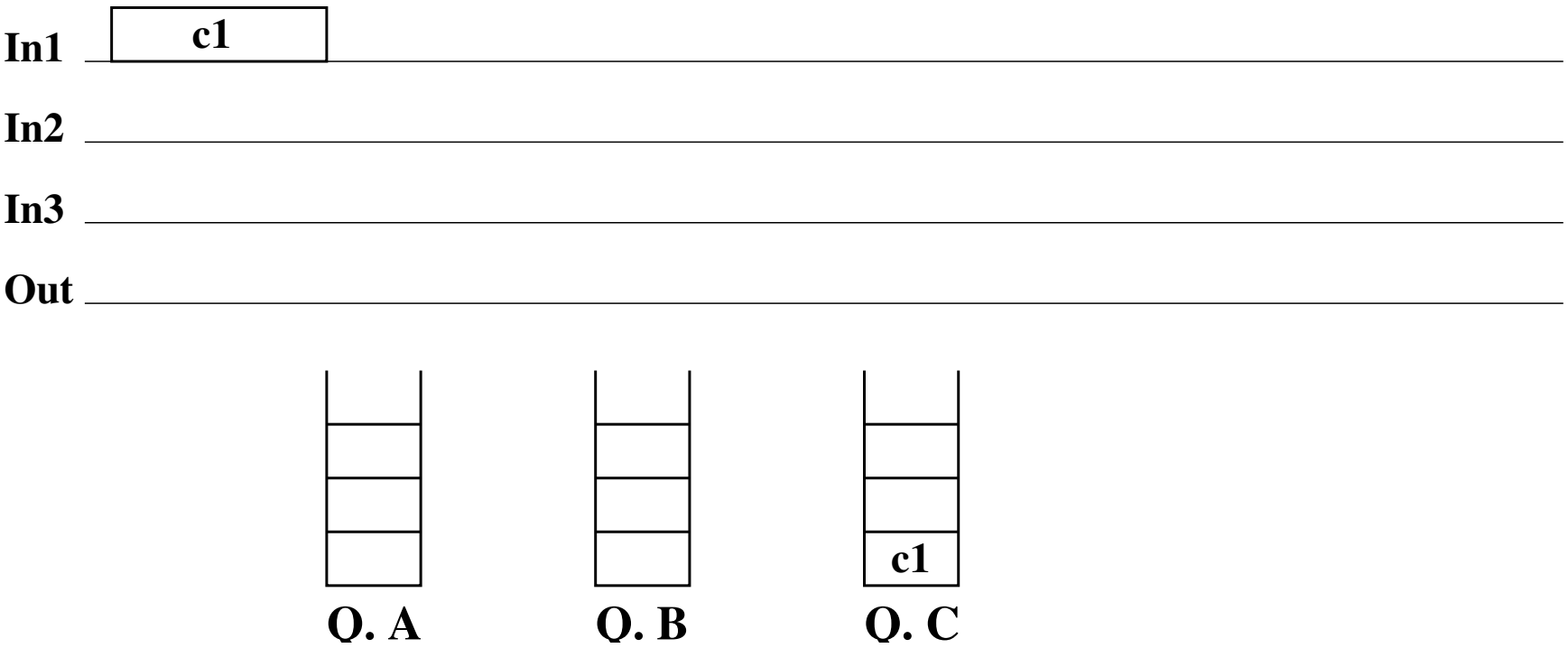
- Work conserving algorithm
  - A frame is transmitted as soon as at least one queue contains at least one frame
- Next frame for transmission
  - From queue with the highest traffic class number that has a frame available for transmission
- Might lead to starvation problems for low priority queues



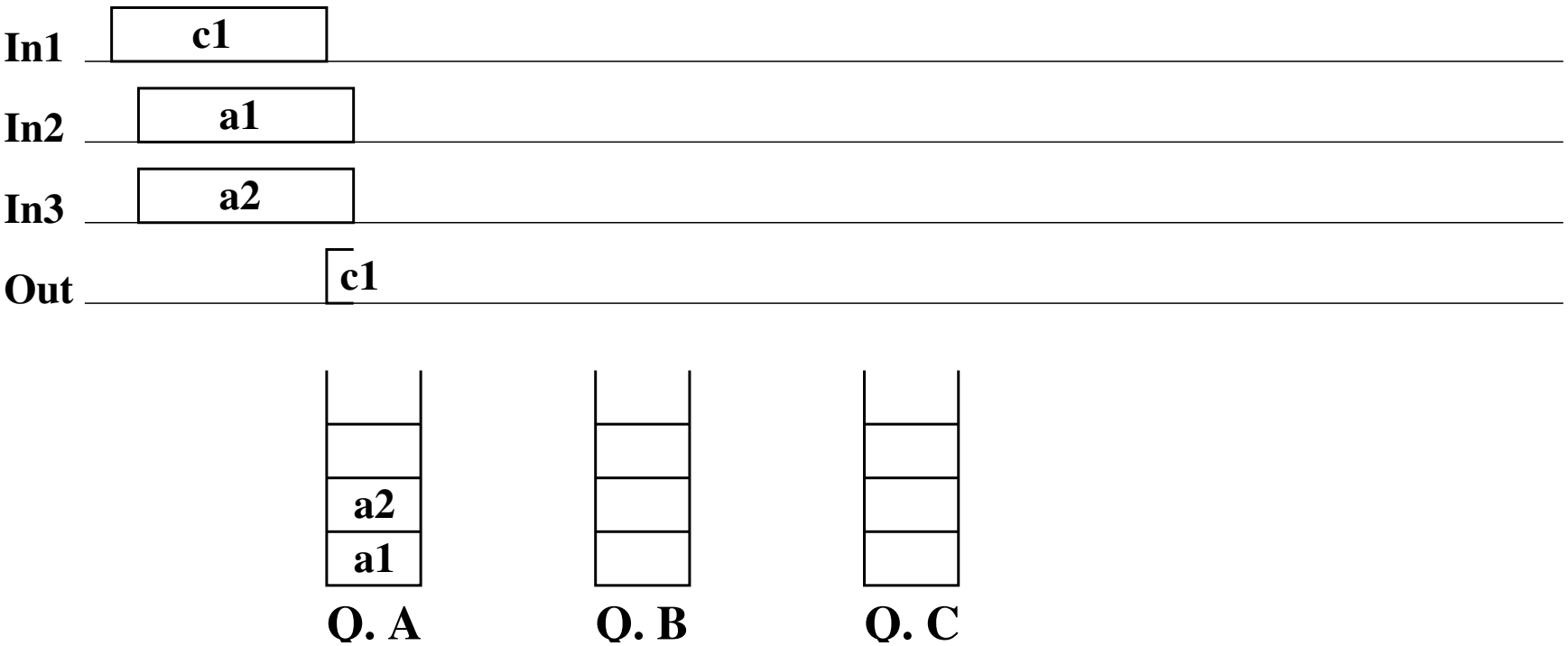
Strict priority algorithm



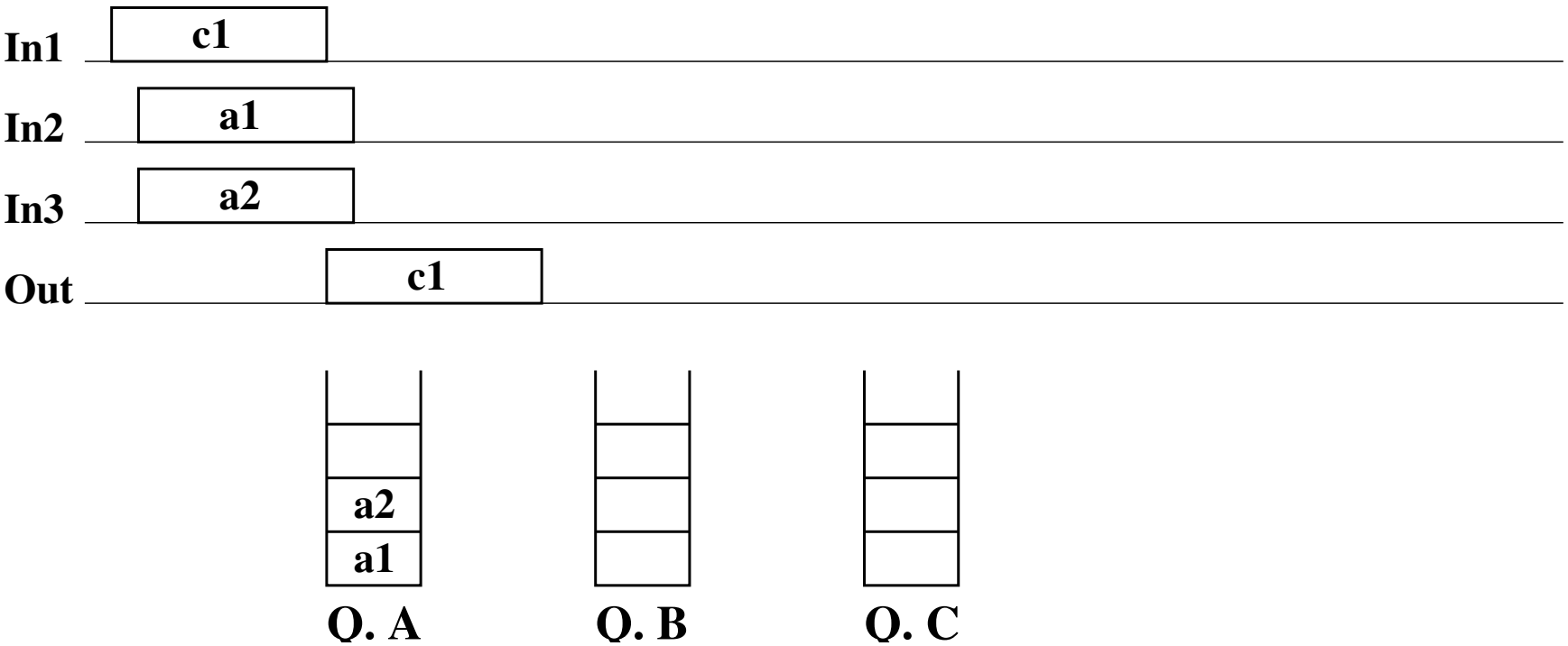
Strict priority algorithm



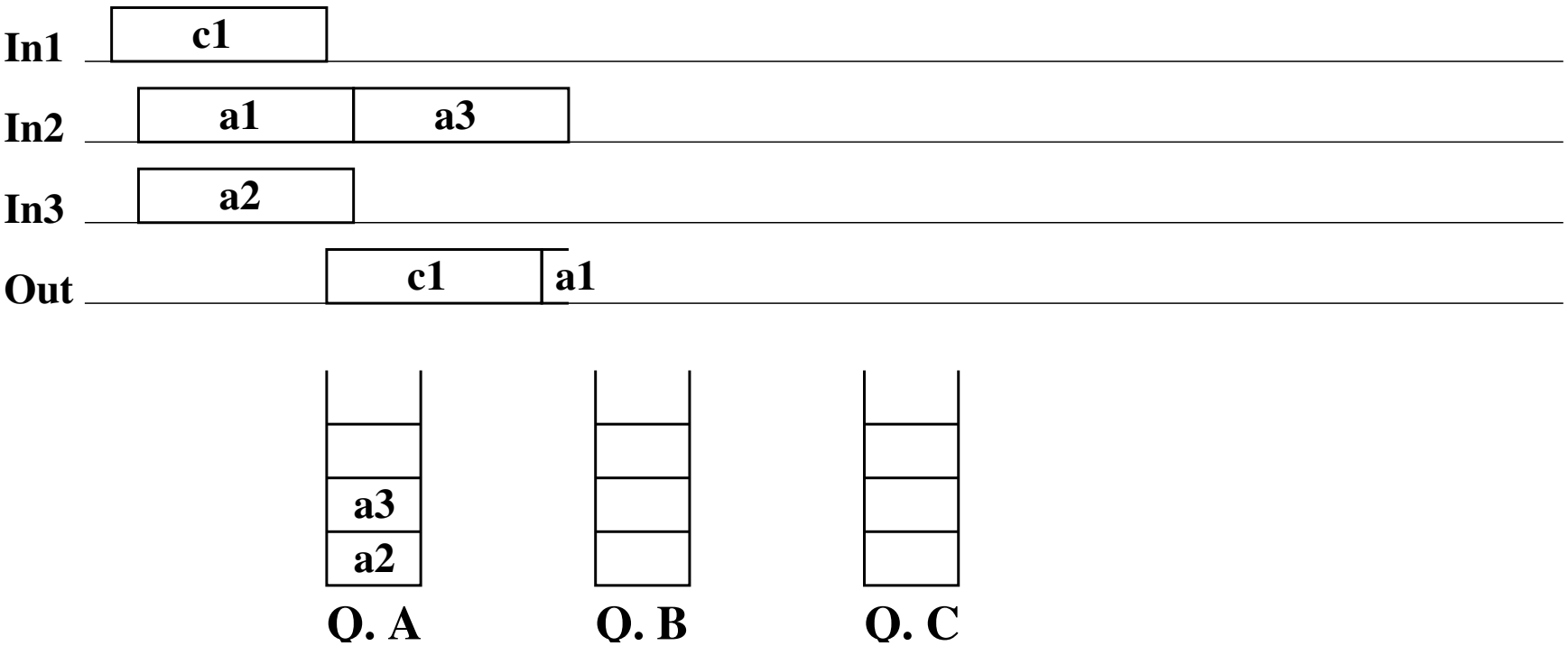
Strict priority algorithm



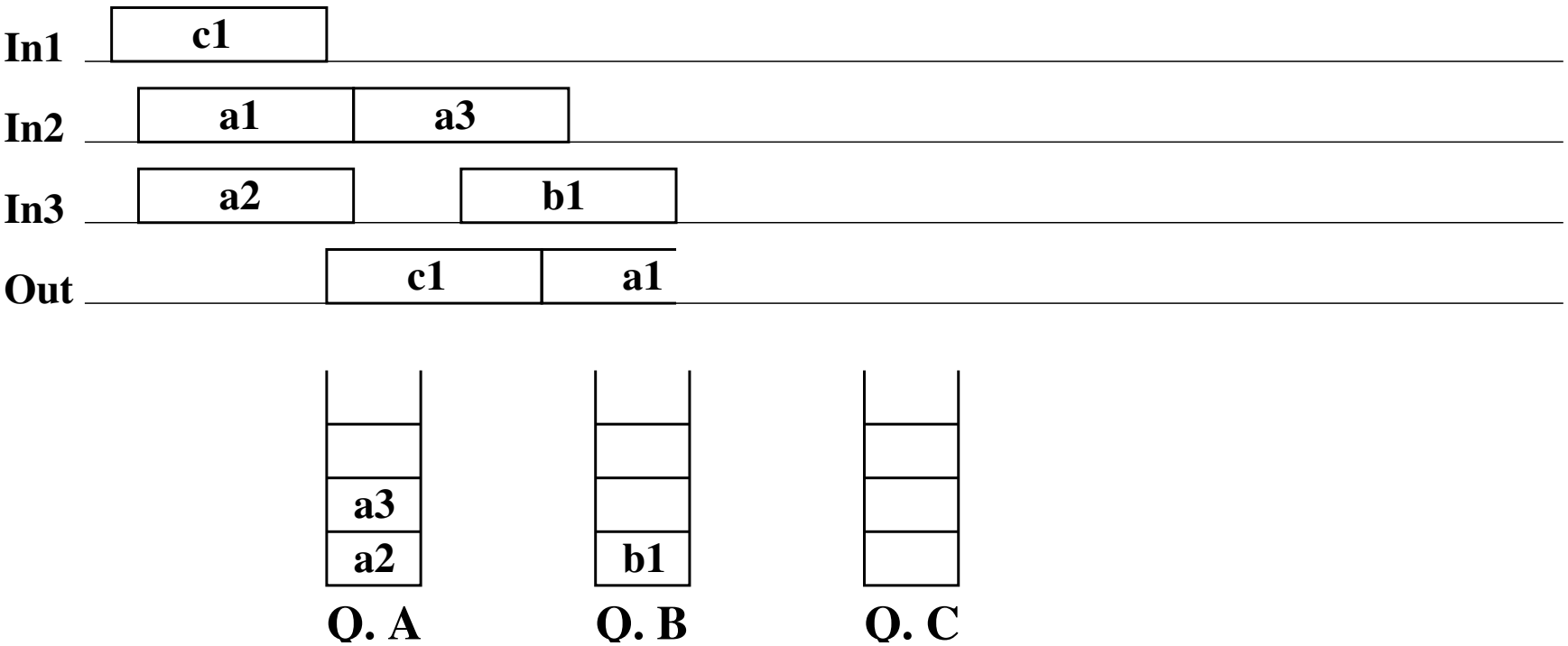
Strict priority algorithm



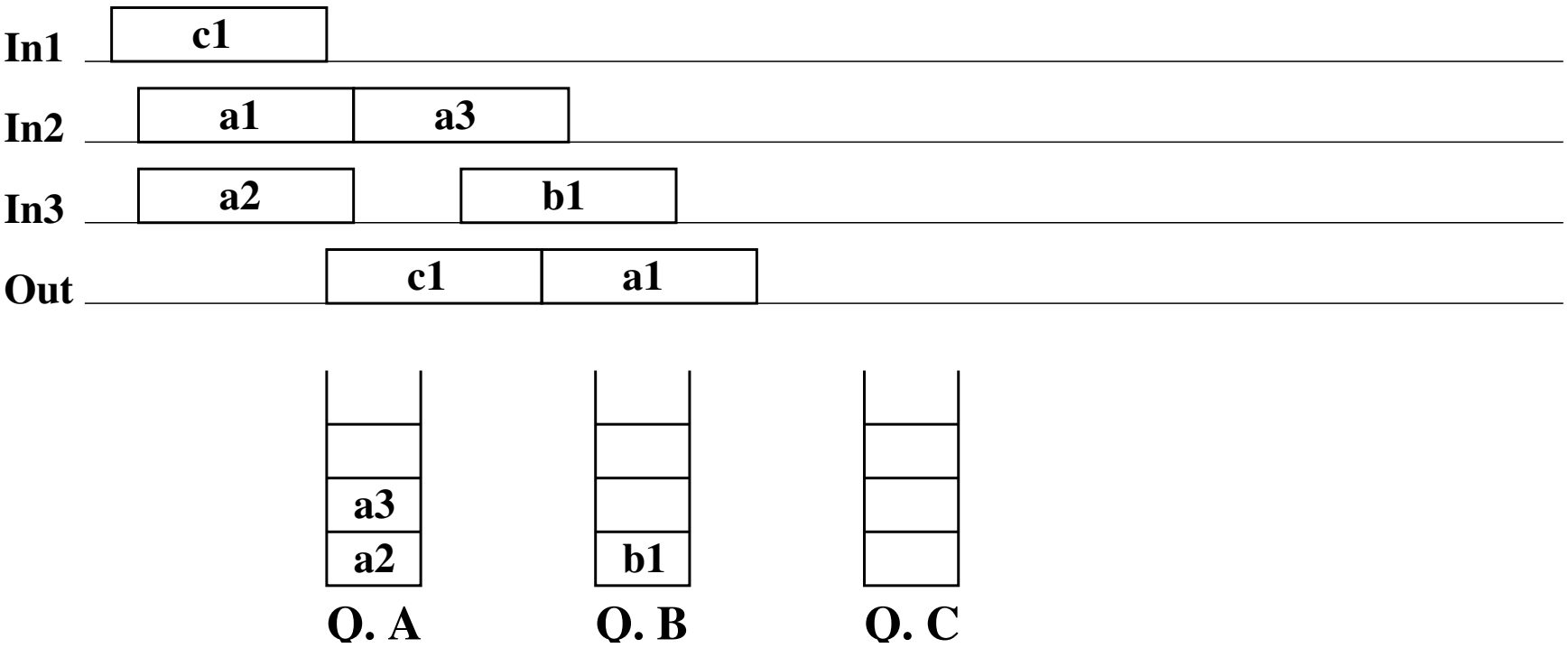
Strict priority algorithm



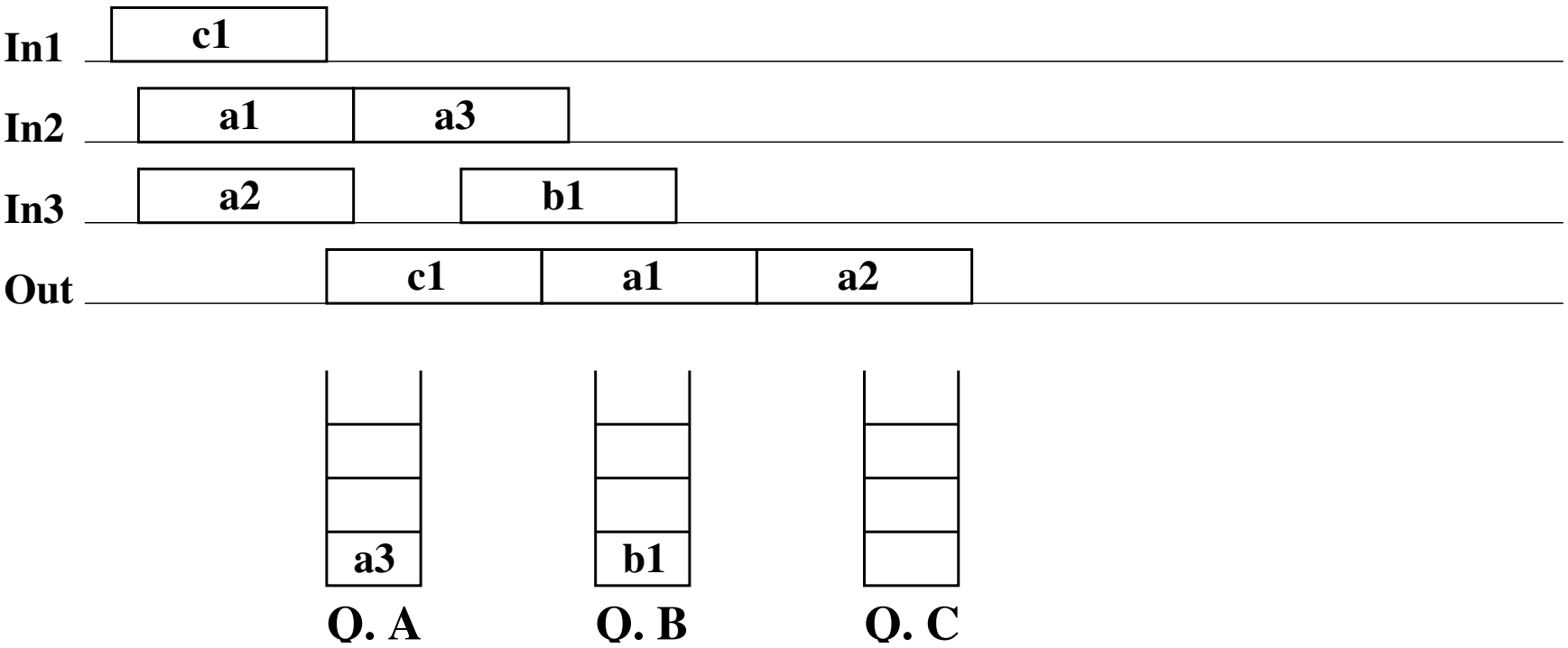
Strict priority algorithm



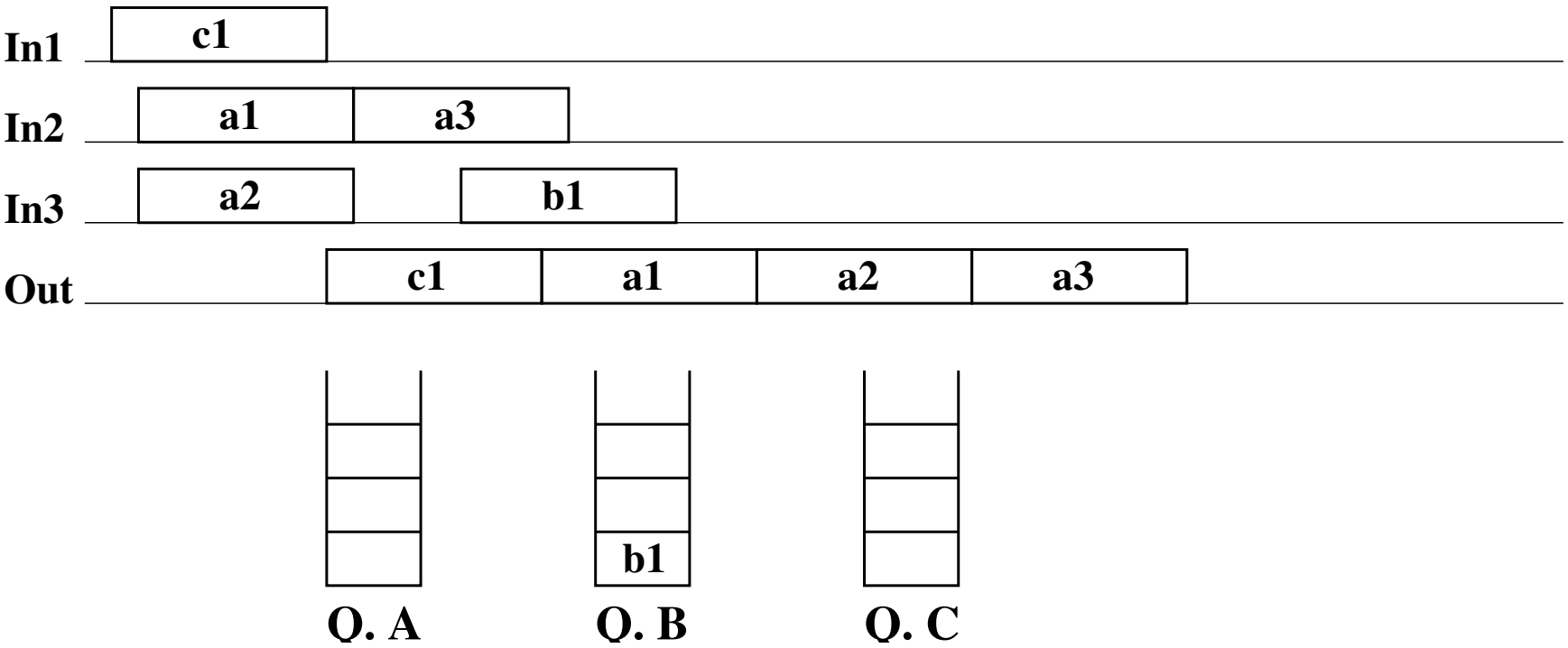
Strict priority algorithm



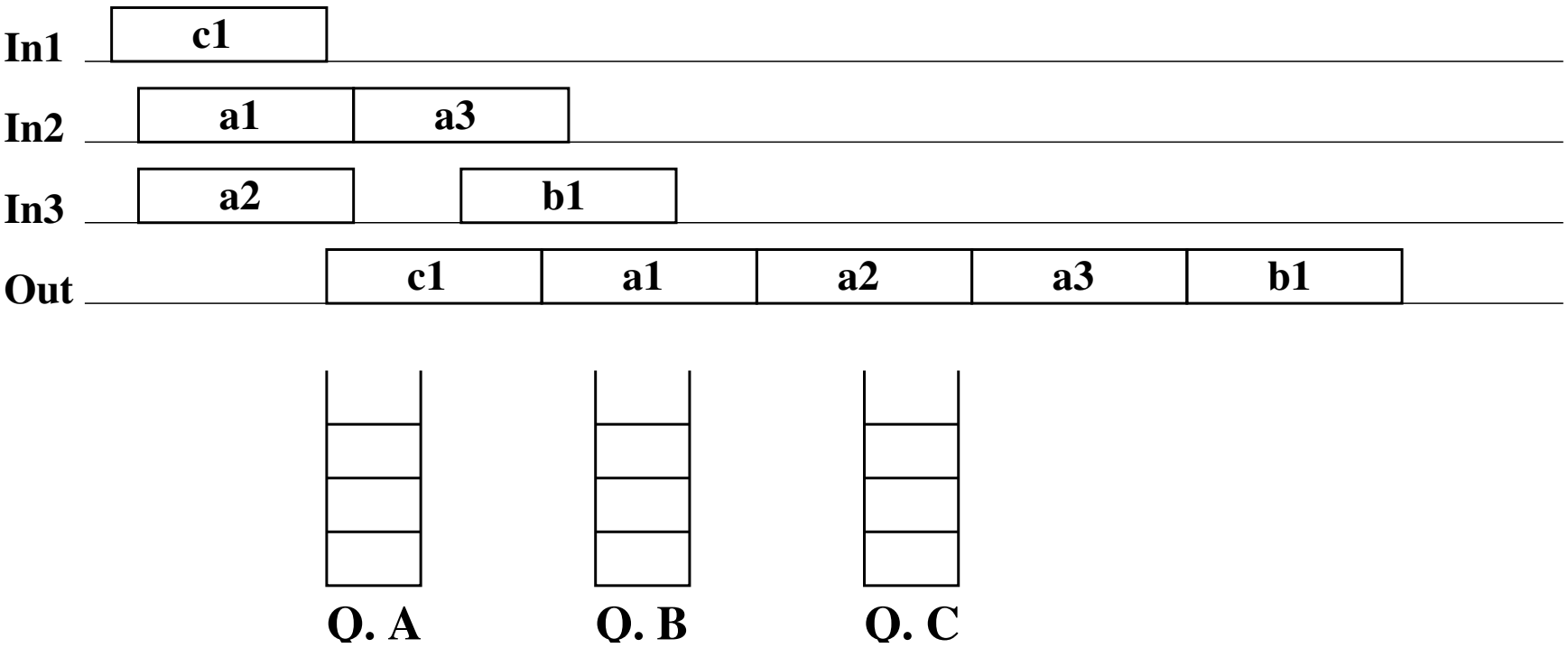
Strict priority algorithm



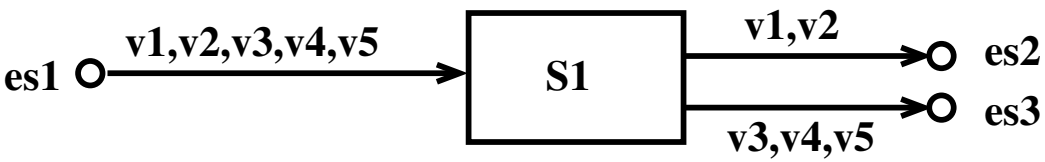
Strict priority algorithm



Strict priority algorithm



Some limitation of a strict priority algorithm



Transmission time of any frame on a link: 40 us

Switching latency: 0 us

	First frame	Period	Deadline
v1	0	120	120
v2	15	240	160
v3	5	240	200
v4	10	240	200
v5	20	240	280

1. Do all the flows respect their deadline with a FIFO scheduling in end systems and switches?
2. Same question with a strict priority policy
   - High priority: $v1$, $v2$
   - Low priority: $v3$, $v4$, $v5$
3. Same question with a round robin policy with 1 frame per class in each round
   - Class 1: $v1$, $v2$
   - Low priority: $v3$, $v4$, $v5$

Weighted Round Robin

- Each flow is allocated to a traffic class
- Each class is assigned a weight (number of frames)
- Classes are served, based on a round robin policy

```
current_buffer ← 0;
While True do begin
    cpt ← weight[current_buffer];
    While not empty(current_buffer) and cpt > 0 do begin
        send_frame(current_buffer);
        cpt ← cpt − 1;
    End;
End;
```

Deficit Round Robin

- Each class is assigned a budget (number of bytes)
- Remaining budget can be used in the following round

```
For i from 0 to N-1 do Deficit[i] ← 0;
Cur ← 0;
While True do begin
    Deficit[Cur] ← Deficit[Cur] + Weight[Cur];
    While not empty(Cur) and
            Deficit[Cur] ≥ size(head(Cur)) do begin
        Deficit[Cur] ← Deficit[Cur] - size(head(Cur));
        send_frame (Cur);
    End;
    If empty(Cur) then begin
        Deficit[Cur] ← 0;
    End;
    Cur ← (Cur+1) mod N;
End;
```

Deficit Round Robin Vs Weighted Round Robin

	Frame sizes (bytes)		
	Case 1	Case 2	Case 3
Class C1	1500	1500	1500
Class C2	1500	1500	1000,1500,1000,...
Class C3	1500	250	1500,250,500,1000,1500,...

1. Give the portion of bandwidth for each class with WRR and 1 frame for each class in each round. We assume that there are always pending frames for all the classes.
2. Same question with DRR and 1500 bytes for each class in each round.

Deficit Round Robin Vs Weighted Round Robin

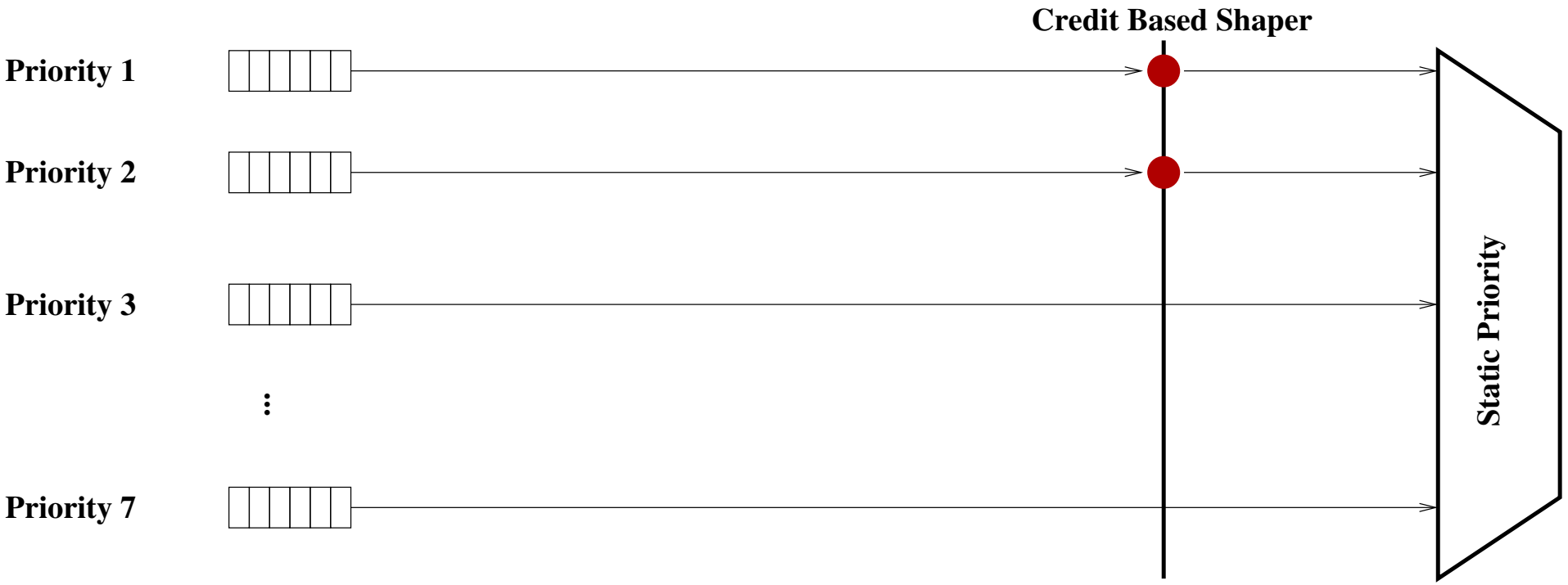
- Implementation
  - DRR is more complex
- Fairness
  - DRR acheives good fairness
  - WRR acheives good fairness when frames have homogeneous sizes
  - WRR acheives poor fairness when frame sizes highly vary within a class

Ethernet AVB

- Defined by the IEEE 802.1 Audio/Video Bridging Task Group
- Specifications for time-synchronized low-latency streaming services through IEEE 802 networks
- Includes three specifications
  - IEEE 802.1AS: timing and synchronization protocol for time-sensitive applications
  - IEEE 802.1 Qat: Stream Reservation Protocol
    - ★ Three-step signalling process (stream advertising, registration, de-registration)
    - ★ Resource reservation within switches (buffers, queues) along the path between sender and receiver
  - IEEE 802.1Qav: forwarding and queueing enhancements for time-sensitive streams
    - ★ Extension of the IEEE 802.1Q standard: credit based shaper transmission algorithm

Credit Based Shaper (CBS)

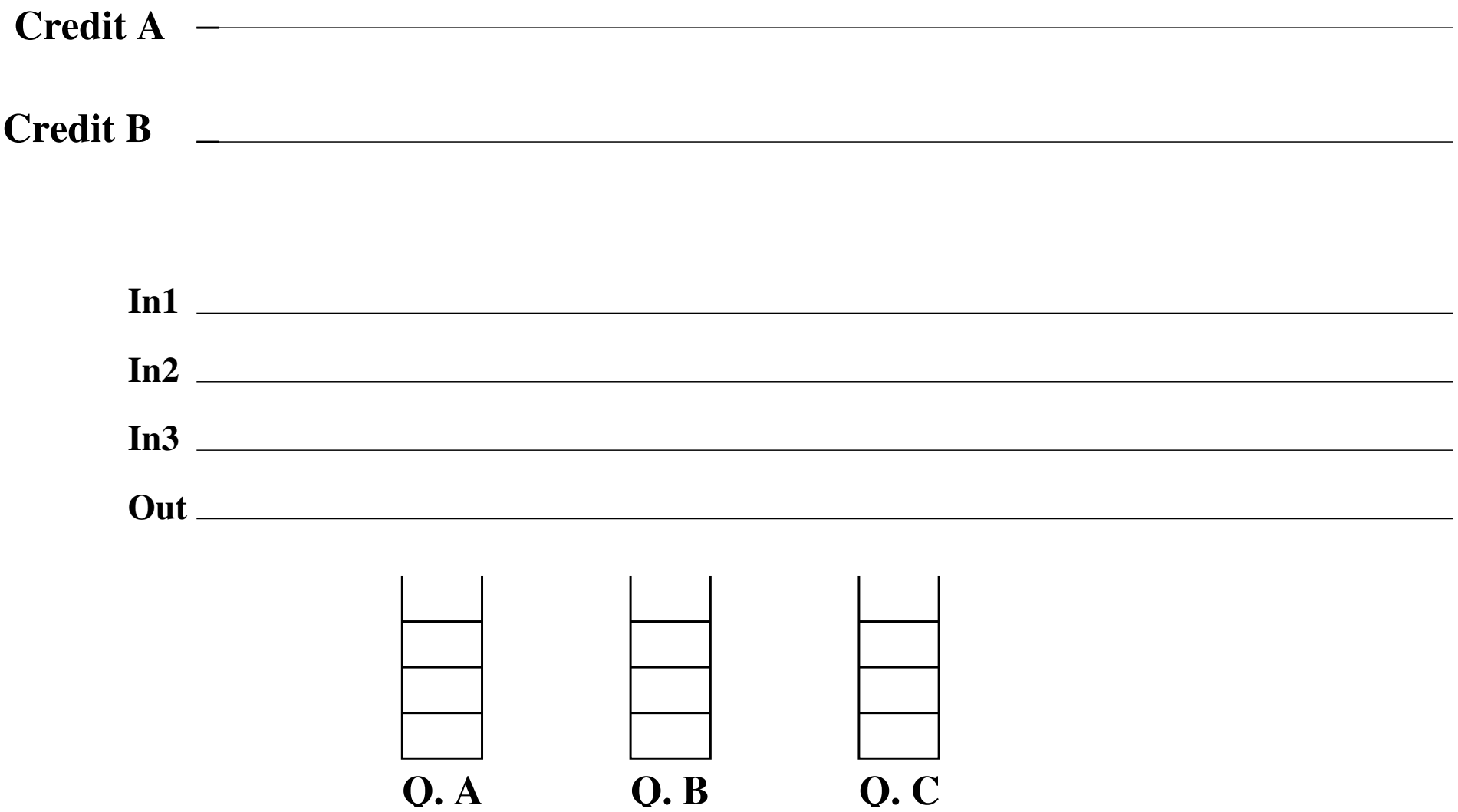
- Goal: limit starvation problem $\Rightarrow$ improve fairness between flows
- Limit bursts for highest priority queues



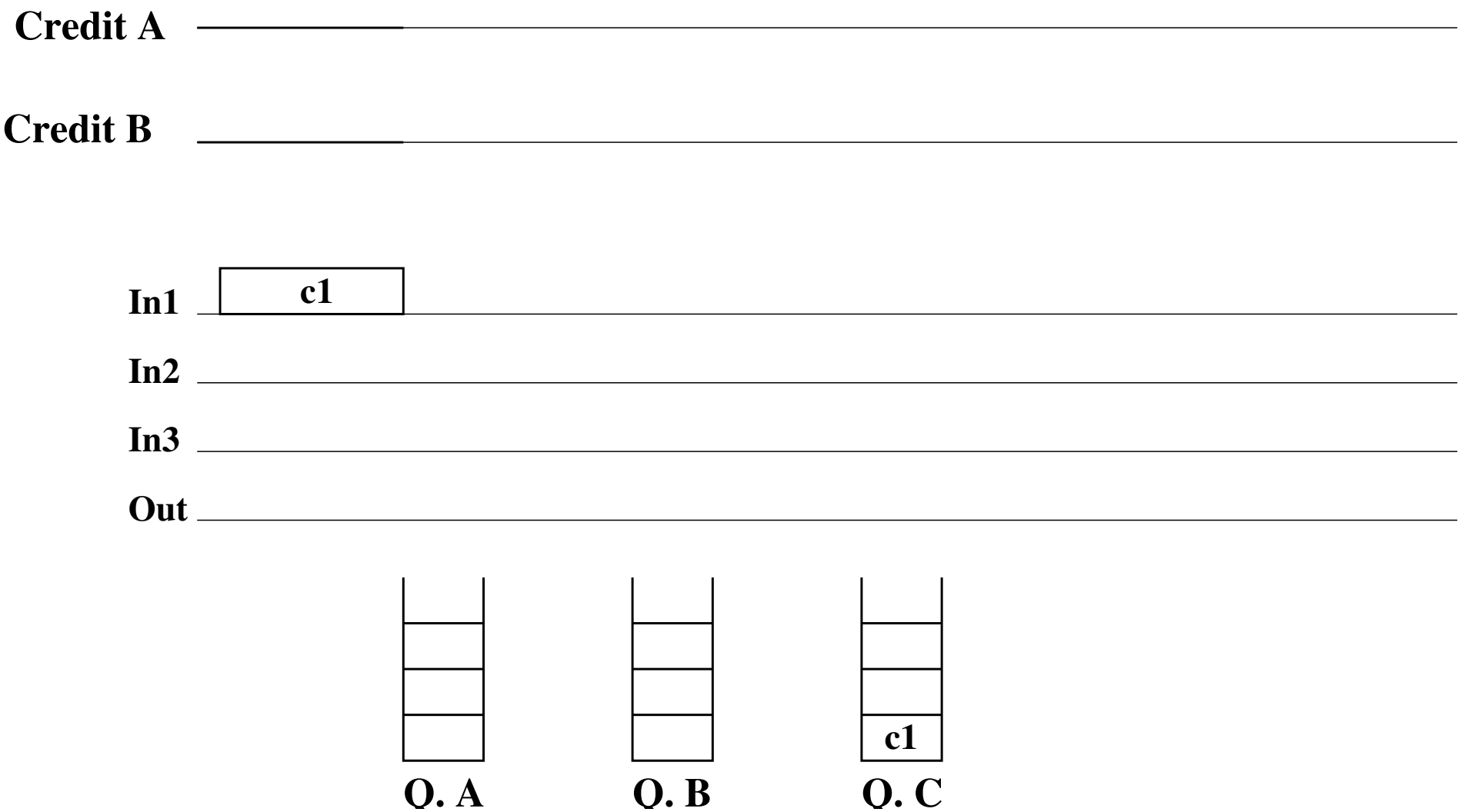
Credit Based Shaper (CBS)

- Rules
  - Credit for a given class initialized to 0
  - When no frame in the queue and credit positive, credit set to 0
  - Credit decreased by *sendslope* when a frame is being transmitted
  - Credit increased by *idleslope* when
    - ★ there is at least one frame in the queue and
    - ★ no frame from the queue is being transmitted

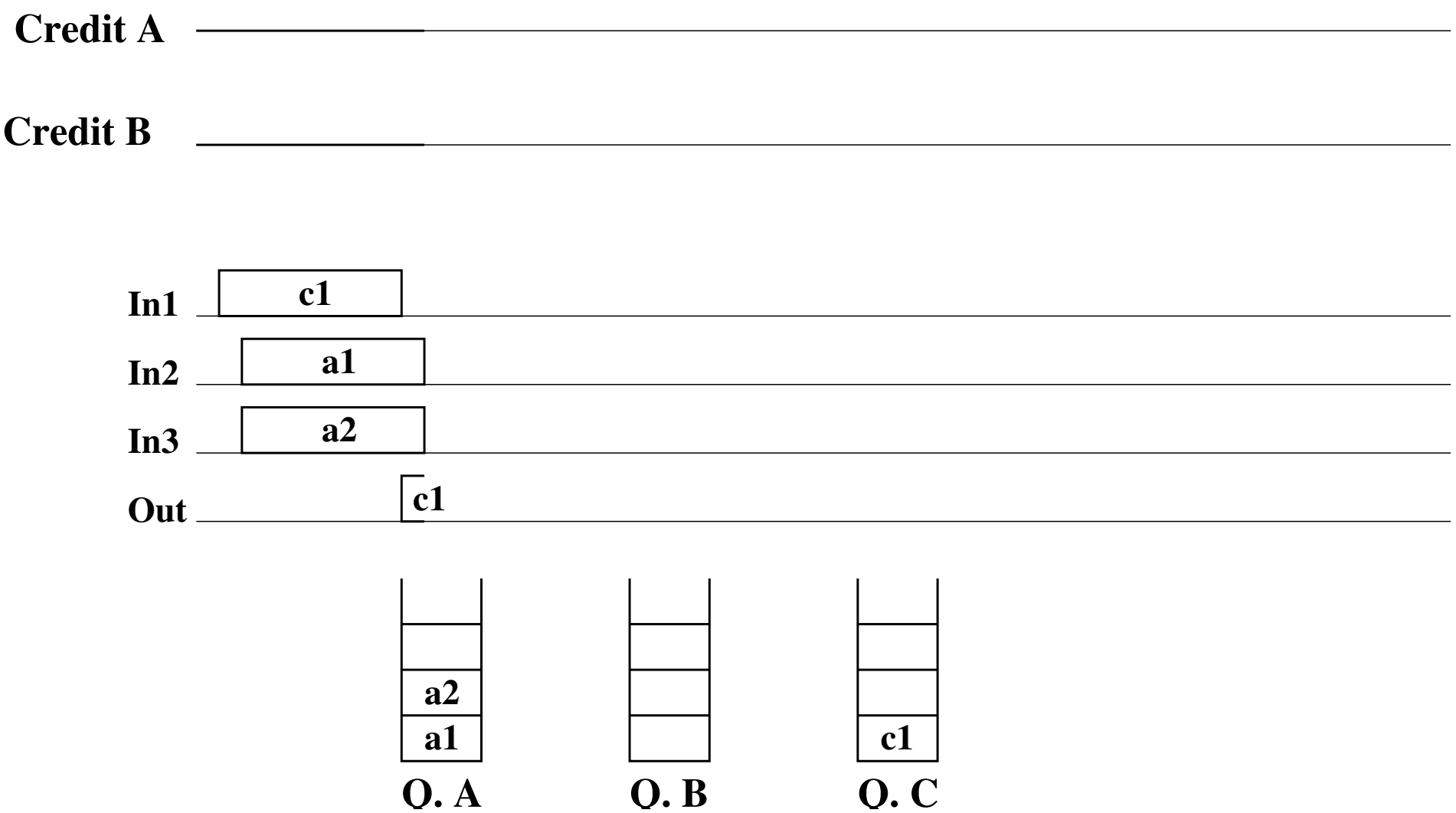
Credit Based Shaper (CBS)



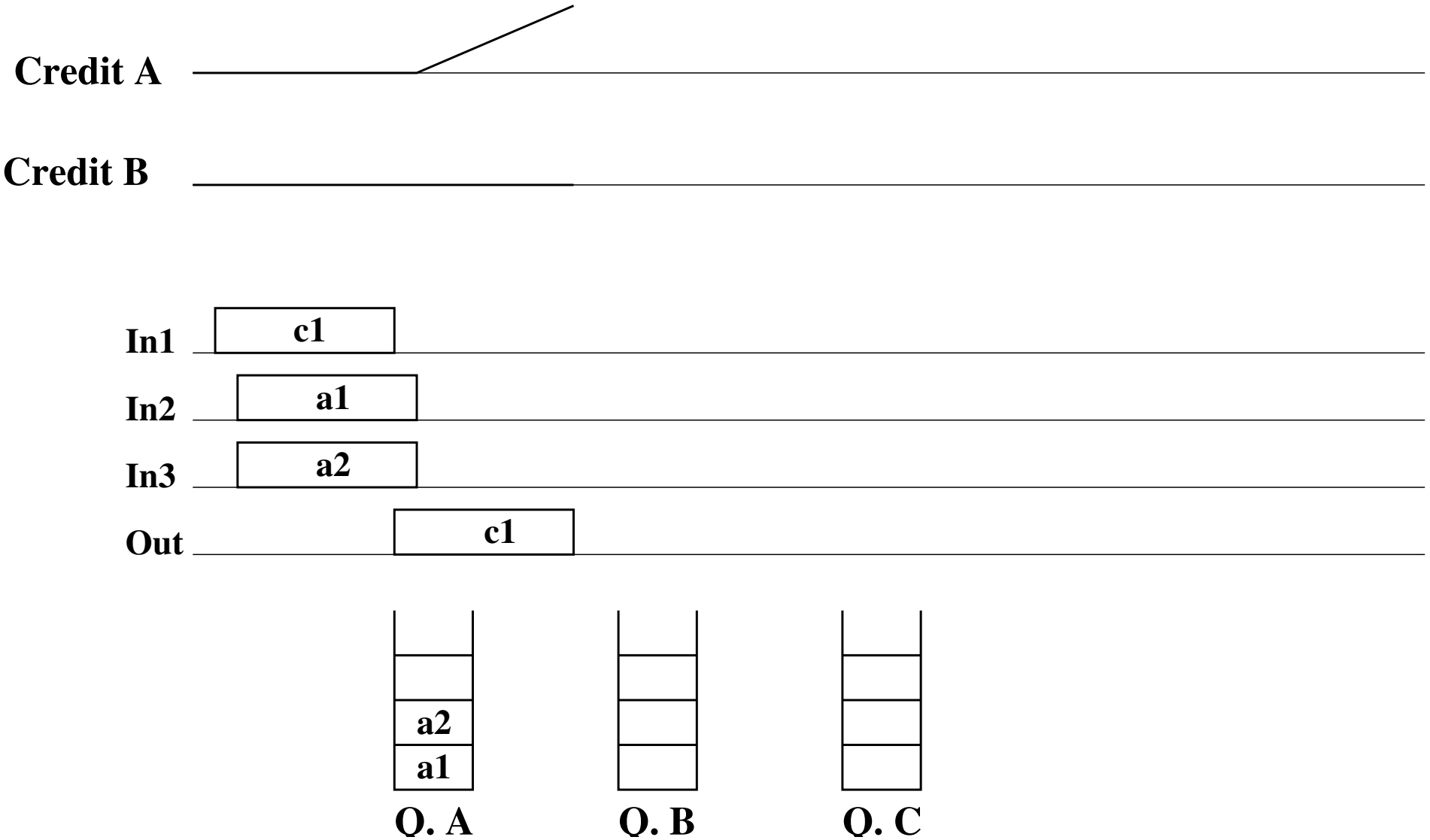
Credit Based Shaper (CBS)



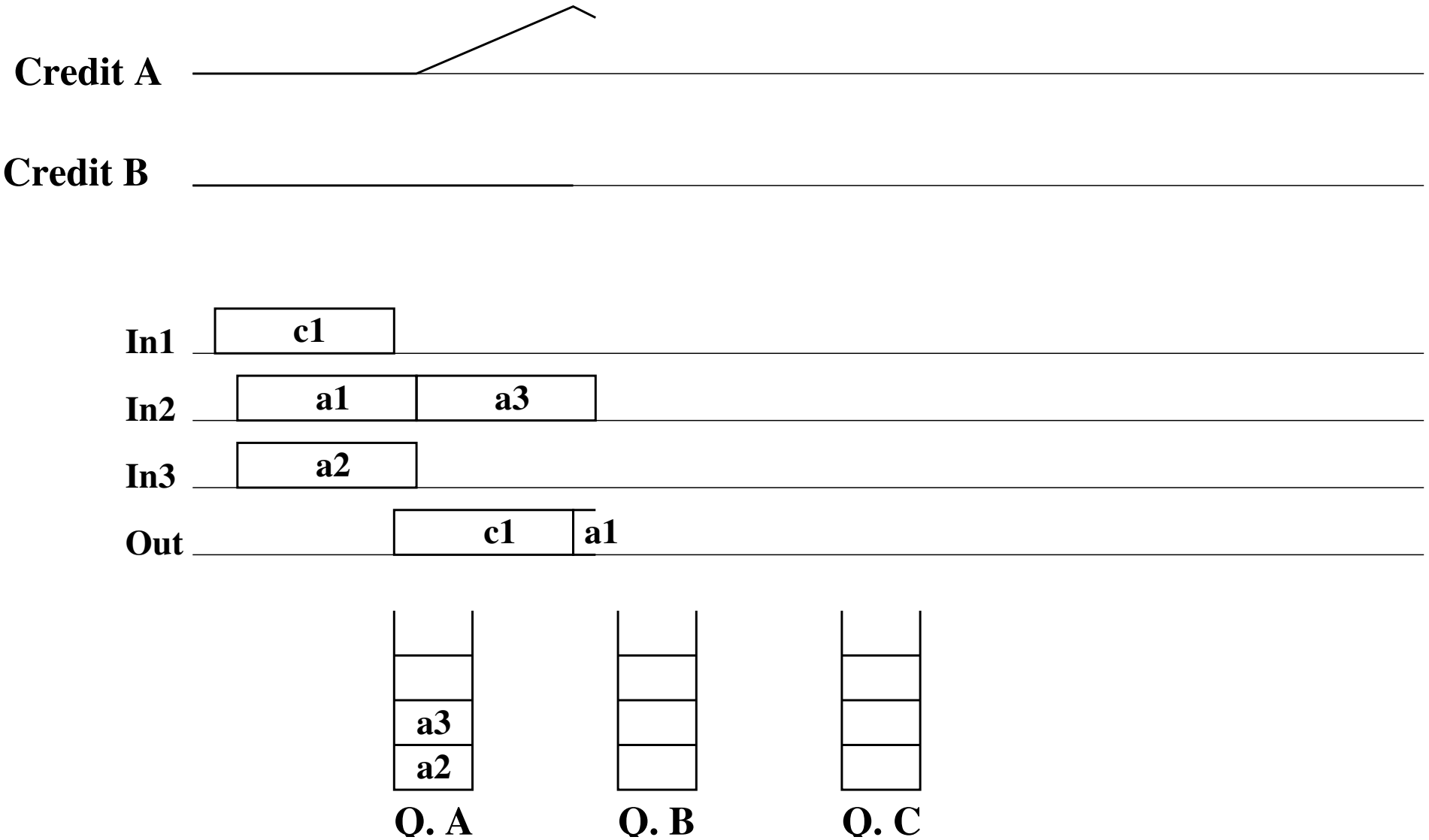
Credit Based Shaper (CBS)



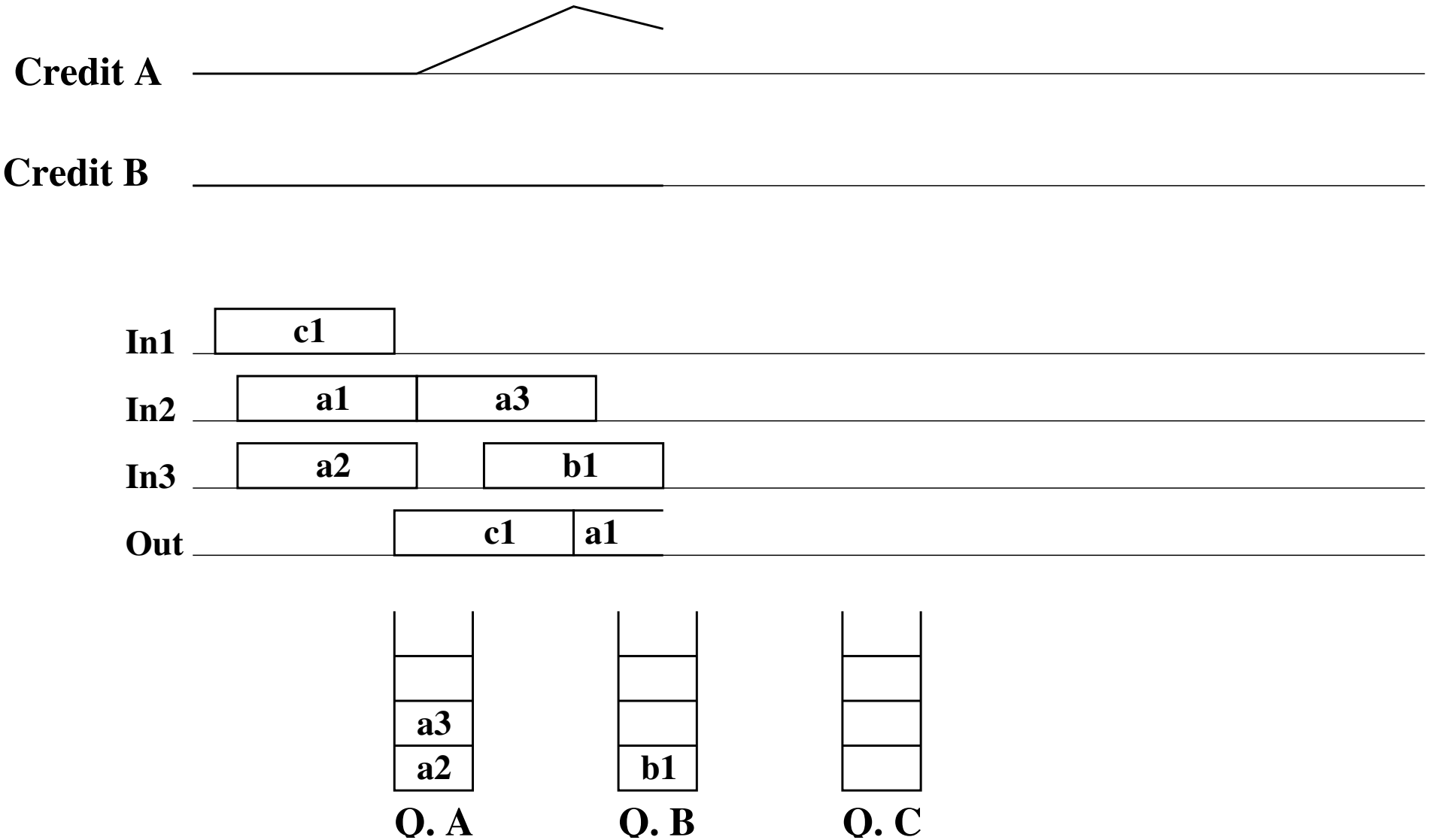
Credit Based Shaper (CBS)



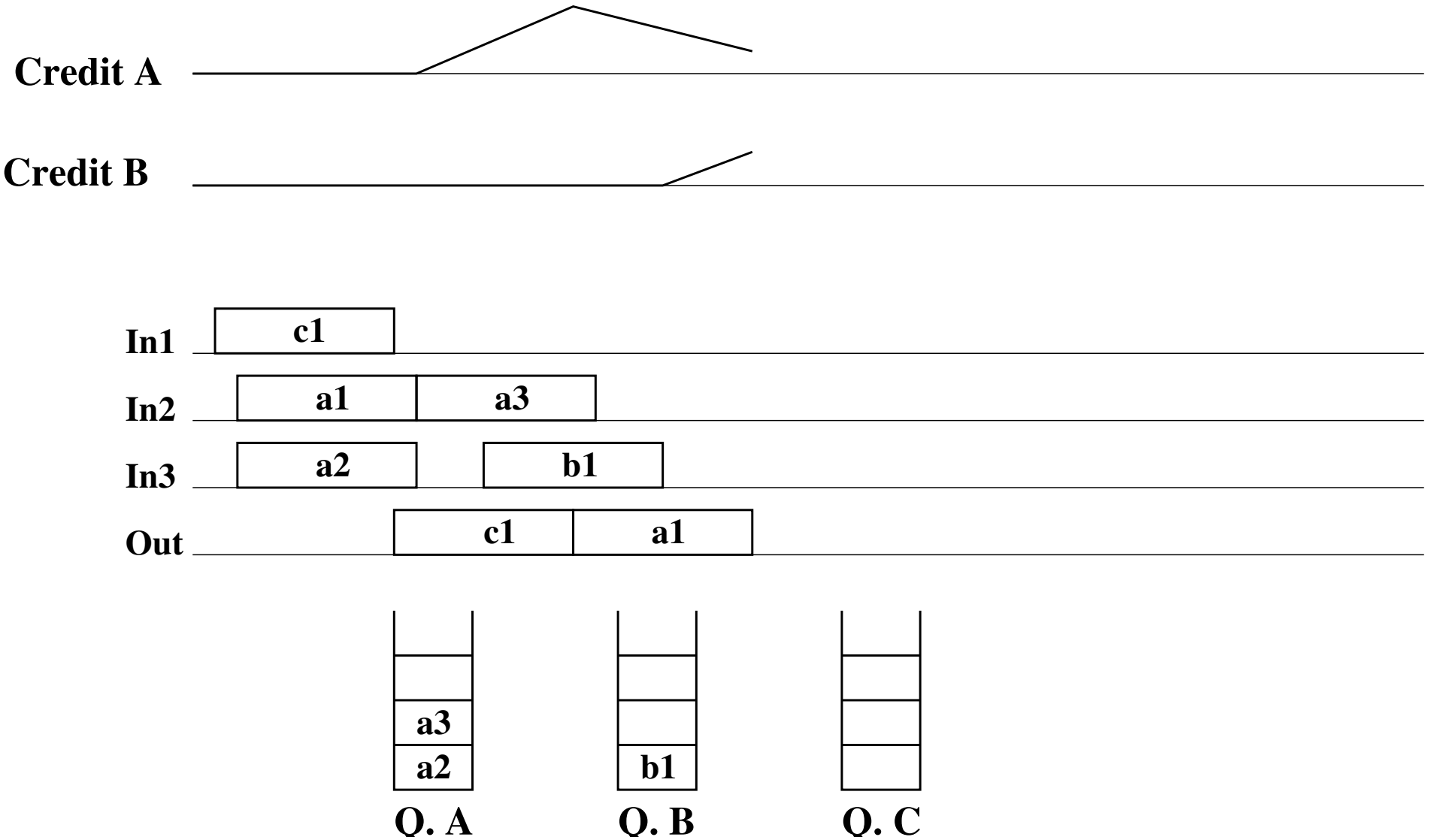
Credit Based Shaper (CBS)



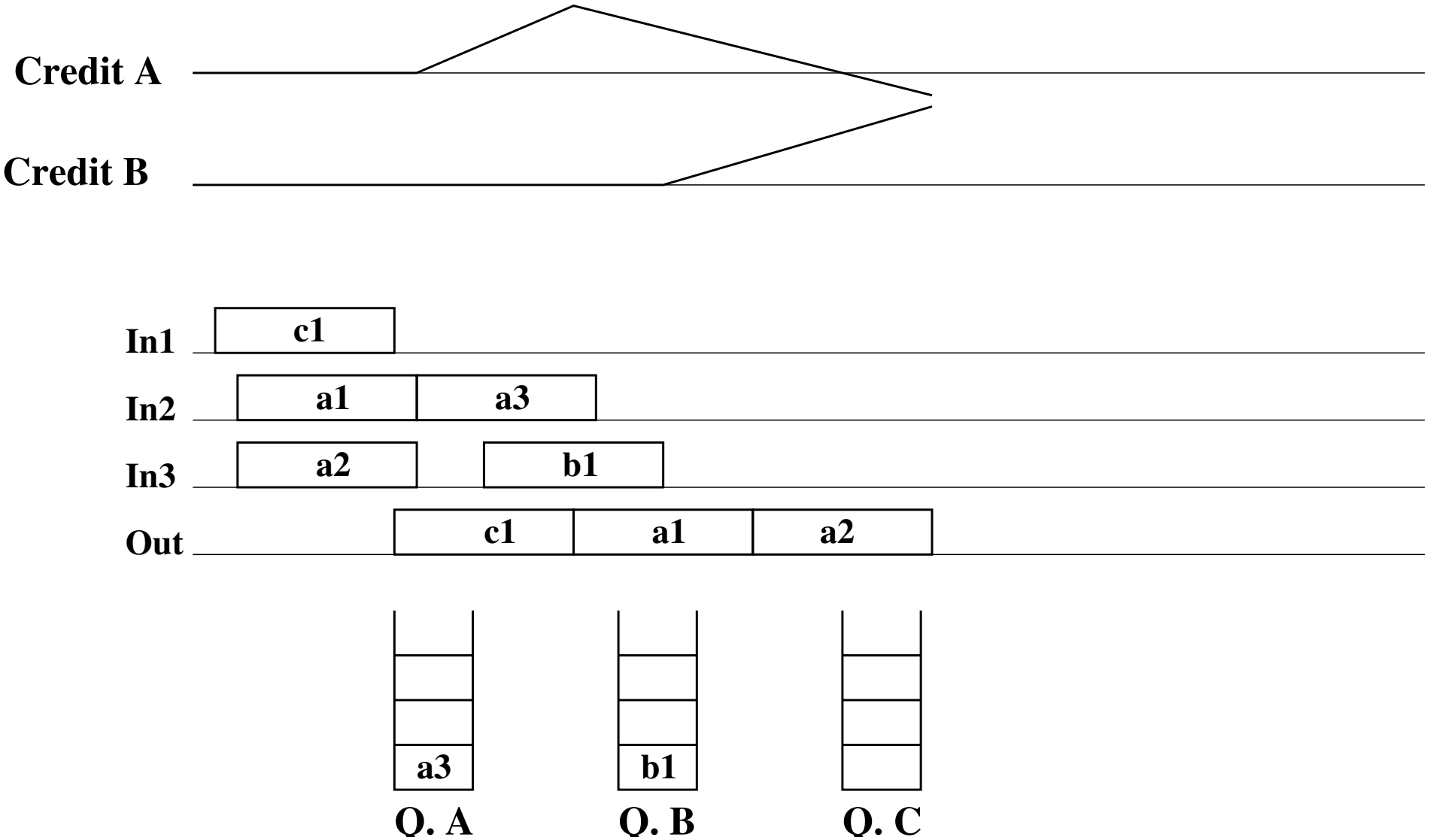
Credit Based Shaper (CBS)



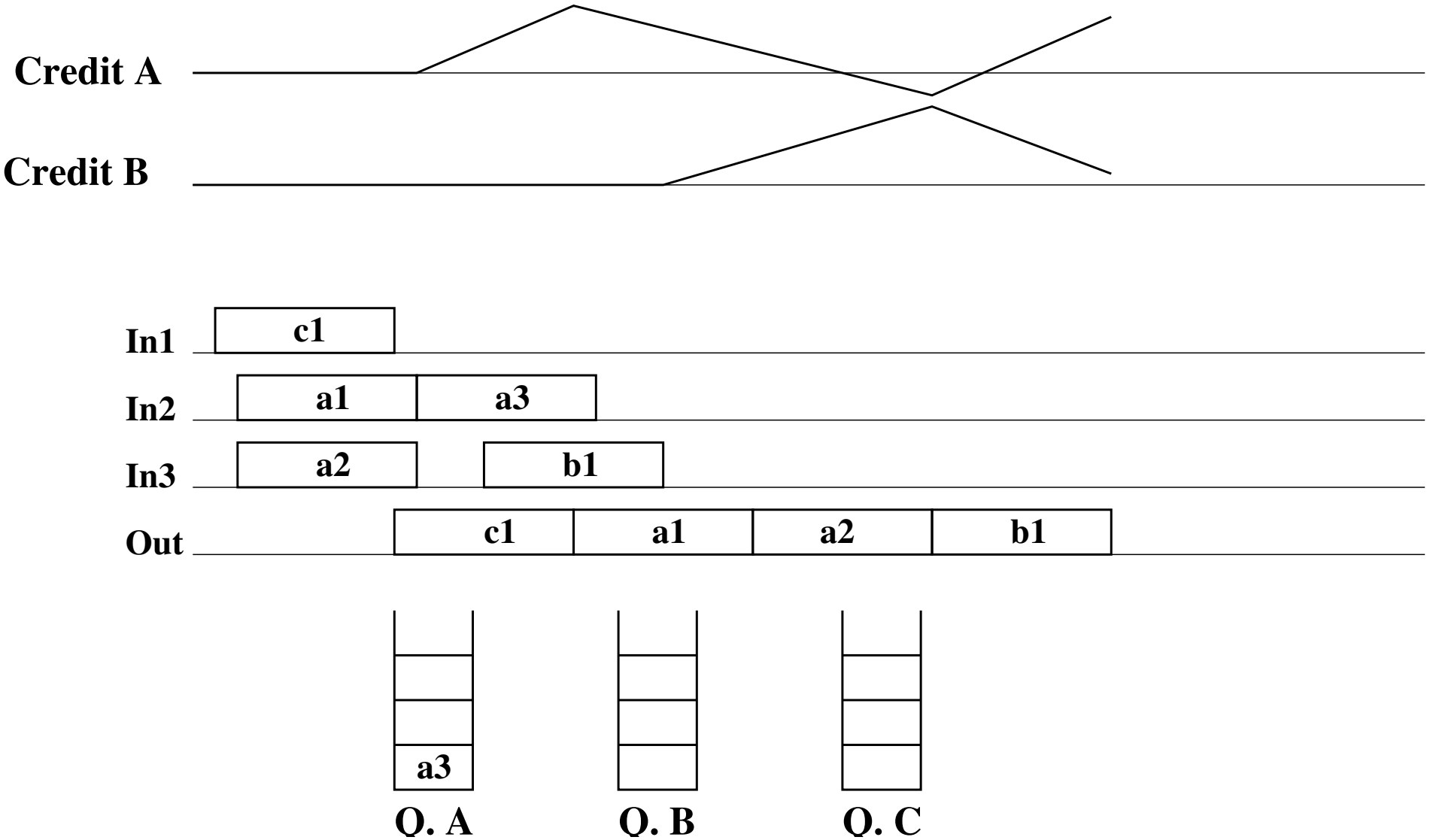
Credit Based Shaper (CBS)



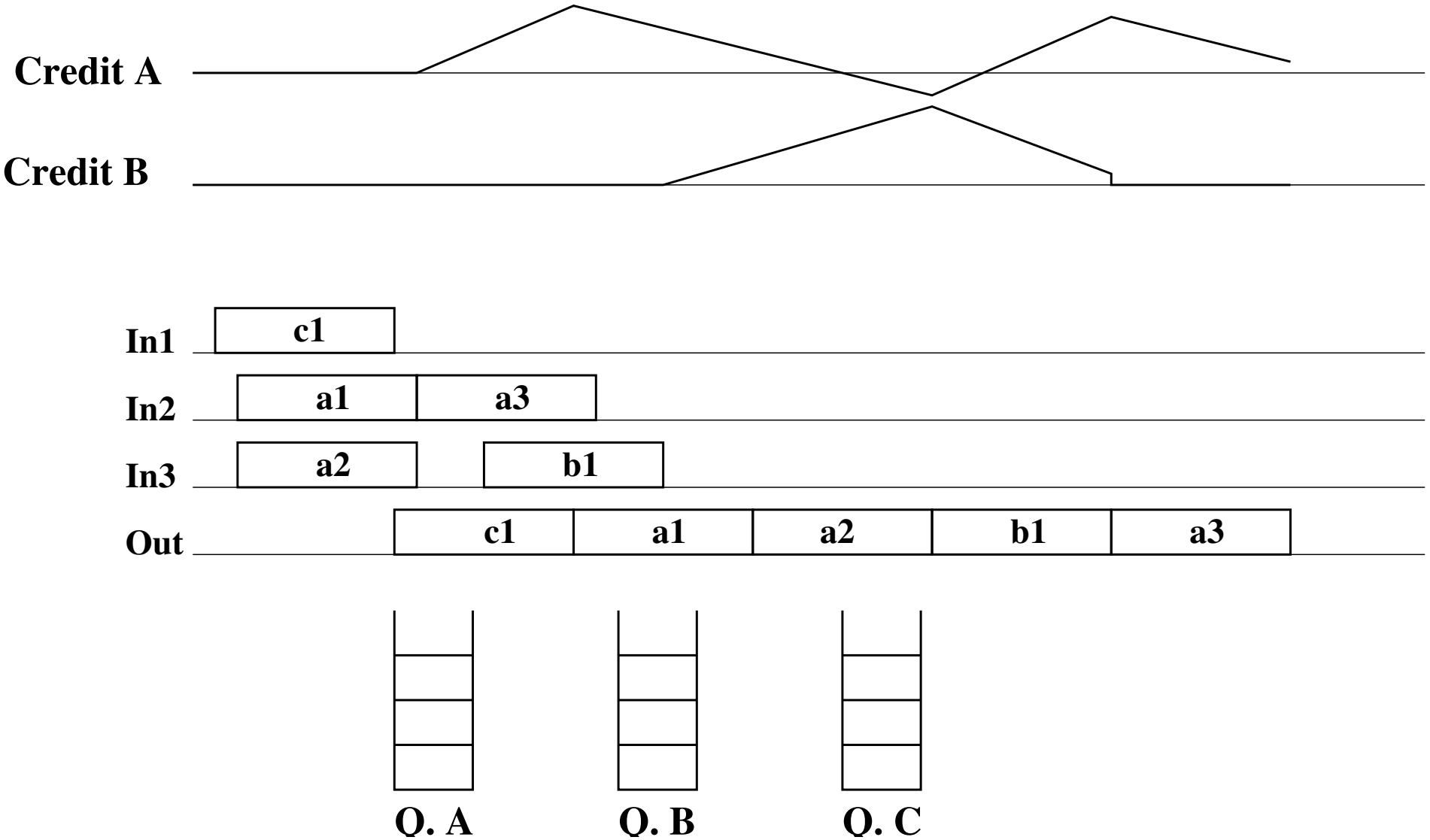
Credit Based Shaper (CBS)



Credit Based Shaper (CBS)

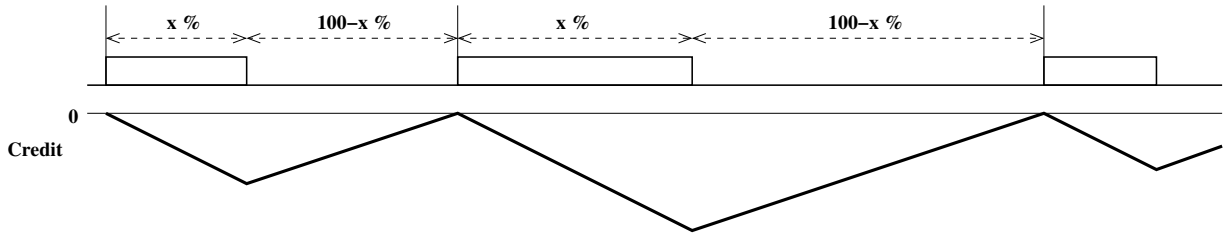


Credit Based Shaper (CBS)



Configuration of SendSlope and IdleSlope

- Reservation of a percentage of the bandwidth for the associated class



- Send slope: the credit is decreased by s every time unit
- Idle slope: the credit is increased by i every time unit
- We have: $s \times x = i \times (100 - x)$
- Thus:

$$i = \frac{s \times x}{100 - x}$$

- Some examples

x (%)	s	i
50	1	1
80	1	4
25	3	1

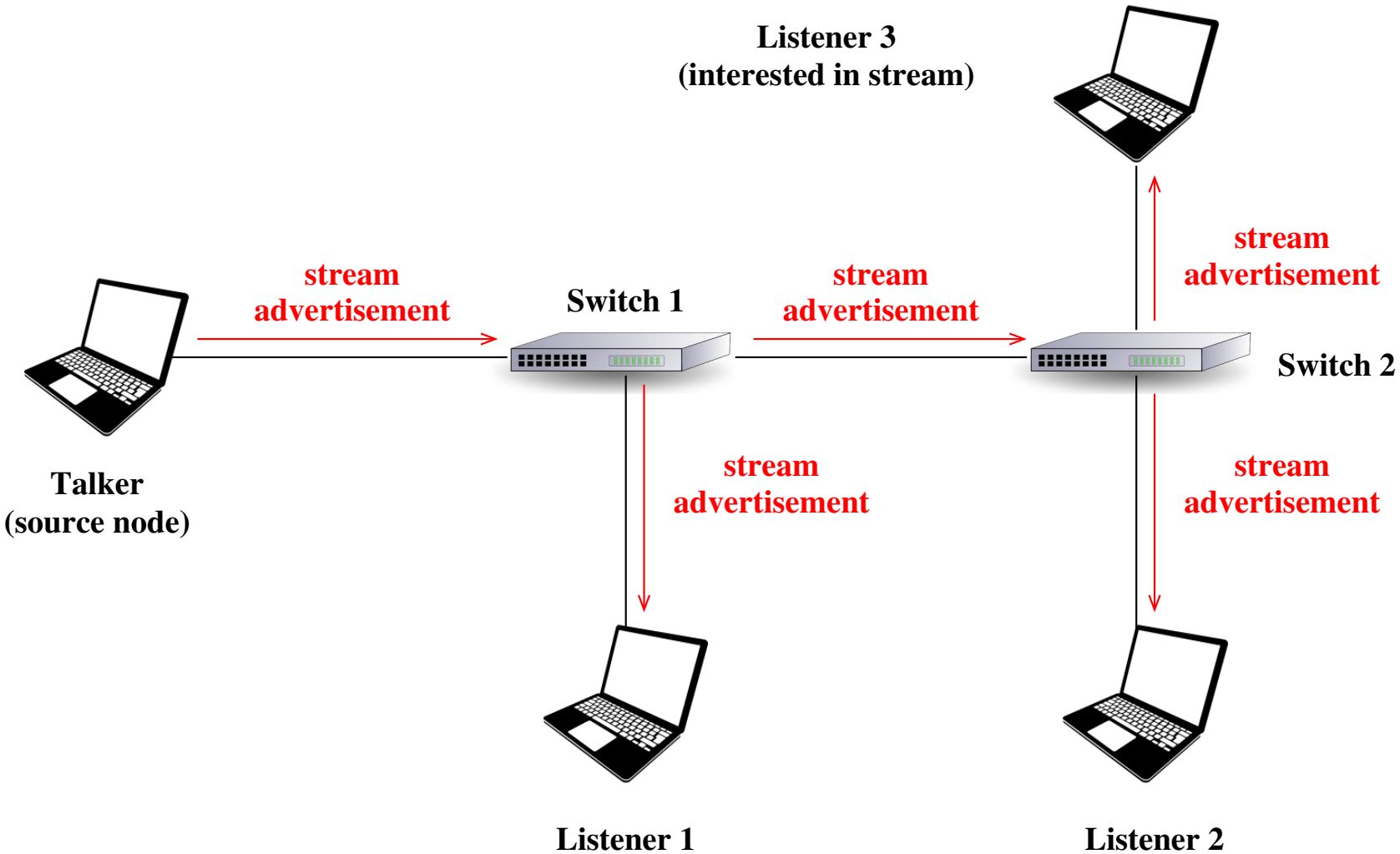
Exercise: SPQ vs AVB/CBS

- Two traffic classes at a given switch output port
  - Class 1 with priority 1 (high): 3 flows
    - ★ Flow 1: first frame at $t = 0\ \mu s$, $period = 250\ \mu s$, $Tx = 100\ \mu s$
    - ★ Flow 2: first frame at $t = 10\ \mu s$, $period = 1000\ \mu s$, $Tx = 100\ \mu s$
    - ★ Flow 3: first frame at $t = 20\ \mu s$, $period = 1000\ \mu s$, $Tx = 100\ \mu s$
  - Class 2 with priority 2 (high): 1 flow
    - ★ Flow 4: first frame at $t = 0\ \mu s$, $period = 250\ \mu s$, $Tx = 100\ \mu s$
- What is the order of transmission with Strict Priority Queueing?
- What is the order of transmission with AVB and a credit base shaper for class1 such that it gets no more that 60 % of the bandwidth?
  - First, you have to determine the idle slope and the send slope for class 1

Stream Reservation Protocol (SRP)

- IEEE 802.1Qat standard
- Allocation of network resources for a specific streaming data
- Required to ensure that QoS requirements are met
- Signaling protocol to register or de-register a specific streaming application $\Rightarrow$ allocate or release bandwidth in a whole AVB network
- Two stream reservation (SR) classes
  - SR-Class A
    - ★ Worst-case latency requirement over seven hops: 2 ms
  - SR-Class B
    - ★ Worst-case latency requirement over seven hops: 50 ms
- At most 75 % of the total bandwidth allocated for both SR classes $\Rightarrow$ At least 25 % of the capacity for other classes

SRP stream advertisement



SRP stream advertisement

- *Talker advertise frame* with the traffic specifications (TSpec)
  - *StreamID*: unique identifier for a specific stream
  - *MaxFrameSize (MFS)*: maximum payload size of each frame transporting the stream
  - *MaxIntervalFrames (MIF)*: amount of frames in each interval time
    - SR-Class A: $IntervalTime = 125\ \mu s$
    - SR-Class B: $IntervalTime = 250\ \mu s$
- Required bandwidth of a streaming data at each port

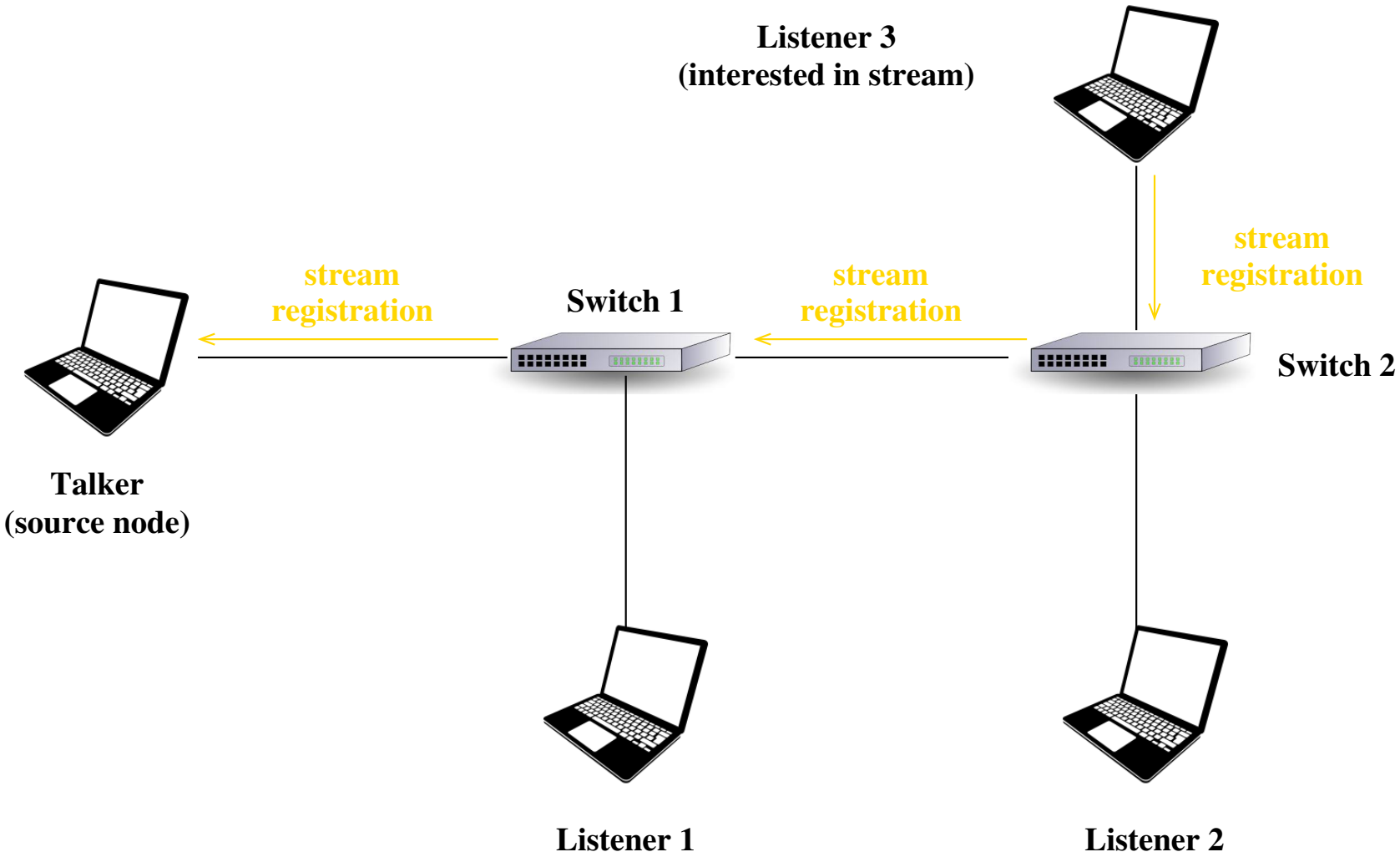
$$StreamBW = \frac{(MFS + OH) \times MIF}{IntervalTime}$$

- *OH*: overhead of a streaming frame (Ethernet header, CRC, ...), typically 42 bytes
- Output port of a switch has enough resources $\Rightarrow$ the frame is forwarded without any modification
- Output port of a switch has no sufficient resources $\Rightarrow$ frame is modified to *talker failed frame* and forwarded

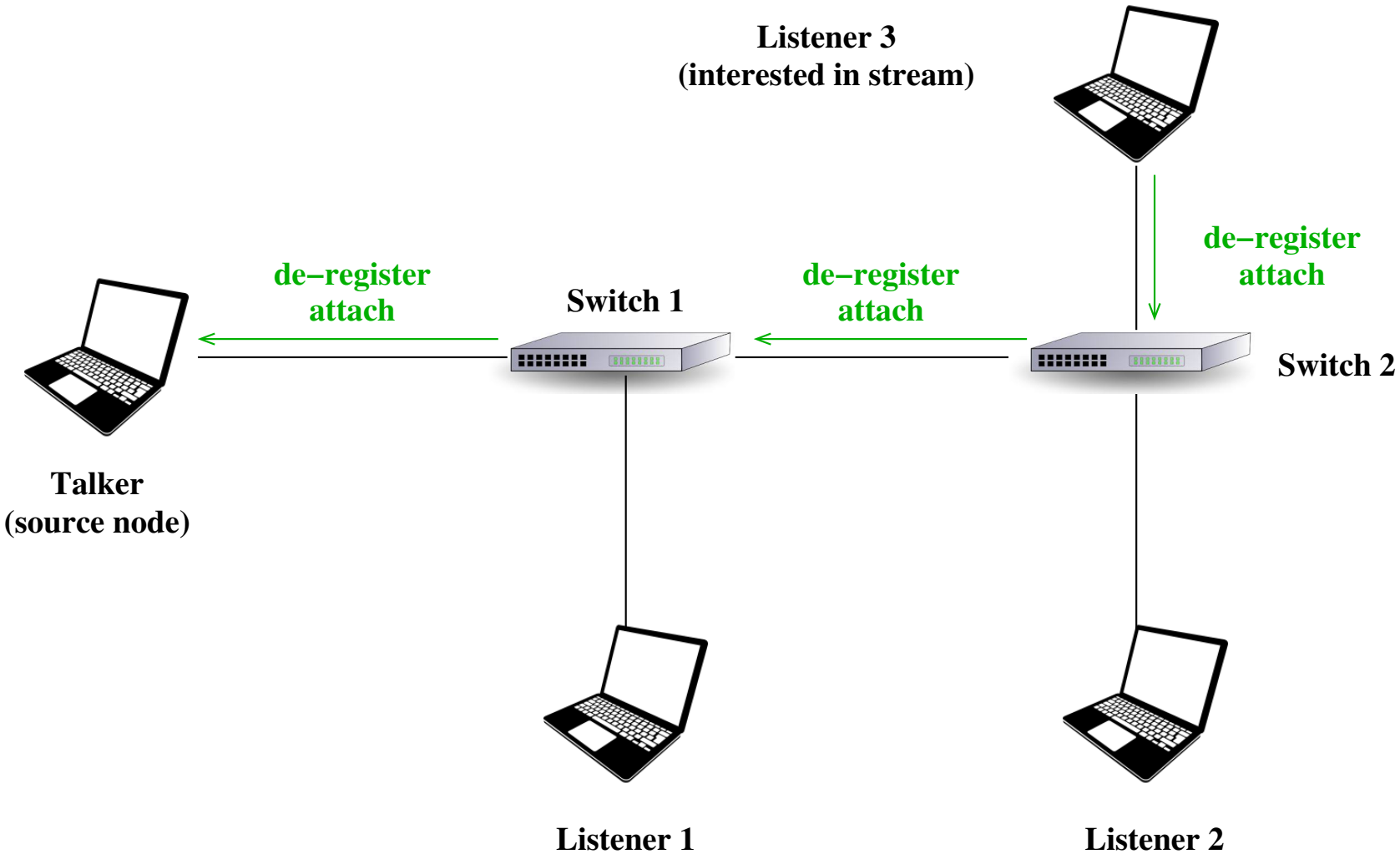
Stream examples

Source	Raw Bit Rate	Framing overhead	MFS	MIF
48 kHz stereo audio stream SR-Class A	3 Mb/s	4.7 Mb/s	80	1 (8000 frames/s)
96 kHz stereo audio stream SR-Class A	6 Mb/s	4.7 Mb/s	128	1 (8000 frames/s)
MPEG2-TS video SR-Class B	24 Mb/s	2.5 Mb/s	786	1 (4000 frames/s)
SD SDI (level C) uncompressed SR-Class A	270 Mb/s	15 Mb/s	1442	3 (24000 frames/s)
SD SDI (level D) uncompressed SR-Class A	360 Mb/s	20 Mb/s	1442	4 (32000 frames/s)

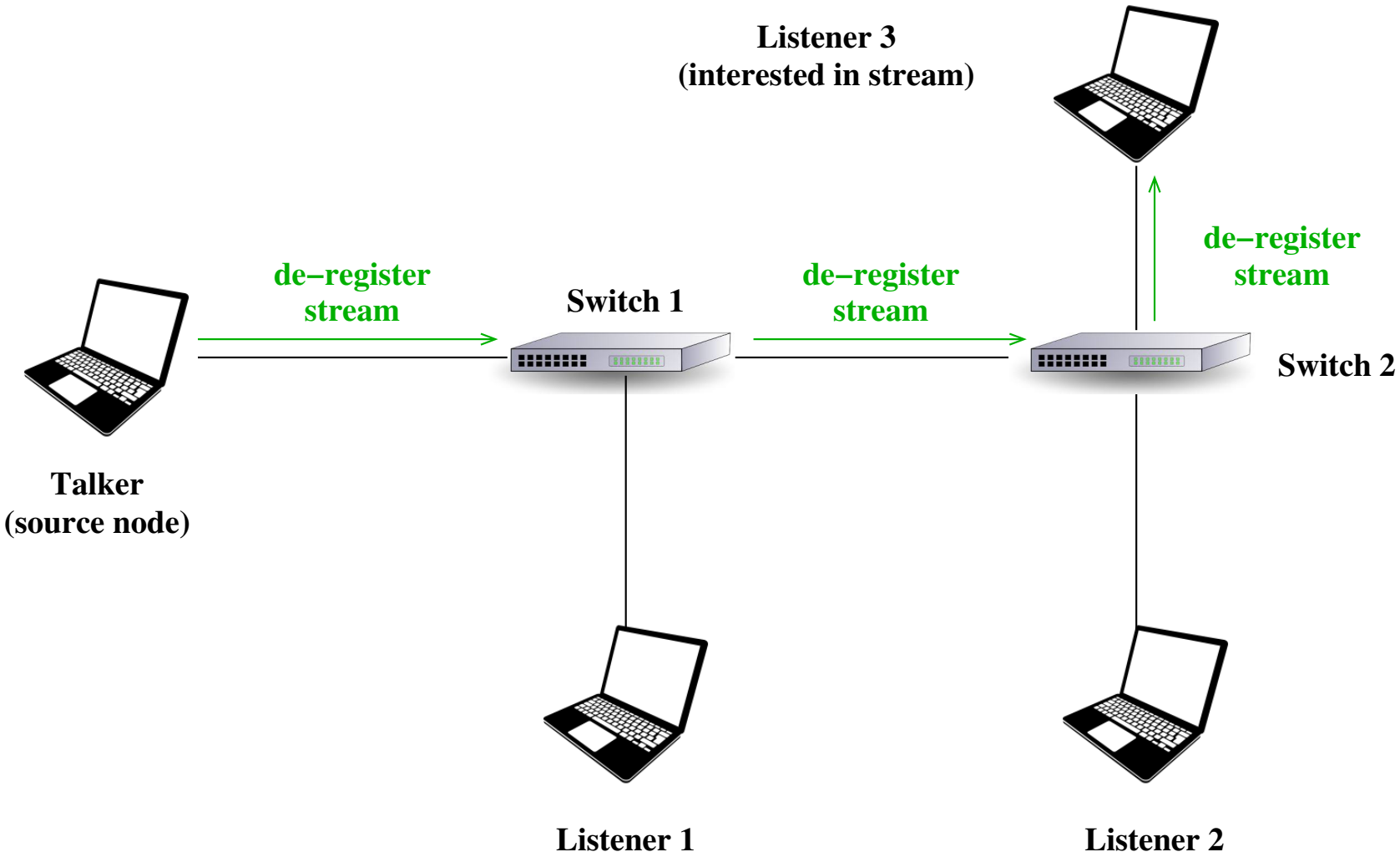
Stream registration



Stream de-registration by listener

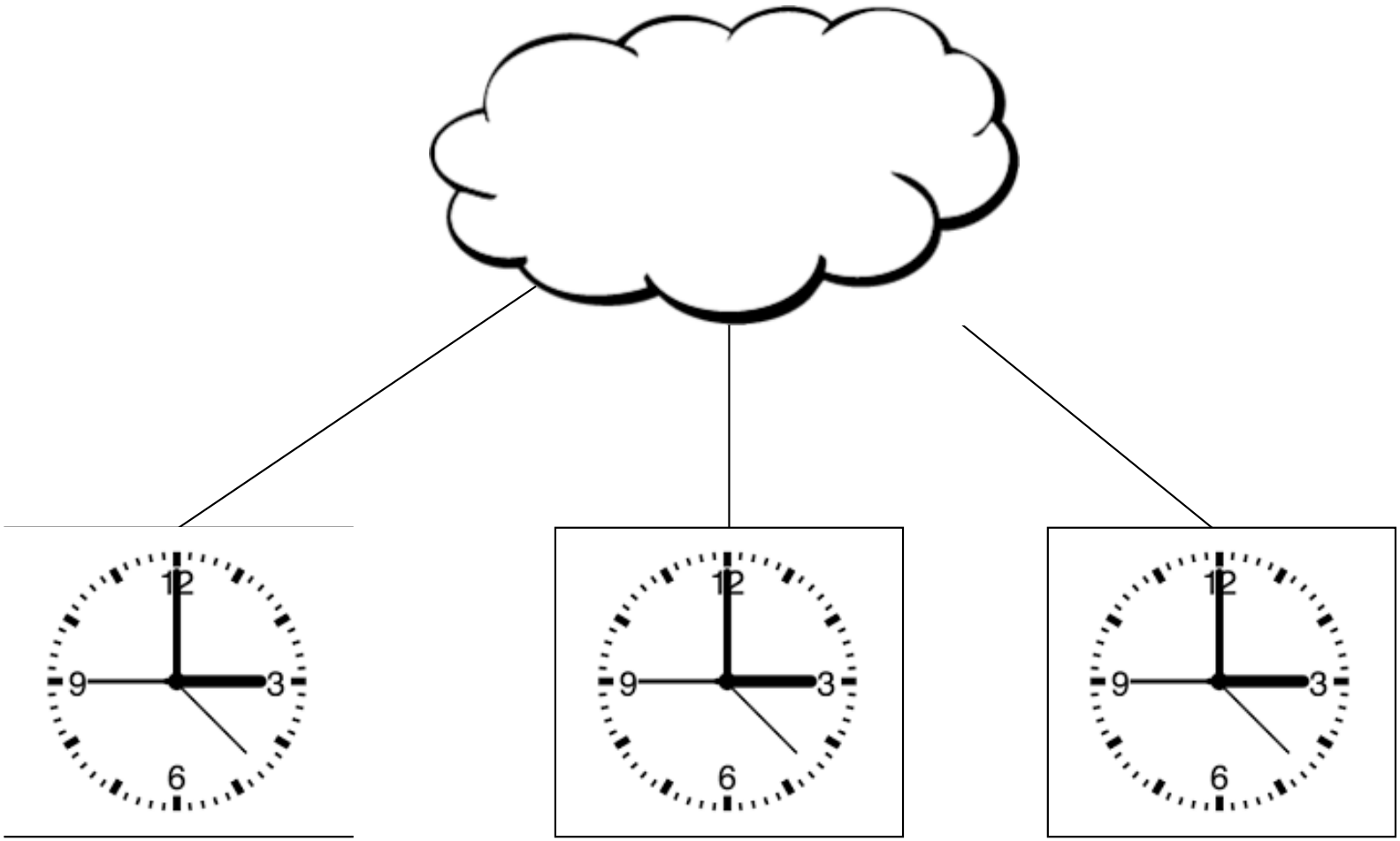


Stream de-registration by talker



IEEE-1588: precision clock synchronization protocol

- Protocol designed to synchronize real-time clocks on the nodes of a distributed system that communicate using a network

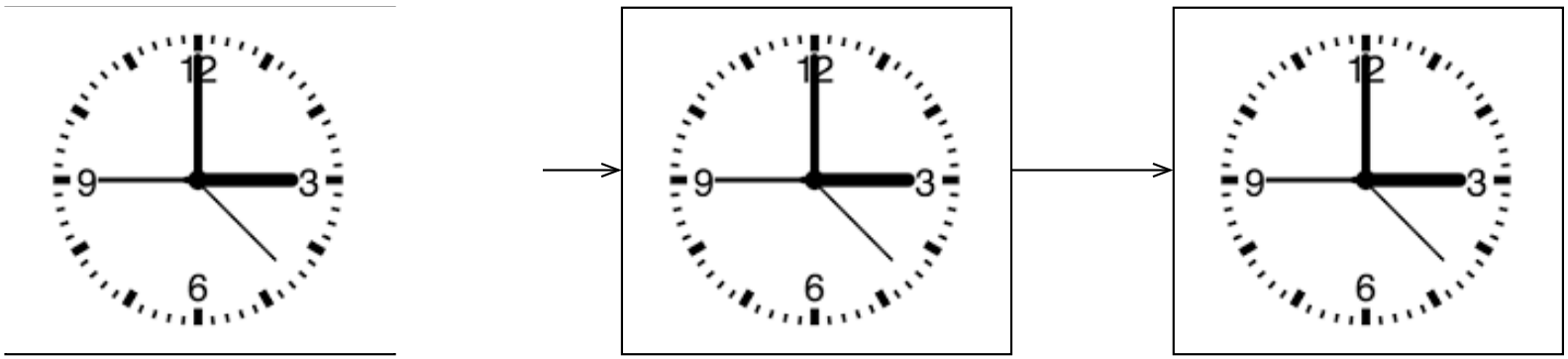


Objectives of IEEE-1588

- Sub-microsecond synchronization of real-time clocks in components of a networked distributed measurement and control system
- Applicable to local area networks supporting multicast communications (including but not limited to Ethernet)
- Simple, administration free installation
- Support heterogeneous systems of clocks with varying precision, resolution and stability
- Minimal resource requirements on networks and host components

IEEE-1588 basics

- Step1: organize the clocks into a master-slave hierarchy (based on observing the clock property information contained in multicast sync messages)
- Step2: Each slave synchronizes to its master

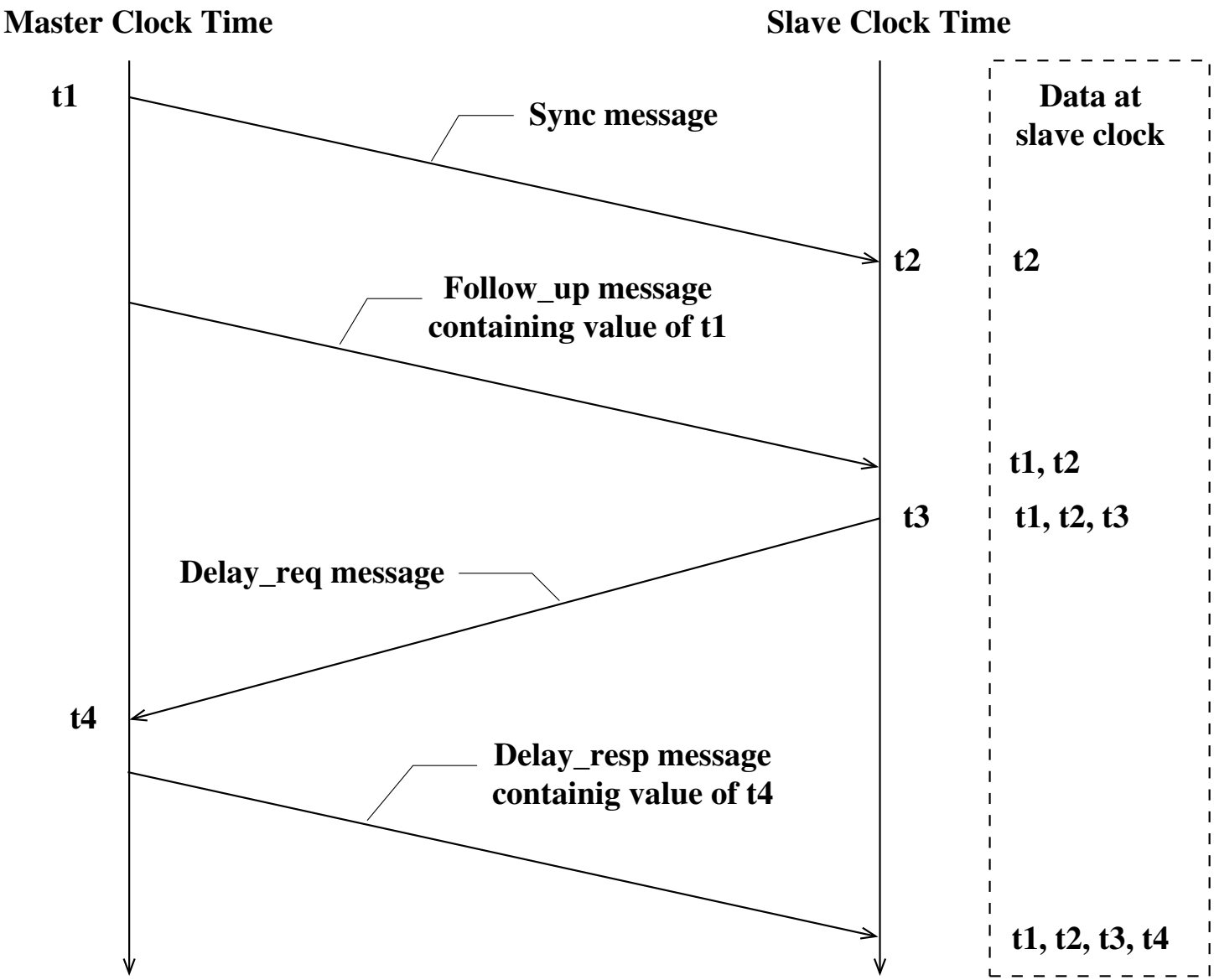


Grandmaster clock
Time base for
the system

Slave to the
grandmaster clock
and master to its slave

Slave to its master

IEEE-1588 synchronization basics



IEEE-1588 synchronization basics

- To synchronize a pair of clocks
  - Send a Sync message from master to slave and measure the apparent time difference between the two clocks

$$\begin{aligned} \text{MS\_difference} &= \text{slave's receipt time} - \text{master's sending time} \\ &= t2 - t1 \\ &= \text{offset} + \text{MS\_delay} \end{aligned}$$

  - Send a Follow_up message with precise value of $t1$
  - Send a Delay_req message from slave to master and measure the apparent tim difference between the two clocks

$$\begin{aligned} \text{SM\_difference} &= \text{master's receipt time} - \text{slave's sending time} \\ &= t4 - t3 \\ &= -\text{offset} + \text{SM\_delay} \end{aligned}$$

  - Send a Delay_resp message

IEEE-1588 synchronization basics

- Two measured quantities
  - MS_difference and SM_difference
- Two equations
  - MS_difference = offset + MS_delay
  - SM_difference = − offset + SM_delay
- Determine offset, MS_delay and SM_delay
- Rearranging the equations, we get
  - offset = MS_difference − MS_delay
  - offset = SM_delay − SM_difference
- Thus
  - offset = {(MS_difference − SM_difference) − (MS_delay − SM_delay)}/2
- We also have
  - MS_delay + SM_delay = MS_difference + SM_difference
- Assume
  - MS_delay = SM_delay = one_way_delay
- Then
  - offset = (MS_difference − SM_difference)/2
  - one_way_delay = (MS_difference + SM_difference)/2

Exercise on IEEE-1588

- We assume a master and a slave
- The master initiates a synchronisation with the slave at time 2000 for its (master) clock and time 1000 for the slave clock
- The delay from master to slave or from slave to master is 100

Detail the process and the computation leading to the offset

Selecting a Master Clock

- All nodes run an identical **Best Master Clock Algorithm (BMCA)**
- A master clock issues periodic Sync messages (sync_interval)
- A master clock may receive Sync messages from other clocks (which think they are master) $\Rightarrow$ it calls them foreign masters
- Each master clock uses the BMCA to determine whether it should remain master or not
- Each non-master clock uses the BMCA to determine whether it should become a master

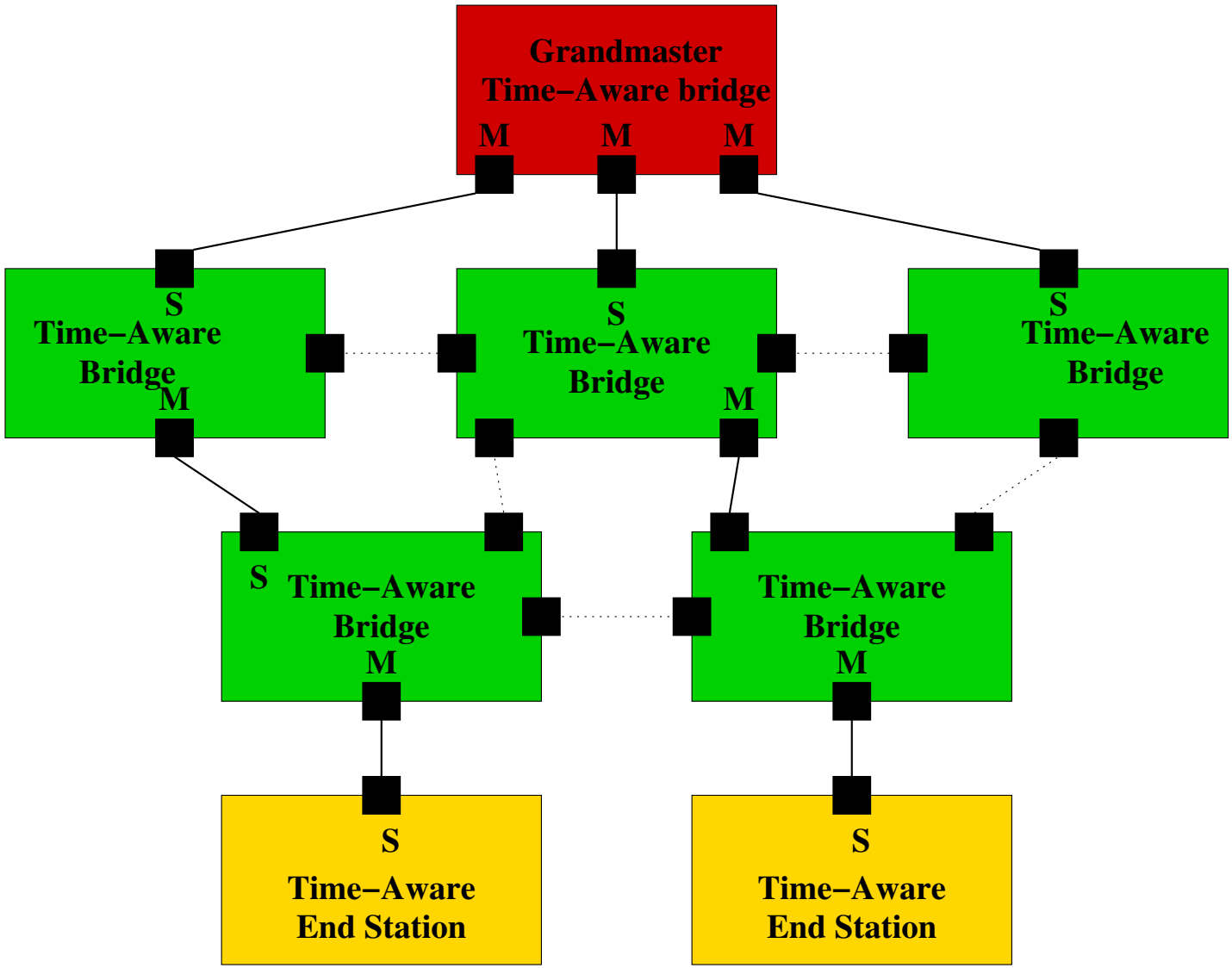
BMCA overview

1. A master clock 'A' can receive Sync messages from other potential master clocks 'B', 'C', ...
2. Clock 'A' decides:
   1. Which of the clocks 'B', 'C', ... is the best clock
   2. Whether clock 'A' is better than the best of 'B', 'C', ...
3. Using the BMCA, it does this by pair wise comparisons of the data sets describing each of the clocks
4. Based on the results of this comparison the BMCA returns a recommended clock state: typically either master or slave
5. All clocks operate on the same information and therefore arrive at consistent results
6. Data for these comparison is maintained by each clock in data sets

Characterization of clocks

- Primary source of time
  - GPS
  - Local oscillator
  - ...
- Accuracy
- Variance
- UUID (tie-breaking)
- ...

Resulting spanning tree by the BMCA

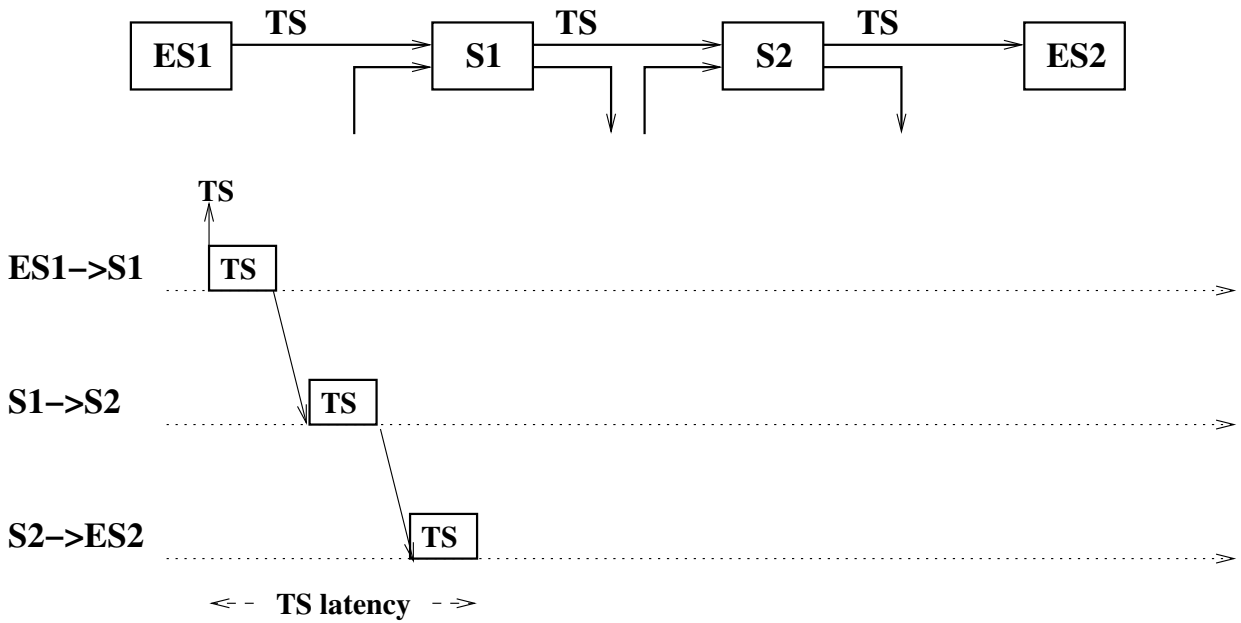


Need of a mechanism for Time-sensitive traffic

- Time-sensitive traffic requires deterministic and very small delays
  - Ex: delay-sensitive command and control traffic
- Procedures to suspend the transmission of a non-time critical frame and allow for one or multiple critical frames to be transmitted
- Policies to support scheduled traffic
- Requires to schedule frame transmission based on time

Latency assessments for time-sensitive traffic

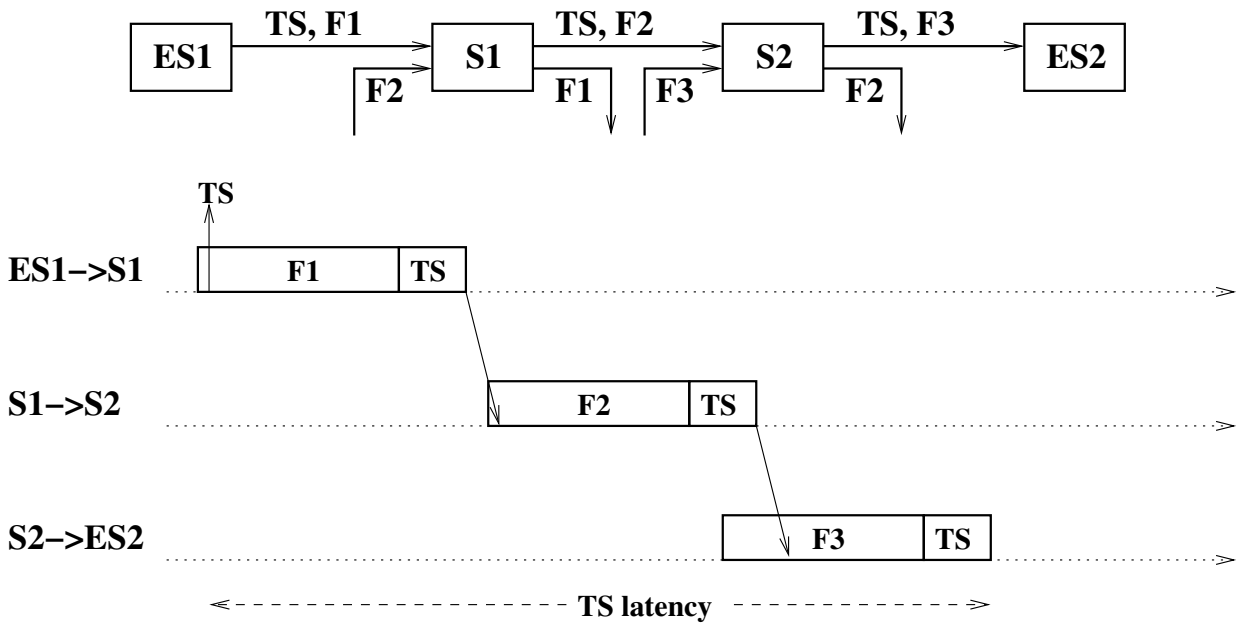
- Without interfering traffic
  - Maximum latency of a time-sensitive frame mainly depends on
    - ★ Transmission time of the time-sensitive frame on links (computed from the frame size)



- Good news since time-sensitive flows typically feature small frames

Latency assessments for time-sensitive traffic

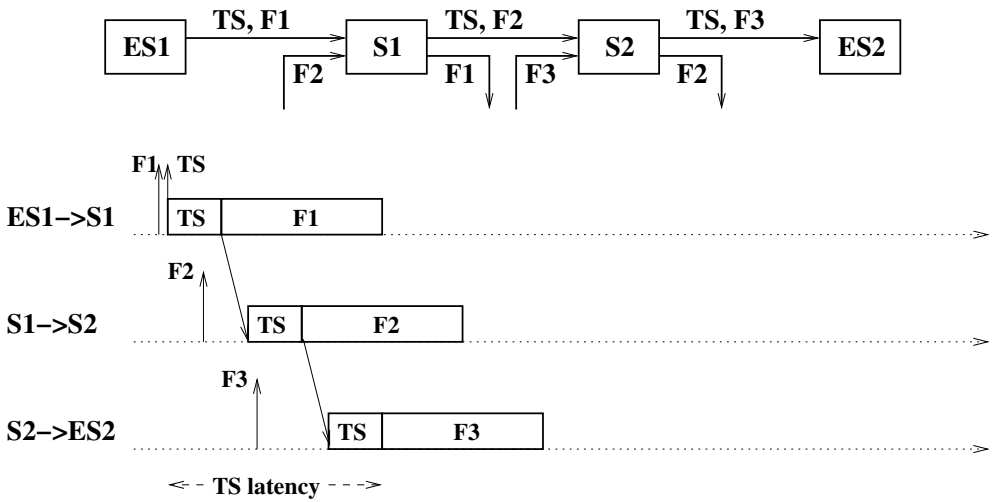
- With interfering traffic
  - Maximum latency of a time-sensitive frame also depends on
    - ★ Maximum size of the interfering frames, due to non-preemptive scheduling at the egress port of each network device



- Bad news since non time-sensitive frames might be large

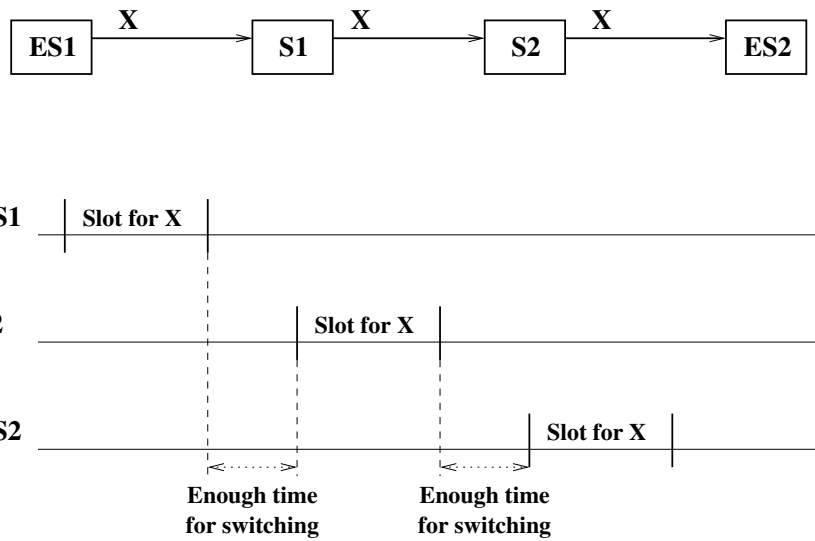
Time-Aware Shapers (TAS)

- Time-Aware Shapers to deal with interfering frames
- Knowledge of the next arrival time for scheduled frames
  - Global synchronization has to be provided
  - Time-sensitive traffic typically follows a regular pattern
- TAS blocks any lower priority frame that would interfere with an upcoming time-sensitive frame
  - Time-sensitive traffic mapped onto highest priority class



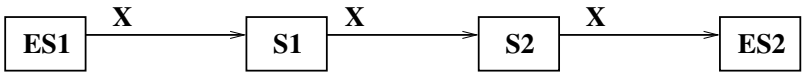
Summary of Time-Triggered switched Ethernet networks

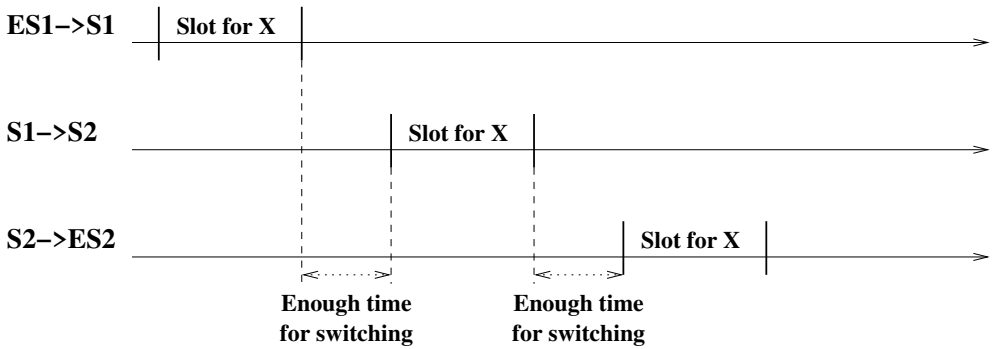
- Dedicated slots for each time-triggered flow for each hop of its path
- Slots have to be carefully organized in order to limit delays



- TAS corresponds to the insertion of a guard band before the slot when no transmission can start
  - Guard band = duration of the longest possible frame $\Rightarrow$ no transmission overlap with the slot
  - Guard band shorter than the longest possible frame $\Rightarrow$ possible overlap
  - Frame preemption can mitigate this problem

Exercise: Impact of guard bands on delays





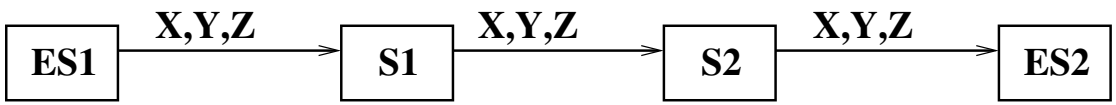
- Transmission of an X frame on a link: 100 μs
- Links are shared with other flows
- Transmission of a frame of any flow: 100 μs
- Switching is never more than 10 μs

Q1: Slot scheduling with guard bands of 100 μs. E2E delays?

Q2: Slot scheduling with guard bands of 0 μs. E2E delays?

Slot allocation strategy

- Several TT flows on a given link ⇒ different strategies
  - Uniform distribution of TT slots
  - grouped TT slots
  - Anything between these two strategies



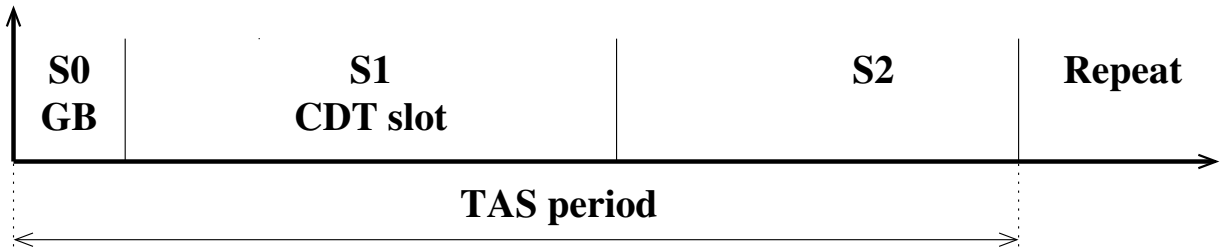
Uniform distribution of slots

Grouped slots

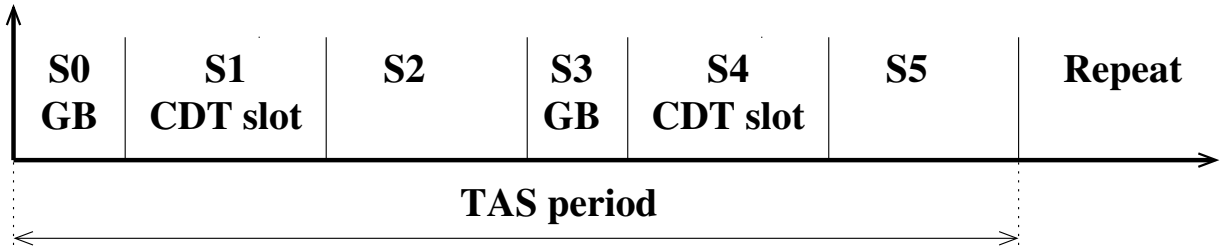
Implementation of TAS

- A set of traffic classes with a queue per class, for example
  - Control Data Traffic (CDT) with dedicated slots
  - Class-A traffic
  - Class-B traffic
  - Best-effort (BE) traffic
- CDT frames should never start transmission, except during the dedicated slots
- A, B and BE frames should never start transmission during a dedicated slot
- No frame should start transmission during the guard band before a dedicated slot
- A gate is associated to each queue (traffic class)
- Up to eight gates $\Rightarrow$ up to eight traffic classes
- When a gate is closed, no frame from the associated queue can start transmission
- When a gate is opened, frames from the associated queue can be transmitted (depending of the prioriy and available credit)

Examples of TAS configurations



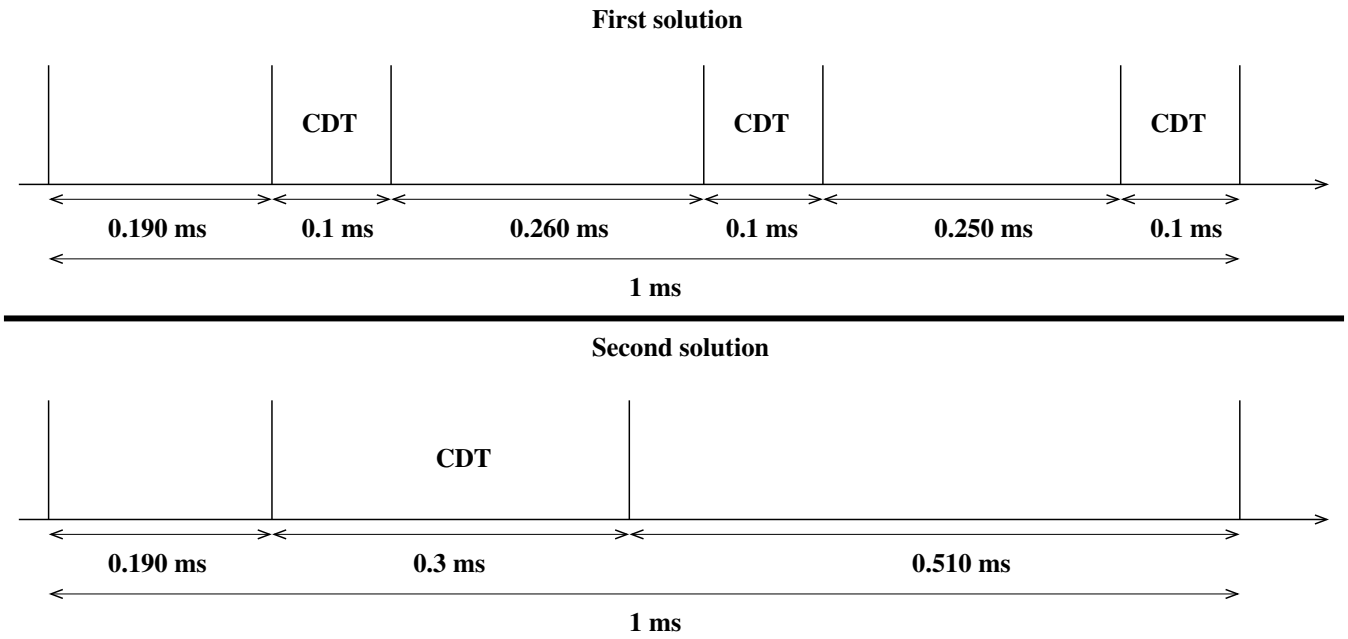
slot	CDT	A	B	BE
S0	0	0	0	0
S1	1	0	0	0
S2	0	1	1	1
Repeat				



slot	CDT	A	B	BE
S0	0	0	0	0
S1	1	0	0	0
S2	0	1	1	1
S3	0	0	0	0
S4	1	0	0	0
S5	0	1	1	1
Repeat				

Exercise

- 3 CDT flows X, Y, Z, period = 1 ms, T_x = 100 μs for 1frame
- Two possible CDT slot allocations, guard bands of 100 μs



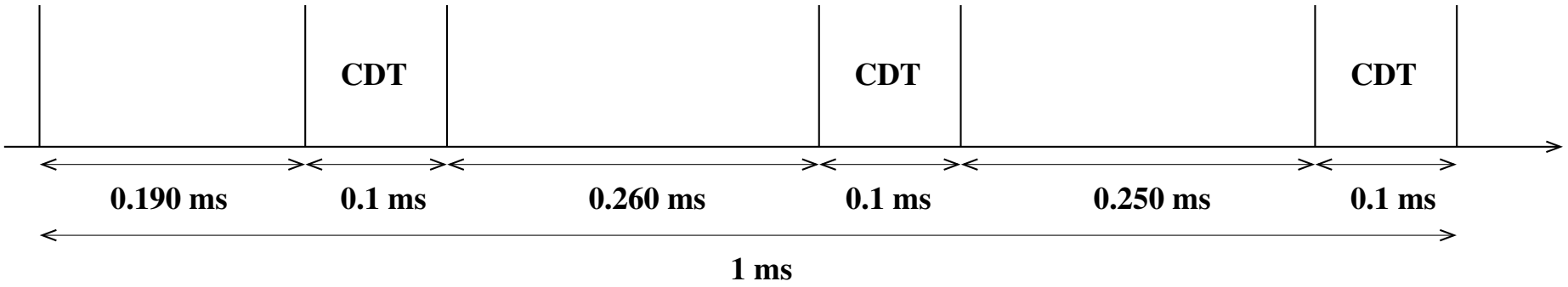
- Give the TAS scheduling table for each solution
- Give the response time of 10 non CDT frames ready at t = 0.1 ms, assuming no priorities between them and no CBS

CBS and TAS

- When its gate is closed because of a CDT slot, a class cannot transmit
- If this class has a CBS, how does this CBS evolves?
- Different rules can be applied, typically
  - Solution 1: the credit is frozen during the guard band and the CDT slot
  - Solution 2: the credit is frozen during the CDT slot, but it can increase during the guard band
- Solution 1 is often considered by people working on this topic
- Solution 2 is what is written in the standart
- Idle slope should take into account durations when credit is frozen

Exercise

- Periodic CDT slot allocation

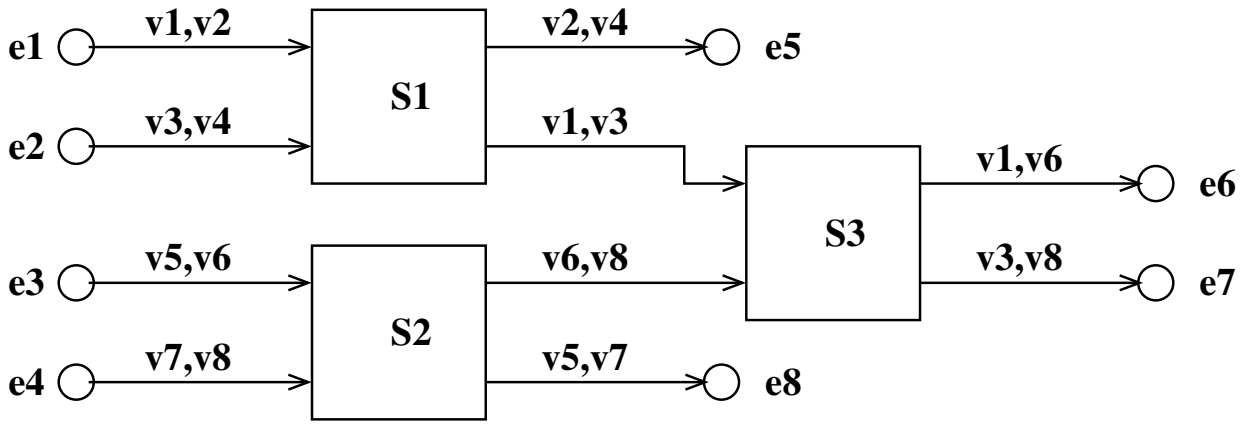


- 5 non CDT A frames ready at t = 0.1 ms, Tx = 0.1 ms for each frame
- 5 non CDT BE frames ready at t = 0.1 ms, Tx = 0.1 ms for each frame
- CBS for class A
  - send slope: -2 every μs
  - idle slope: +3 every μs
- Sequence of frame transmissions for both solutions of previous slide

TTEthernet very short summary

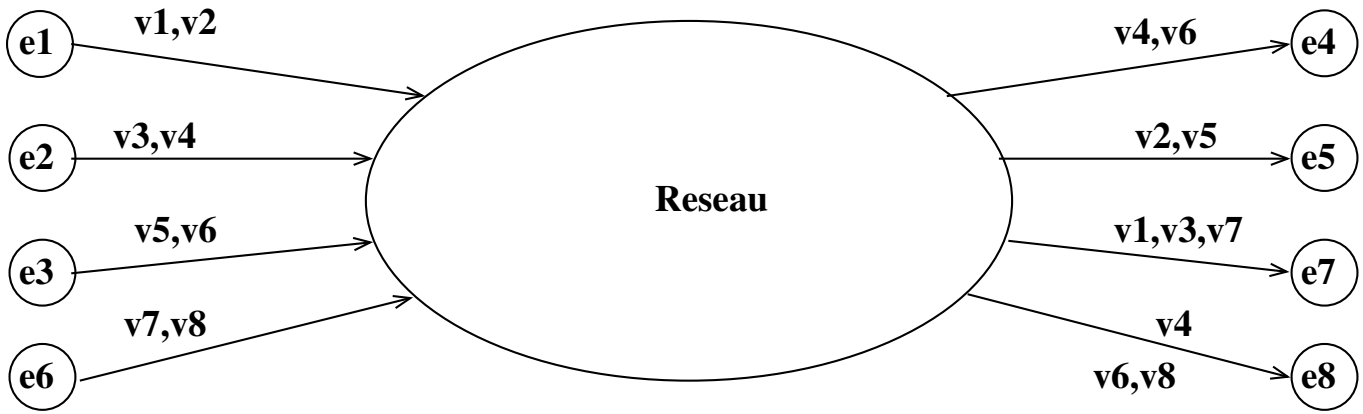
- Time is shared between time-triggered and event-triggered traffic
- Three types of traffic
  - TT flows: scheduled flows, possible thanks to the clock synchronization
  - Rate Constrained flows (RC): very similar to AFDX VLs
  - Best Effort flows (BE)
- Priority order: TT, then RC, then BE
- RC and BE frames might interfere with TT frames
- Three solutions when a RC or BE frame is ready before a TT slot
  - RC or BE frame starts transmission and might delay TT frame
  - RC or BE frame starts transmission and is preempted if TT frame is ready
  - RC or BE frame starts transmission only if there is enough time before TT slot

Event-triggered Vs Time-triggered



- Period for all the flows: 1 ms
- Transmission time for any frame: 50 μs
- Switching latency is assumed to be 0
- Question 1: Worst-case delay for each flow, assuming a basic FIFO strategy
- Question 2: Worst-case delay for each flow, assuming a time-triggered strategy ($\Rightarrow$ a scheduling of the flows has to be built)

Ethernet-DCR Vs Powerlink



- Flow features

	Tx (μs)	Period = deadline (μs)
v1	100	600
v2	100	600
v3	100	600
v4	100	600
v5	100	2400
v6	100	2400
v7	100	2400
v8	100	2400

Ethernet-DCR Vs Powerlink

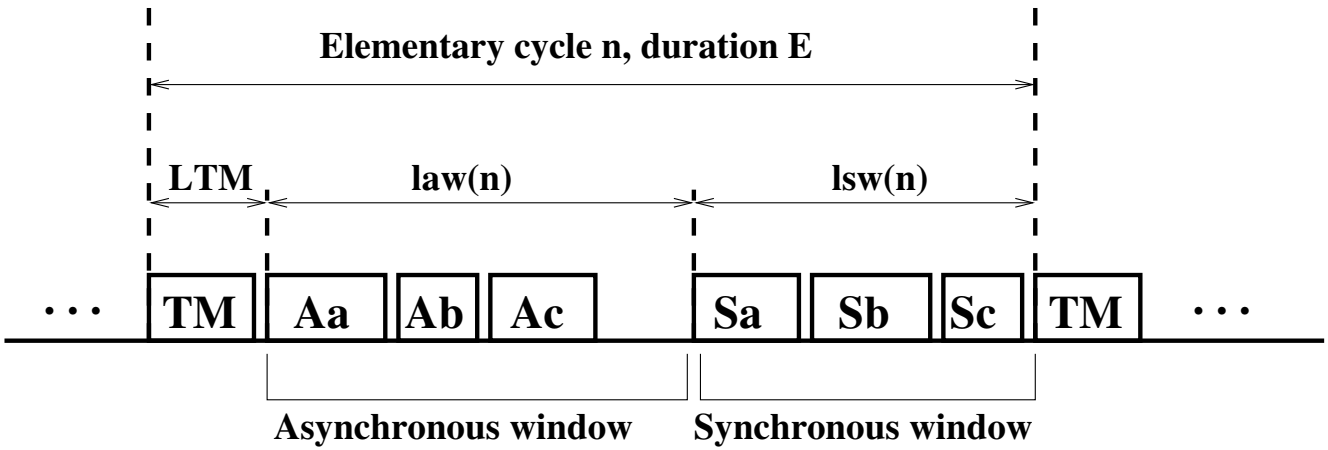
- Do all the flows respect their deadline in the three following situations?
  1. An Ethernet-DCR with the first frame from each flow ready at $t = 0$,
  2. An Ethernet-DCR with the first frame from flows v1, v2, v3, v4, v5 and v7 ready at $t = 0$ and the first frame from flows v6 and v8 ready at $t = 1200$ μs,
  3. An Ethernet Powerlink with the first frame from each flow ready at $t = 0$.
- For Ethernet Powerlink, we assume that Start-of-Cycle messages as well as Poll-Request messages each take 10 μs.

Flexible Time-Triggered Ethernet (FTT-Ethernet)

- Main motivations for FTT-Ethernet
  - Static resource allocation policies might be inefficient
    - ⋆ Operational mode changes, e.g. due to environment stimuli
    - ⋆ Dynamic reconfiguration, according to online requirement updates
    - ⋆ Variable number of requests from other subsystems, e.g. mobile robots operating in dynamic environments
    - ⋆ The level of resources uilised in the system vary dynamically
  - Operational flexibility (dynamic Qos management) is needed
  - Timeliness guarantees are still mandatory $\Rightarrow$ adequate admission control
  - The number of priority levels in Ethernet switches might be too limited
- TTEthernet and TSN are not very flexible
  - TT slots are statically reserved
  - ET part assumes basic service discipline

The FTT principle

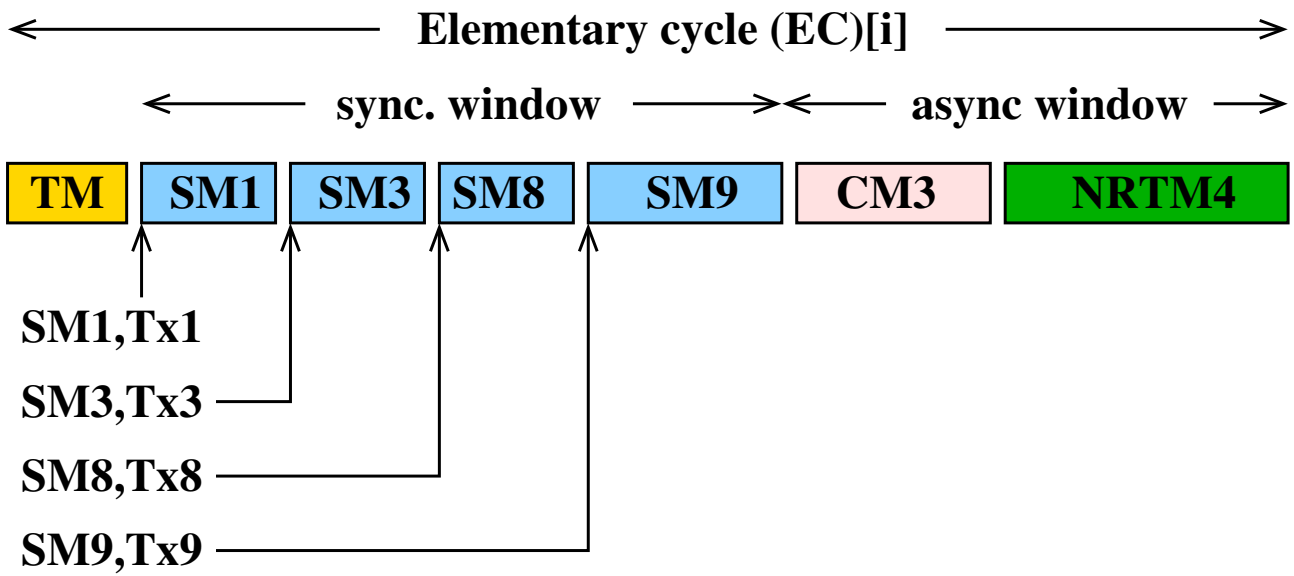
- Master/multi-slave architecture
- An elementary cycle (EC) with a synchronous (time-triggered) and an asynchronous (event-triggered) part
- Dynamic management of the traffic
- First implementation: FTT-CAN



- Application to Ethernet
  - FTT-Ethernet
  - FTT-SE (switched Ethernet)

FTT-Ethernet

- Structure of the elementary cycle



- TM: Trigger message, synchronisation and synchronous messages to be transmitted in the synchronous window with the transmission instants
- The asynchronous window includes two types of traffic
  - asynchronous real-time traffic
  - non real-time traffic (if time is available)

System Requirements Database (SRDB)

- Properties of the message streams to be conveyed
- Synchronous Requirements Table (SRT): N_S synchronous messages

$$SRT = \{SM_i(C_i, Ph_i, P_i, D_i, Pr_i, *Xf_i),\ i = 1 \ldots N_S\}$$

  - C_i: maximum transmission time
  - Ph_i: initial offset, integer multiple of the EC duration
  - P_i: period, integer multiple of the EC duration
  - D_i: deadline, integer multiple of the EC duration
  - Pr_i: fixed priority defined by the application
  - Xf_i: custom structure for QoS specific parameters
- Asynchronous Requirements Table (ART): N_A asynchronous messages

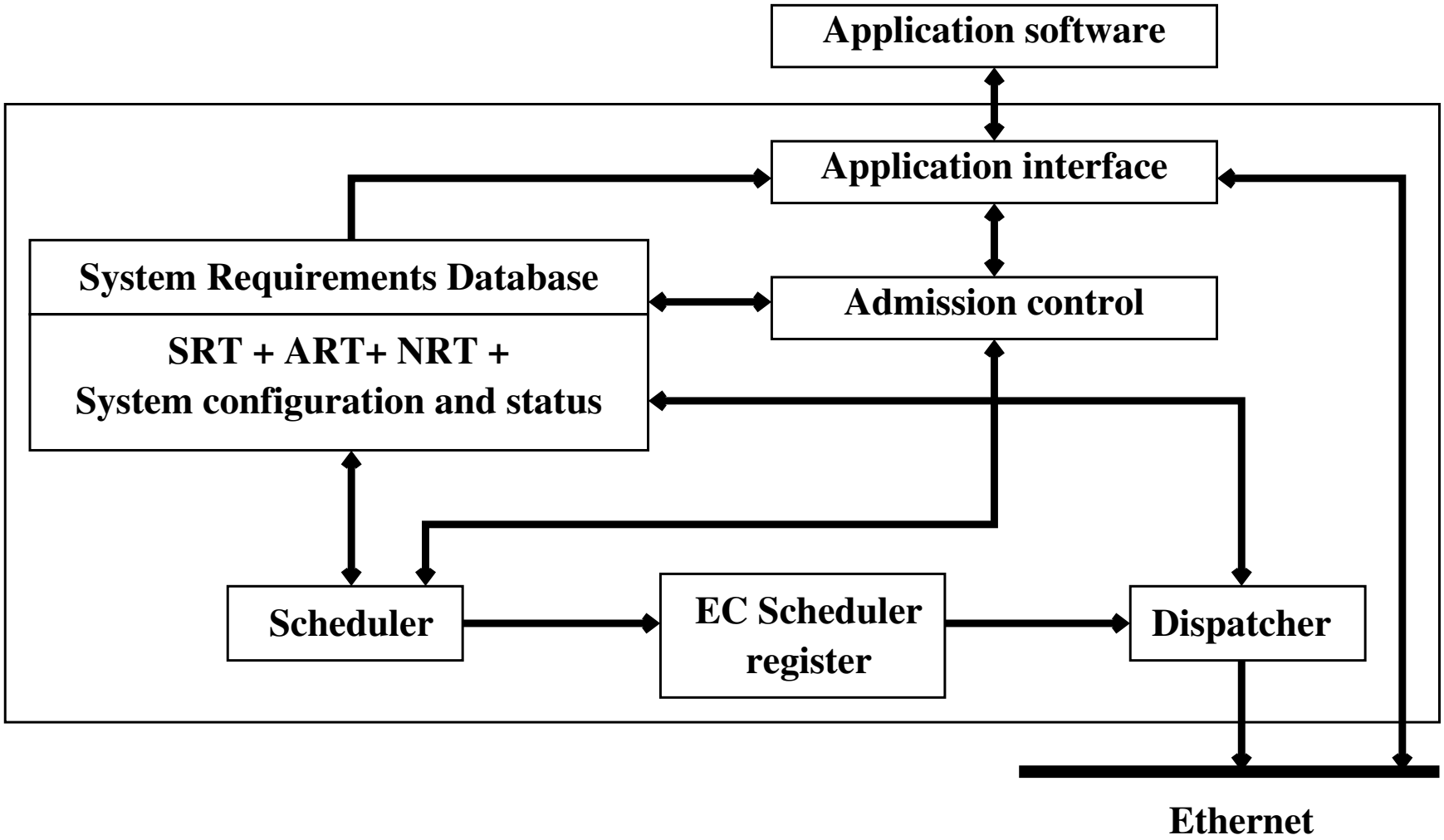
$$ART = \{SM_i(C_i, mit_i, D_i, Pr_i),\ i = 1 \ldots N_A\}$$

  - mit_i (minimum interarrival time) replaces period, no initial offset
- Non-Real-Time Requirements Table (NRT): One line per station

$$NRT = \{NM_i(SID_i, MAX\_C_i, Pr_i),\ i = 1 \ldots N_N\}$$

  - SID_i: node identifier
  - MAX_C_i: transmission time of the longest nrt message frome that node
  - Pr_i: node non-real-time priority

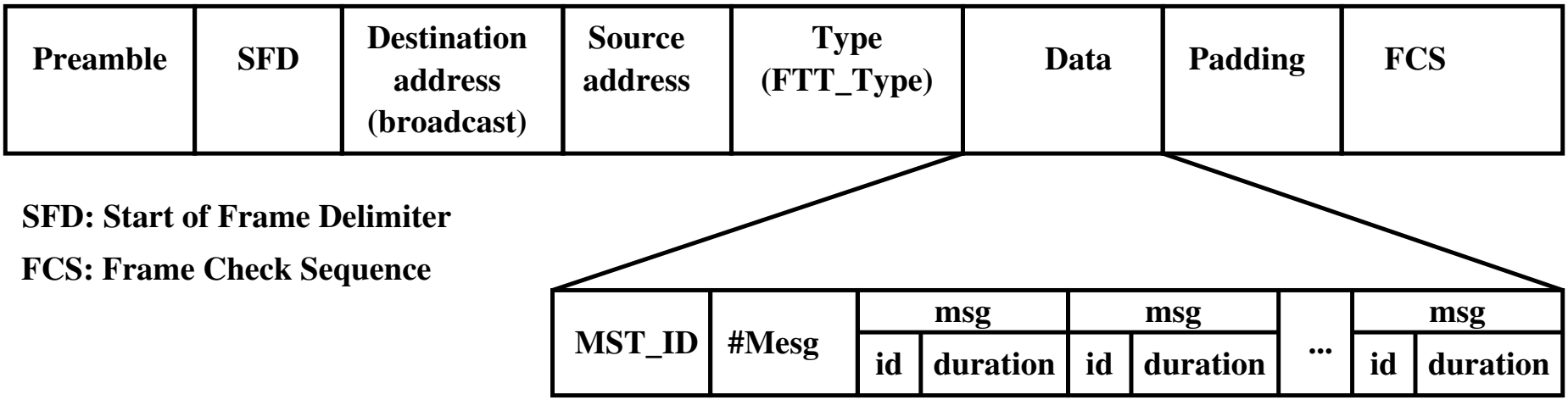
Master internal architecture



Master internal architecture

- Application interface
  - Set of services to the application software
    - ★ SRDP management : add, modify, delete entries in the SRT, ART and NRT
- Scheduler
  - Build the EC schedules, based on the information in the SRDB
- Admission control
  - Schedulability test over the synchronous traffic whenever a request for changes in the SRT is made
  - Changes admitted upon a positive schedulability result
  - Can be extended to QoS negotiation and adaptation
- Dispatcher
  - Building of the next Trigger Message (TM) with the EC schedule
  - Periodic broadcasting of the TM

Trigger Message



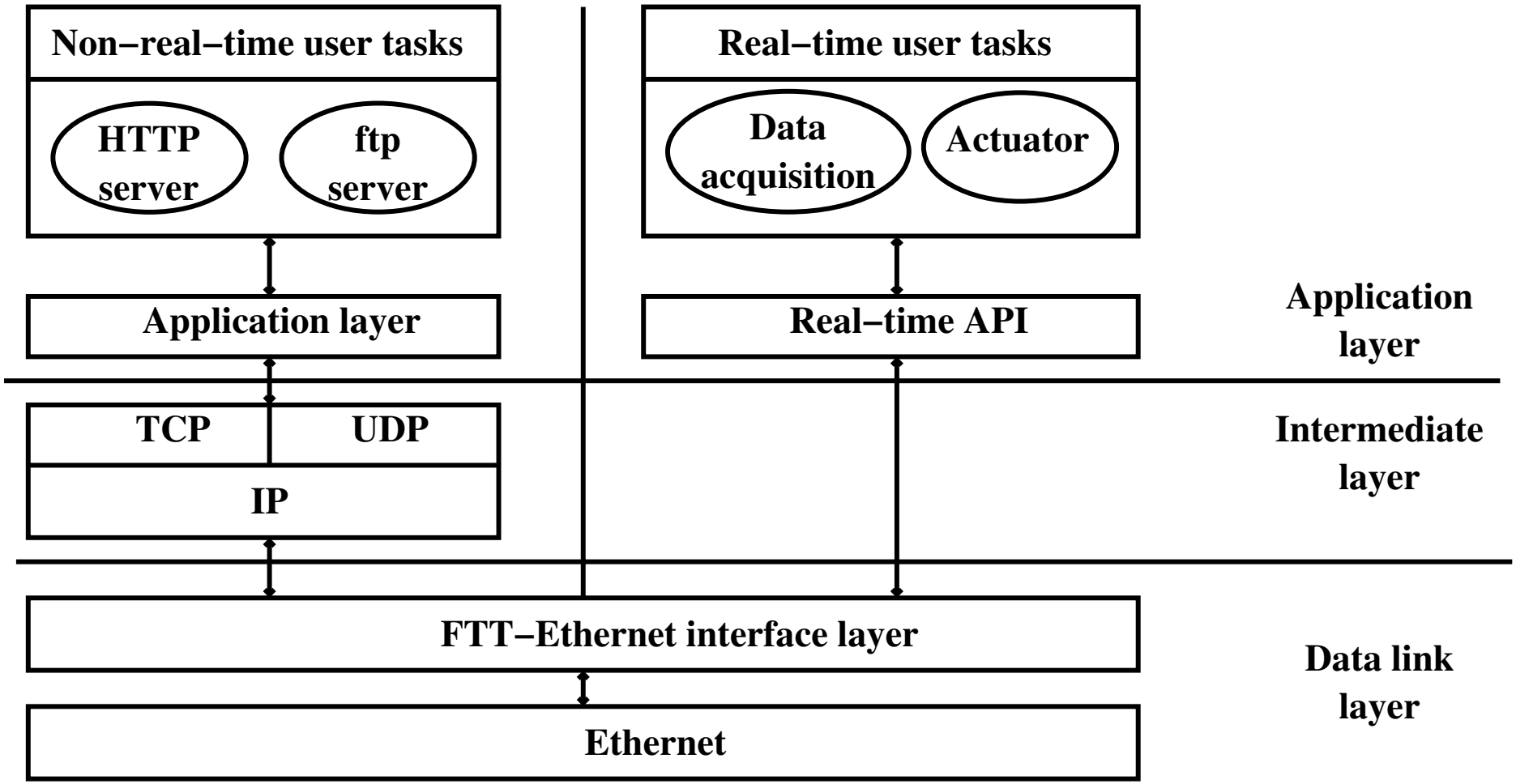
- FTT_Type: all the ftt messages, except non-real-time ones associated with other protocols
- MST_ID: message type for the Trigger Message

Exercise

- An FTT-Ethernet network with an Elementary cycle of 1 ms
- A synchronous window of 300 μs per Elementary cycle
- Four aperiodic messages
  - M_1: $C_1 = 100$ μs, $Ph_1 = 0$, $P_1 = D_1 = 1$ ms
  - M_2: $C_2 = 100$ μs, $Ph_2 = 0$, $P_2 = D_2 = 2$ ms
  - M_3: $C_3 = 100$ μs, $Ph_3 = 0$, $P_3 = D_3 = 2$ ms
  - M_4: $C_4 = 150$ μs, $Ph_4 = 0$, $P_4 = D_4 = 4$ ms
- Scheduling policy: Earliest Deadline First

Question: give the data content of Trigger Messages

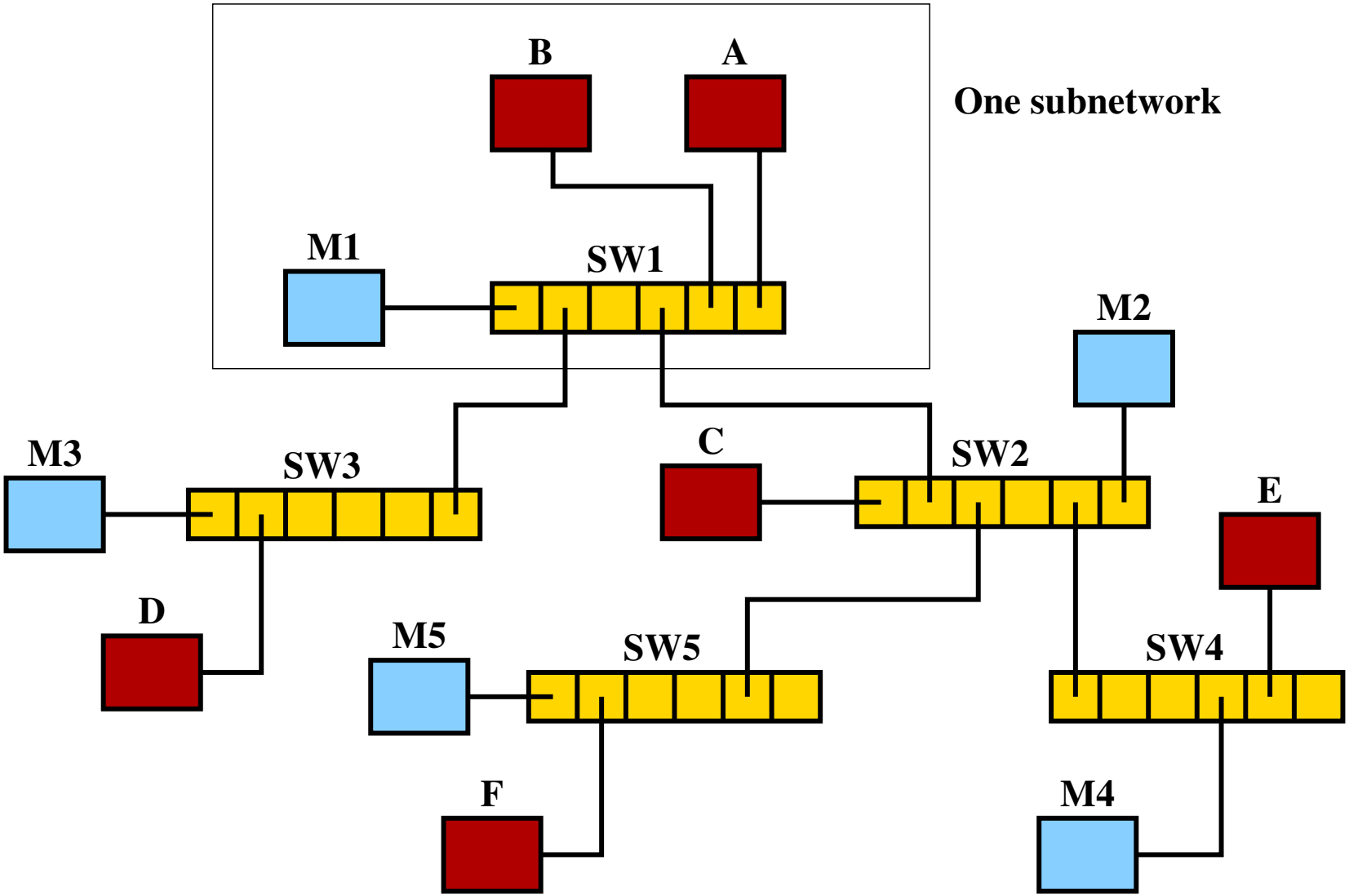
Internal architecture of slave nodes



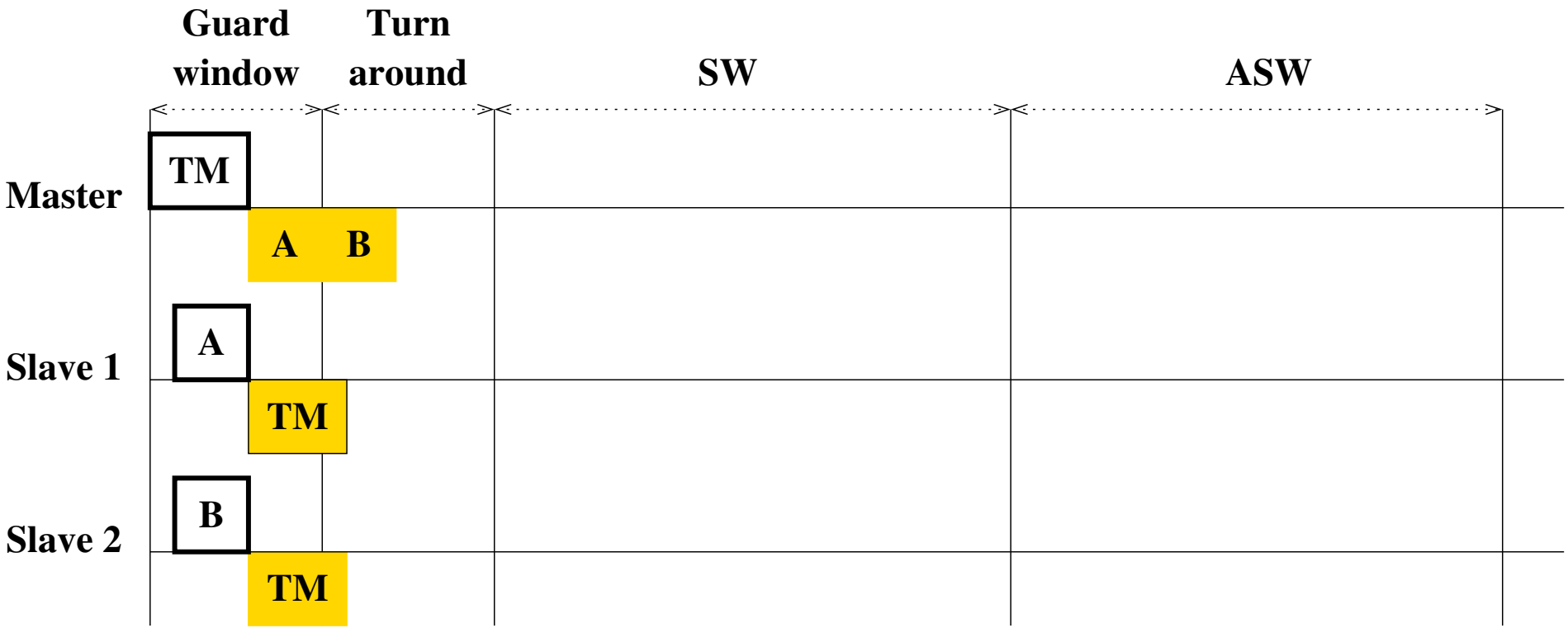
FTT-SE (Switched Ethernet)

- Extension of FTT-Ethernet to a multi-switch architecture
- A tree topoology is assumed
- Different architectures have been considered
- Focus on one possible architecture
- One switch is connected to
  - Other switches
  - A master node
  - One or several slave nodes
- The overall network is divided in subnetworks
  - One subnetwork = a switch with its master node and slave nodes
- Each master manages the scheduling of its subnetwork
- Two kinds of messages
  - Local messages: sender and receiver are in the same subnetwork
  - Global messages: sender and receiver are not in the same subnetwork

Example of an FTT-SE network



Traffic transmission within a subnetwork



- Turn Around window: slave nodes decode the Trigger Message
- Asynchronous requests are made during the Guard and Turn Around windows
- Corresponding asynchronous transmissions are scheduled by the master in the next EC

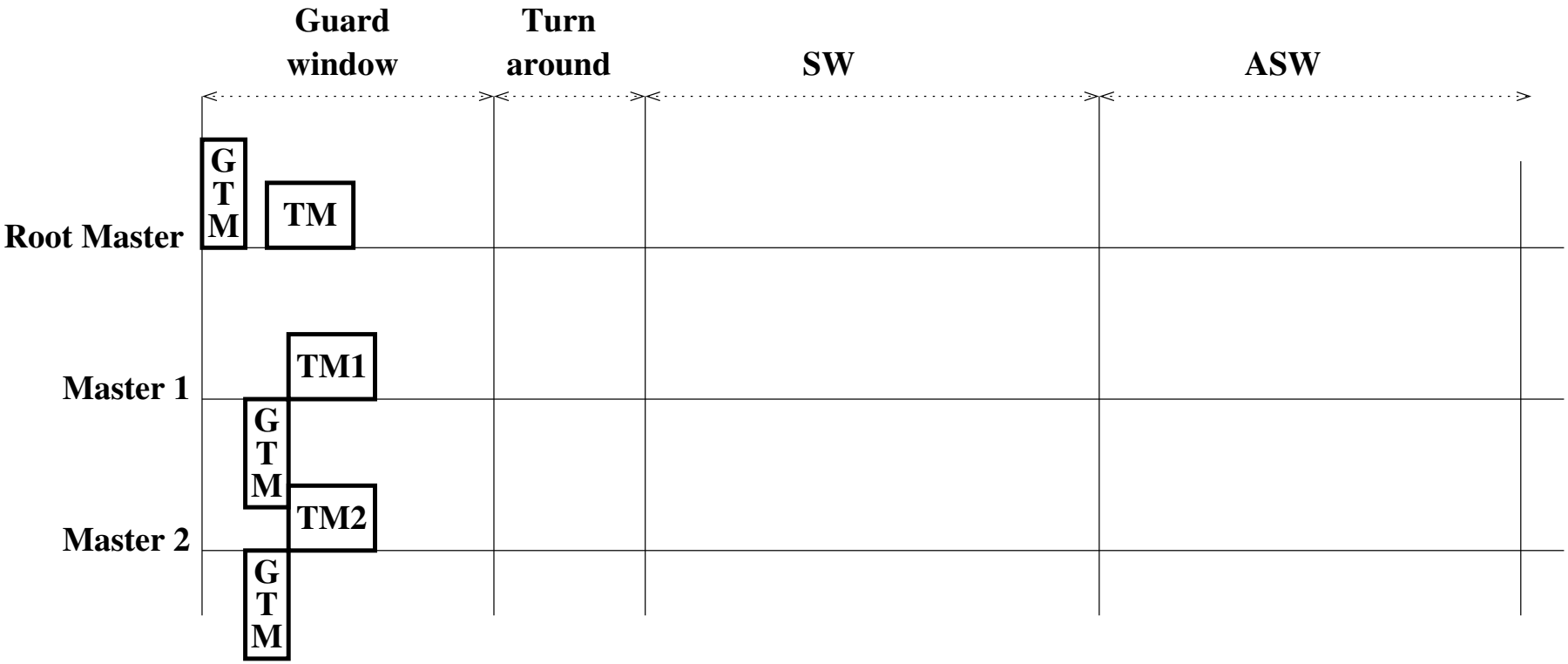
Problems to be addressed

- Confinement of broadcast domain
  - Each master broadcast a Trigger Message
  - Each Trigger Message will generate interferences in all subnetworks
- Time synchronisation
  - All Elementary Cycles for sub-networks have to be timely synchronised $\Rightarrow$ all TM generated from all masters have to be broadcasted at the same time
  - Not respected $\Rightarrow$ might lead to overrun for a global message
- Scheduling synchronisation
  - Scheduling of global messages has to be done consistently by all masters
    - ★ Assume a global message crossing two switches
    - ★ The master of the first switch schedules the message in an EC
    - ★ How can it be sure that there is bandwidth for this message in the same EC in the second switch?

Confinement of broadcast domains

- Usage of multicast groups
- One master = one multicast address including only all its local slave nodes

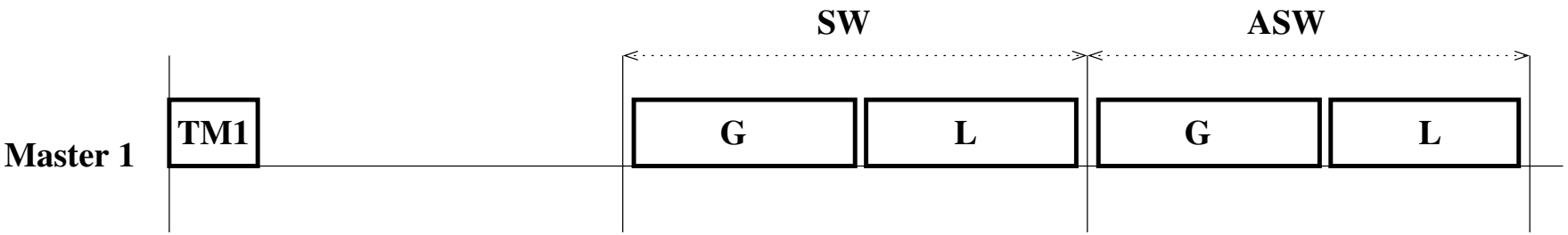
Time synchronisation



- A Global Trigger Message broadcasted by a root master to synchronise all the masters
- A master sends its TM right after receiving the GTM
- The delay depends on the number of hops, acceptable since GTM is of minimum size

Message scheduling

- Each window (synchronous, asynchronous) is divided in two parts
  - One part for the transmission of global messages
  - One part for the transmission of local messages



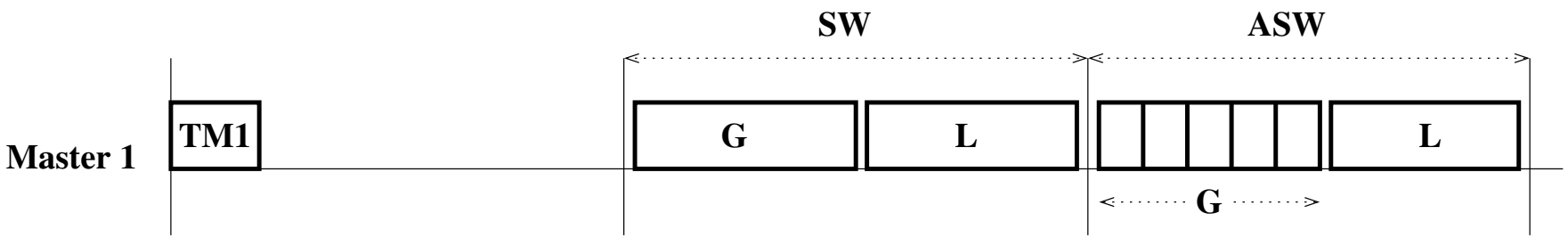
- Each message transmited during a local part is confined within a subnetwork
  - $\Rightarrow$ The scheduling can be managed independently by each master
- Each message transmited during a global part concerns at least two subnetworks
  - $\Rightarrow$ The scheduling has to be managed globally
  - The problem is slightlu different for periodic and aperiodic traffic

Global periodic message scheduling

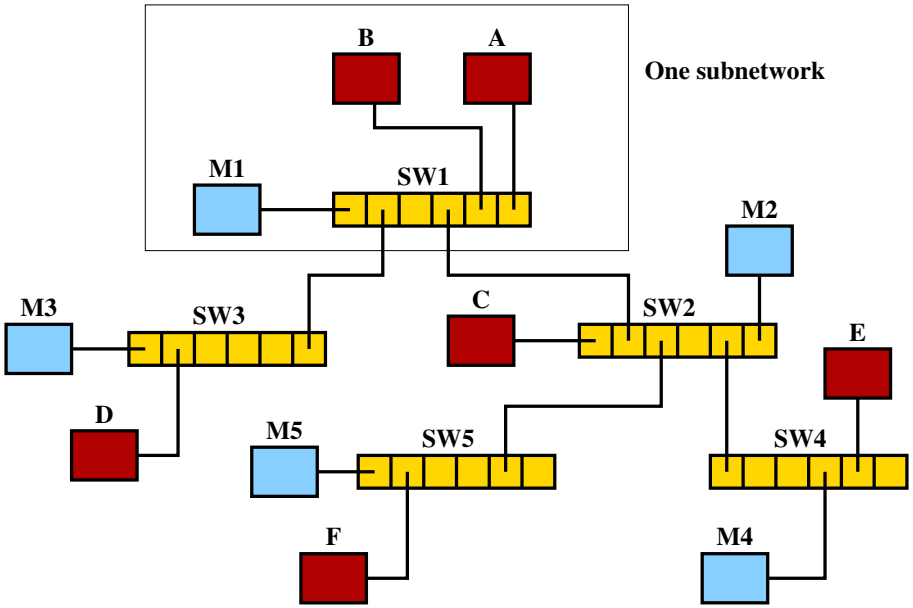
- Same System Requirements Database (SRDB) in all the masters $\Rightarrow$ all the features of all the periodic global messages are known by all the masters
- Generation times of periodic global messages are known statically
- All the masters build a global scheduling for each EC, using the same policy $\Rightarrow$ they get the same result with no overrun
- Each master broadcasts in its TM the part that concerns its subnetwork

Global aperiodic message scheduling

- Masters also share features of aperiodic global messages
- But generation times of aperiodic global messages are not known statically
- They are known dynamically
- Source node of a given aperiodic global message sends a request to its master at the beginning of the EC
- Other masters don't have the information $\Rightarrow$ all the masters cannot build the same scheduling for aperiodic global messages
- One possible solution: one subpart for each subnetwork in the globlal part of the asynchronous window



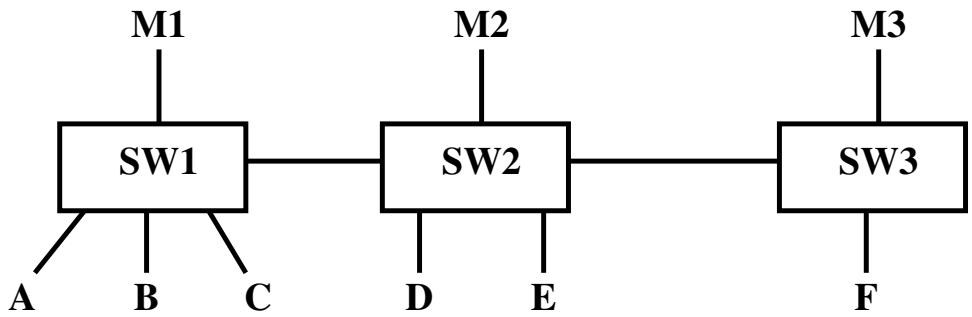
Exercise



- Six periodic global messages M_{AE}, M_{BD}, M_{AD}, M_{BC}, M_{BF}, M_{AF} by decreasing priorities
- Period of all the messages: 5 cycles
- Transmission of one frame on one link: 10 μs
- switching latency: 0
- Global part of synchronous window: 50 μs

Question: How many EC to transmit one frame of all the message when they are all ready at the same time?

Exercise



- Four periodic global messages M_{CD}, M_{EF}, M_{BF}, M_{AD}
- Priority order: $M_{CD} > M_{EF} > M_{BF} > M_{AD}$
- The transmission of 1 message occupies the whole global part of the synchronous window

1. What is the delay (number of EC) for M_{AD} when all the messages are ready at the same time?
2. Same question when M_{CD} and M_{AD} are ready one EC before M_{EF} and M_{BF}?