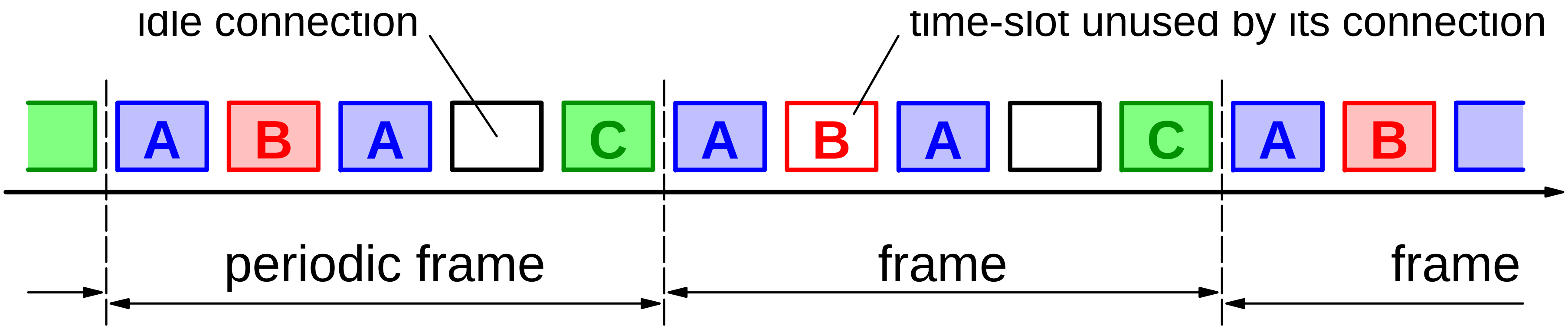
2.2 Circuit Switching, Time-Division Multiplexing (TDM), Time Switching, Cut-through

- Circuit Switching versus Packet Switching
- Digital Telephony, Time-Division Multiplexing (TDM)
- Time Switching, Time-Slot Interchange (TSI)
- Switching and Computers: 1st and 2nd Generations
- Cut-through

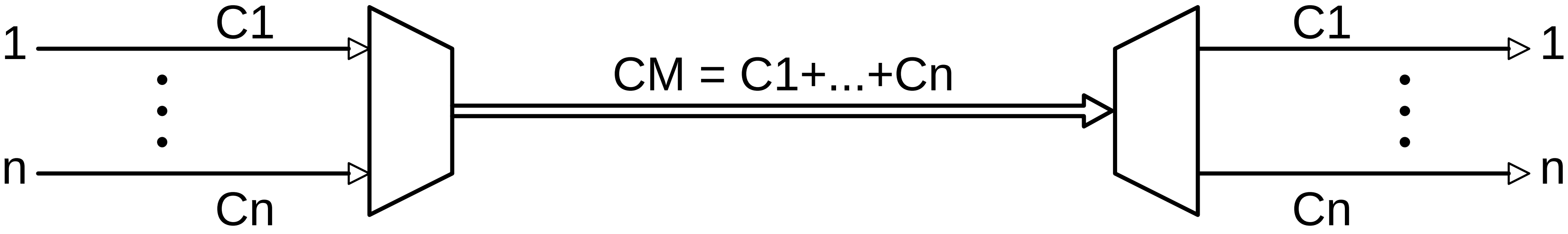
Circuit Switching



- Data in fixed, periodic frames
- Each circuit (connection) is allocated a fixed subset of time-slots
- timeSlotID implicitely provides connectionID and routing information
- Advantage: simplicity – contention, routing, scheduling resolved once, at admission/connection-setup time, rather than separately for each datum (similar to compile-time versus run-time)
- Disadvantage: unused capacity in one connection cannot be used by other connections – “partitioned capacity”: wasteful in transmission capacity when connection rate varies widely over time

Multiplexing - Demultiplexing

at fixed aggregate capacity (circuit-switching style)

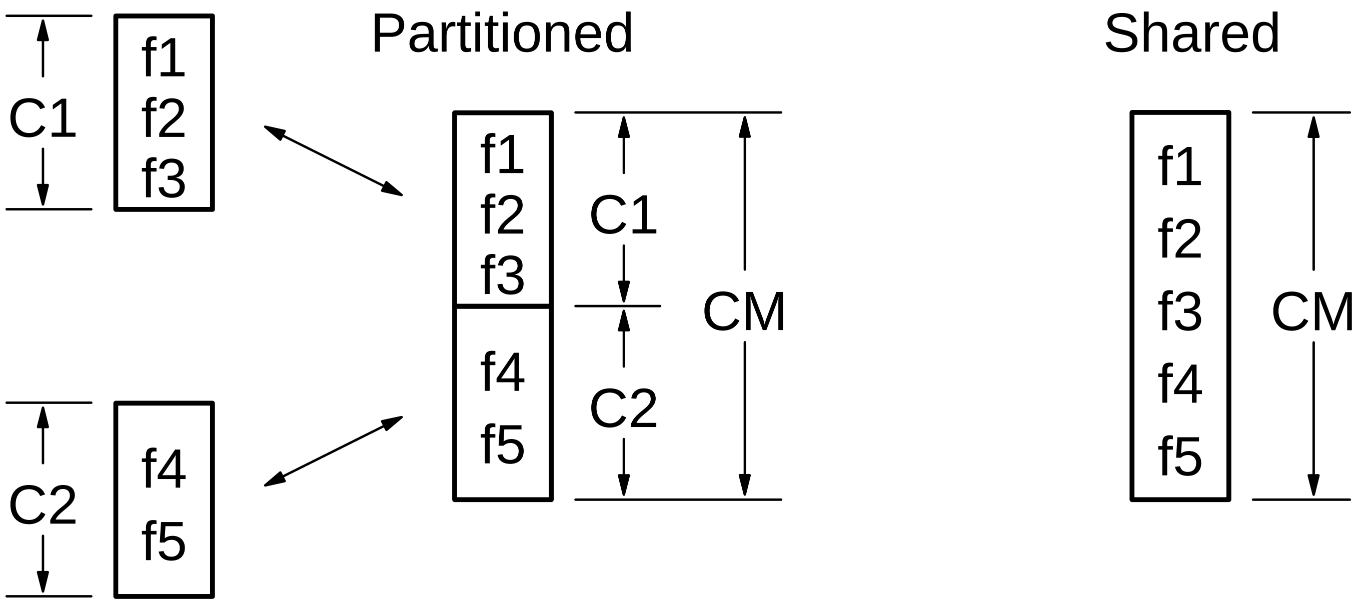


Examples:
- circuit switching: frames & time-slots
- wide (bit-parallel) buses inside switch elements

Minimal buffering requirements:

one time-slot-worth of data per mux'ed/demux'ed link

Partitioned versus Shared Link Capacity

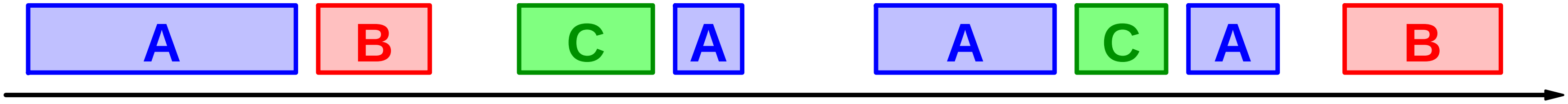


Resource Partitioning leads to Underutilization:

In a link carrying multiplexed traffic of fixed aggregate capacity type, the flows in one partition may lack capacity, while other partitions may have excess capacity.

This is the disadvantage of circuit switching.

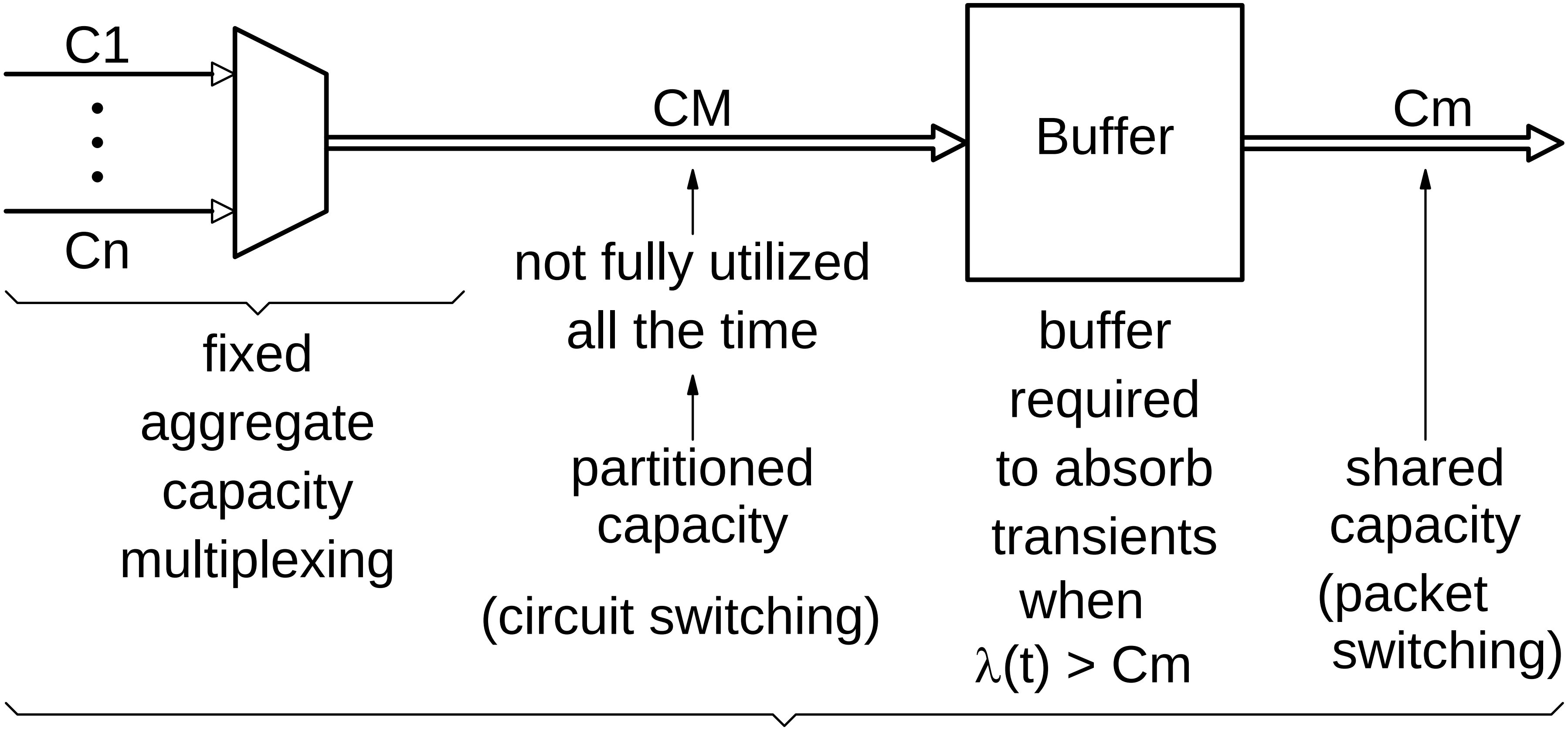
Packet Switching



- Non-periodic multiplexing of packets, on a demand basis; each packet carries its own source and destination (connection) ID, and can be stored and forwarded at any later time.
- The transmission capacity of a link is shared among all flows (connections) that pass through it, on a demand basis; any capacity that is not used by one flow can be used by another.
- Advantage: no waste of transmission capacity.
- Challenges:
  dynamic control (per packet), rather than static (at conn. set-up);
  unpredictability of traffic, leading to contention for resources.

Packet Switching: Statistical Multiplexing

$C1 + ... + Cn = CM > Cm$



C1

Cn

CM

Buffer

Cm

fixed aggregate capacity multiplexing

not fully utilized all the time

partitioned capacity

(circuit switching)

buffer required to absorb transients when $\lambda(t) > Cm$

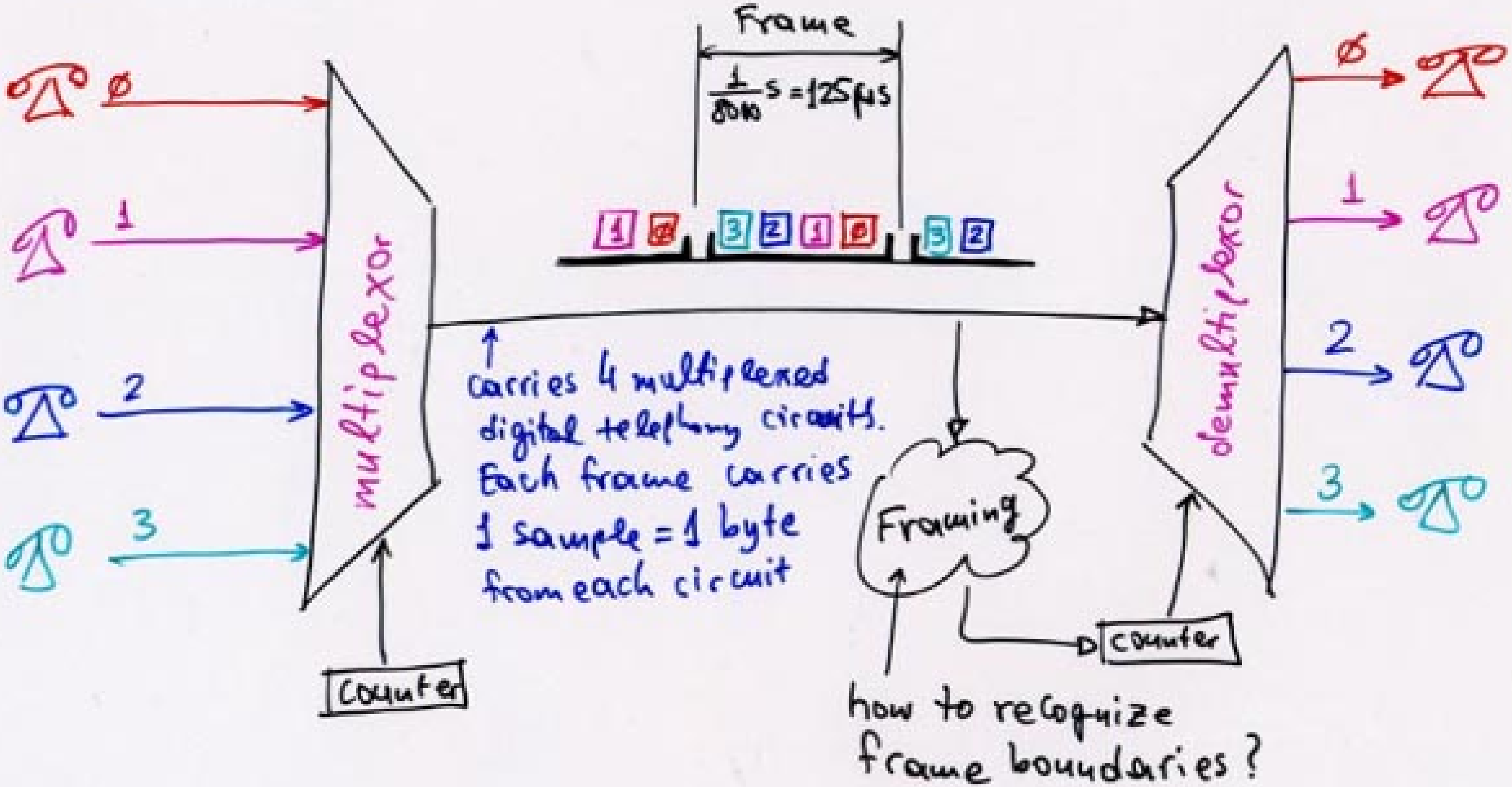
shared capacity (packet switching)

statistical multiplexing

Digital Telephony

telephone quality voice $\lesssim$ 3.5 KHz ⇒ ≈ 8000 $\frac{\text{samples}}{\text{second}}$

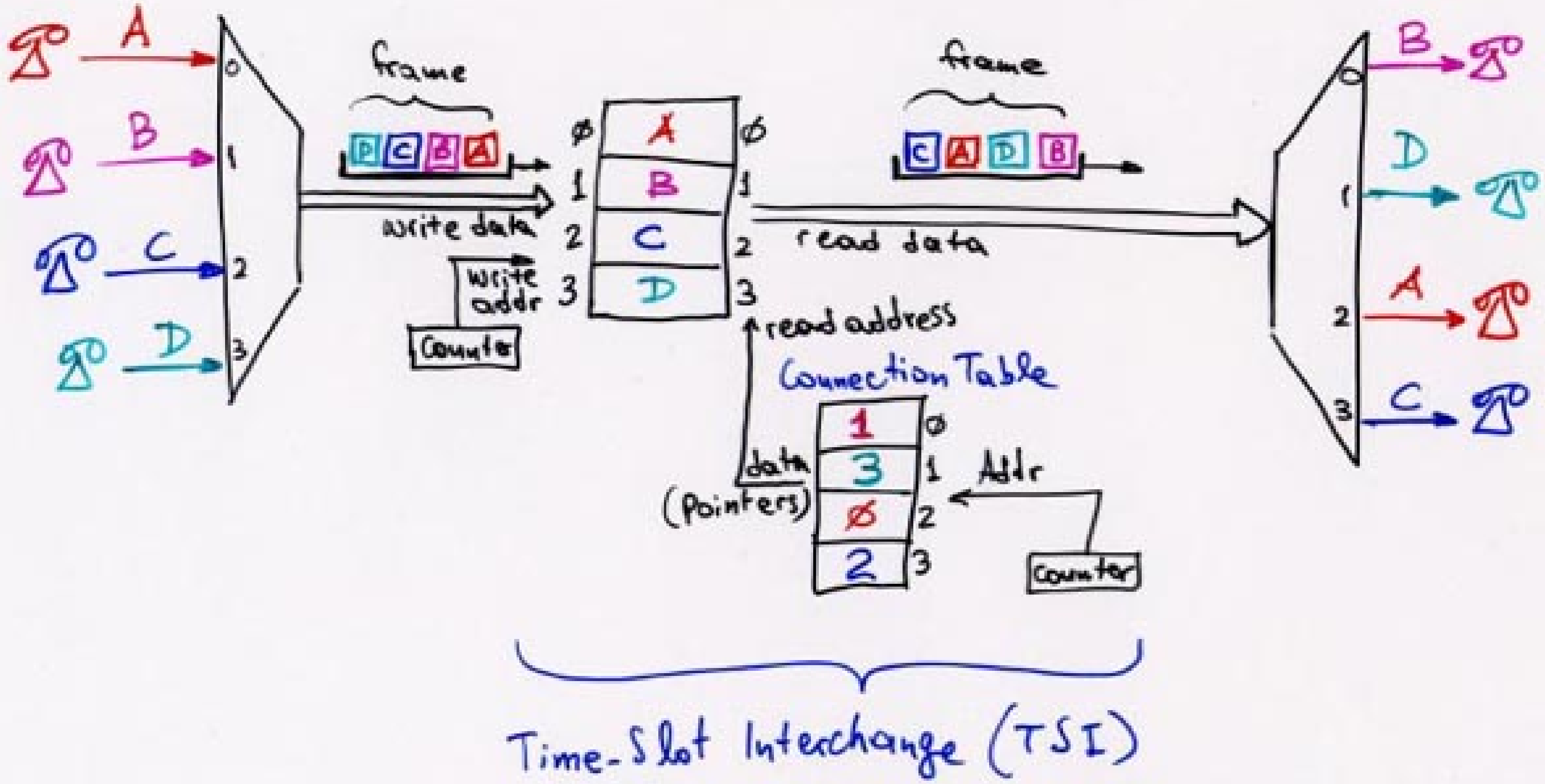
8000 samples/s x 8bits/sample = 64,000 bits/s ... one digital telephony circuit



carries 4 multiplexed digital telephony circuits. Each frame carries 1 sample = 1 byte from each circuit

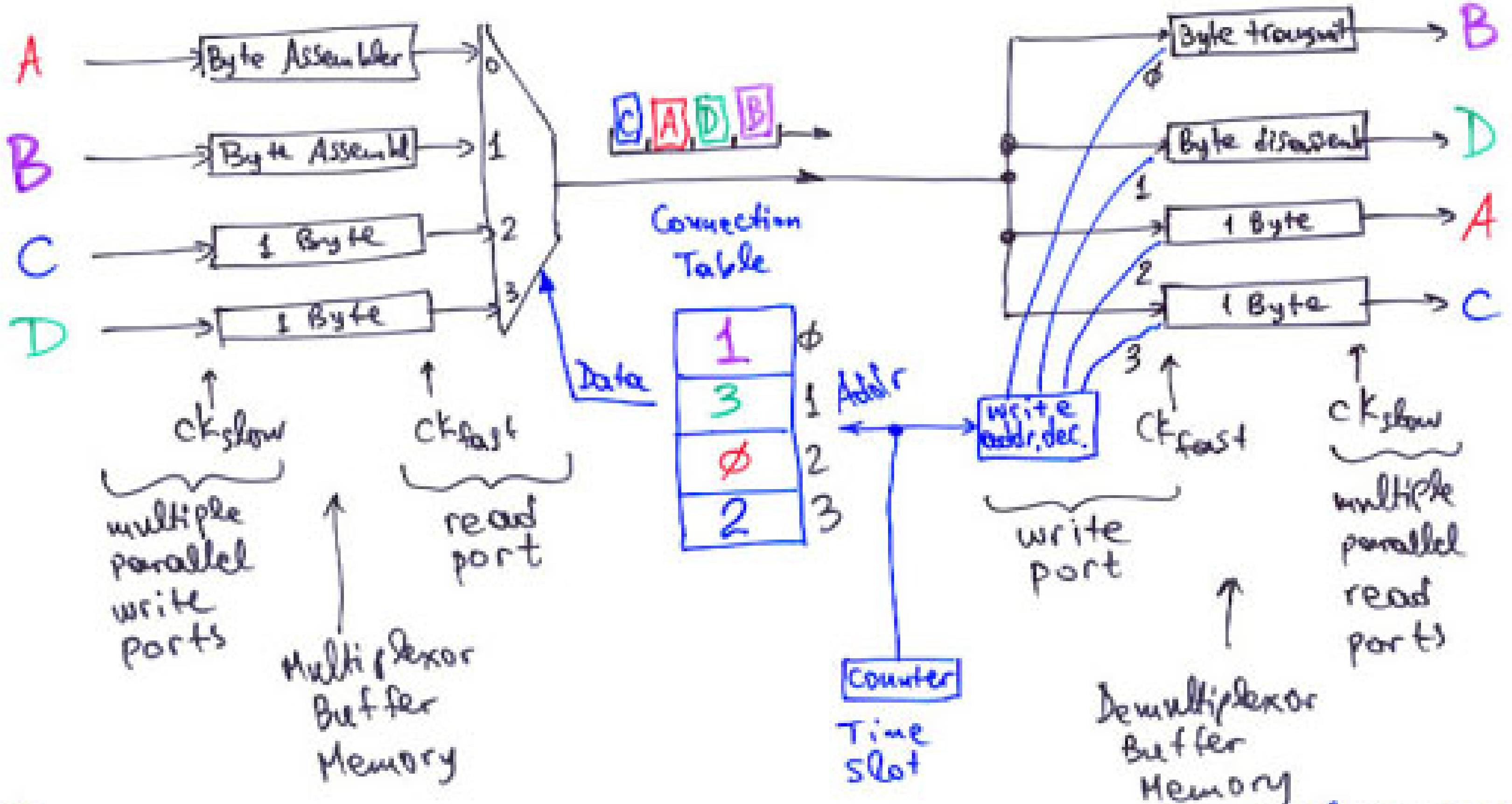
how to recognize frame boundaries?

Time-Division Switching



Time-Division Switching:

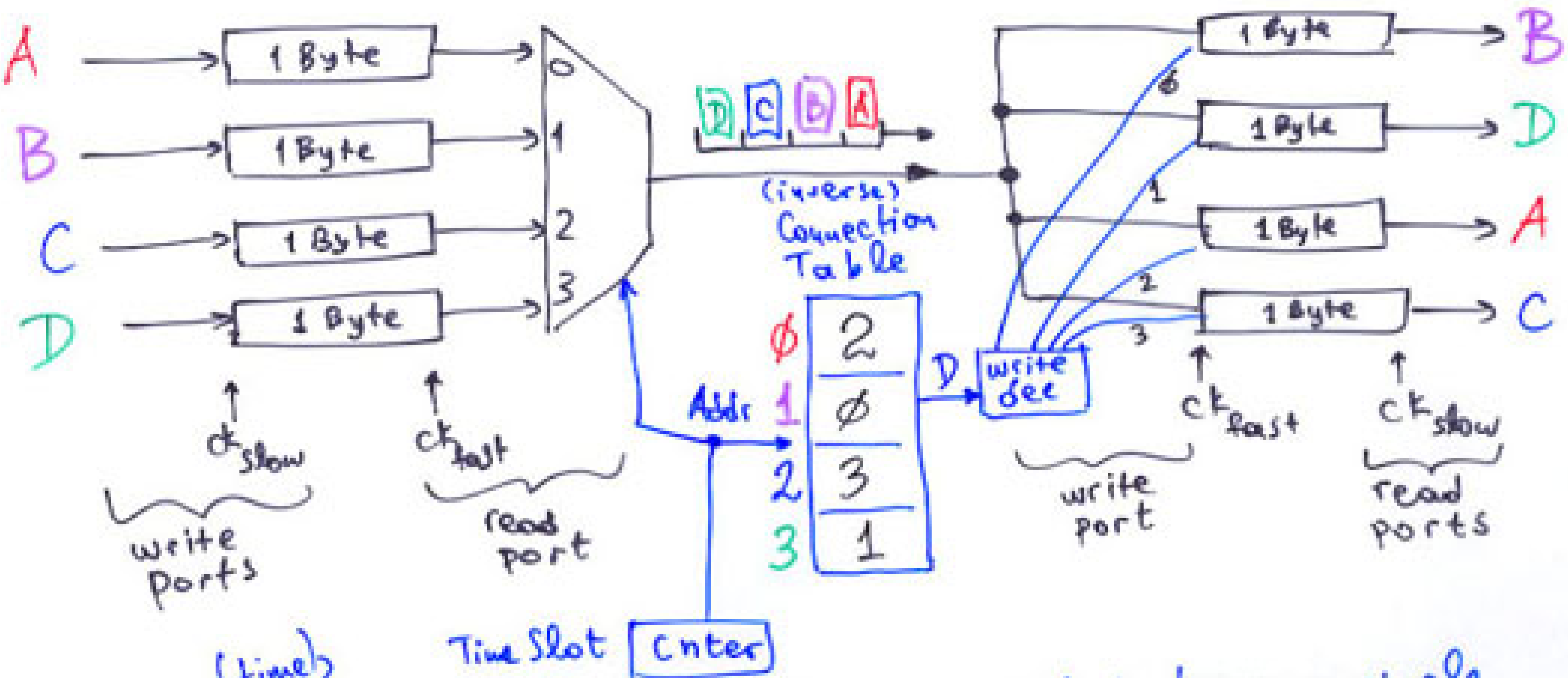
When each external link carries a single connection, the time-slot interchange can be merged with the multiplexor (or with...



Addr	Connection Table
∅	1
1	3
2	∅
3	2

(Could also use TDM at bit granularity, with the connection table running at 8 times higher clock rate)

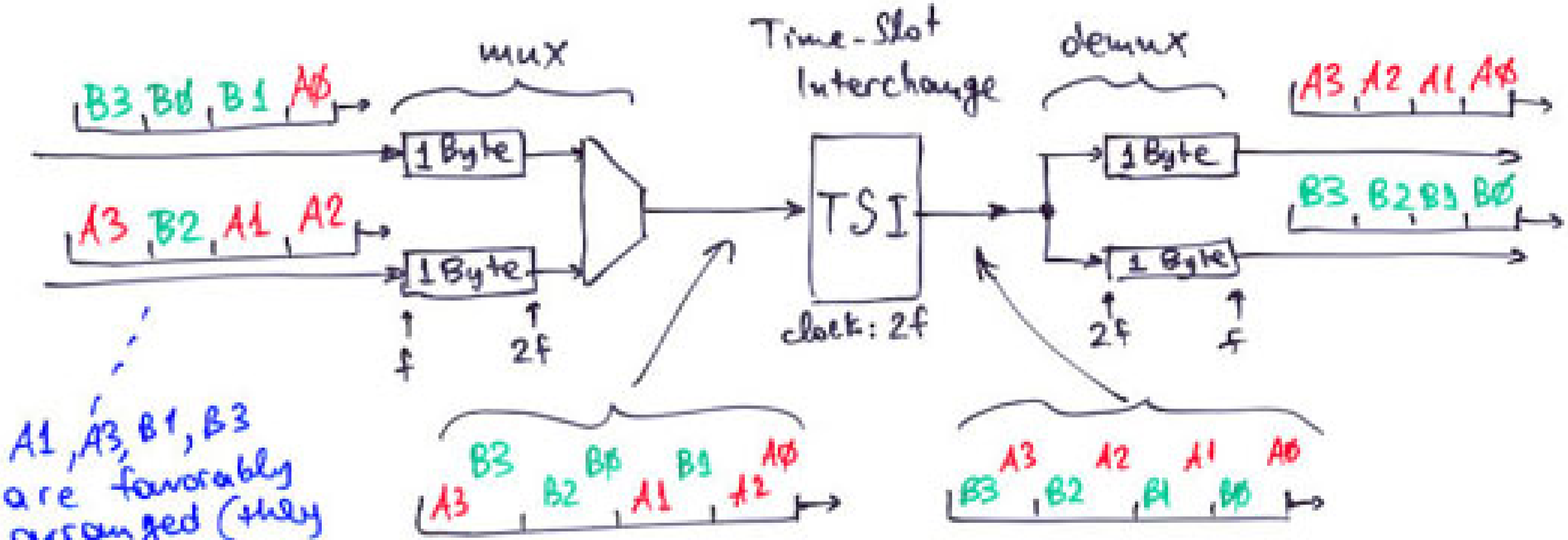
Time-Division Switching with external links carrying a single connection each:
Alternatively, the time-slot interchange can be merged with the demultiplexer



Note: for (time) switching to be possible, it is essential to have a single, shared line going from all inputs to all outputs.
Every single quantum of information goes through that single common line. The throughput of that line $= \Sigma$ inp. thrup. $= \Sigma$ out. thrup.
In this case, $f_{fastclock} = n \cdot f_{slowclock}$ ($n = \#$ ext. lines).

Time-Division Switching: more complex case: multiple connections per external line

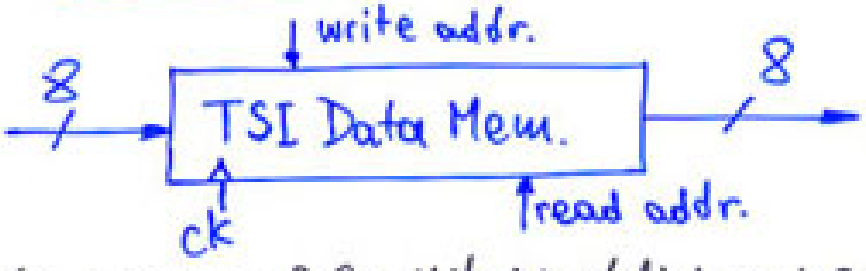
- Mux and Demux need less buffer memory than full frame
- Internal TSI needed with 1 full frame of buf. memory
- Internal TSI cannot be merged w. mux or demux
- worst-case delay = 1 frame time, again.



A1, A3, B1, B3 are favorably arranged (they arrive on correct time slot); however, not so for AØ, A2, BØ, B2, because AØ, BØ arrive on same link ⇒ cannot be placed both in correct time slot.

Discussion: Can I freely rearrange positions of connections inside inp. & output frames?

Byte-by-Byte Time Switching: Throughput Limit ?



Assume 300 MHz 2-port SRAM

⇒ Peak throughput = $300 \frac{\text{MBytes}}{s}$ =

= 2.4 Gbits/s = 37,500 × 64Kb/s

(if making an 8x8 switch ⇒ each link up to 300 Mb/s ... quite low @ today's standards)

Can we Increase Throughput by Widening the TSI Memory?

128 bits = 16 Bytes

128b × 300 MHz = 38.4 Gbits/s

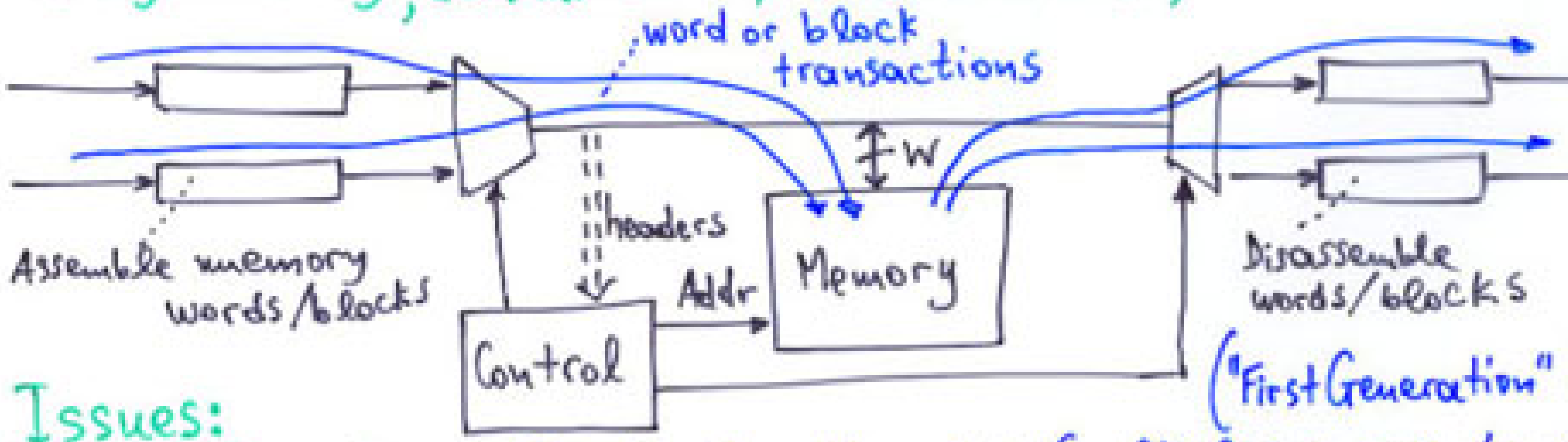
⇝ What is the Multiplexing Quantum???

- 16 Bytes belonging to a same 64Kbps channel ?:
  - → must wait 16 frames = 16 × 125 μs = 2 ms to collect all these bytes!
  - → buffer size for collection = 2 ms × 38.4 Gb/s = 76.8 Mbits
- 16 Bytes belonging to 16 "adjacent" 64 kbps channels ?:
  - → must switch all 16 channels together:
    where one of them goes, all 16 of them must go!

Time Division Switching:

From Circuit Switching to Packet Switching

...from statically, off-line scheduled, fixed-throughput/channel to dynamically, demand-driven, on-line scheduled, variable-throughput/flow



("First Generation" Switches)

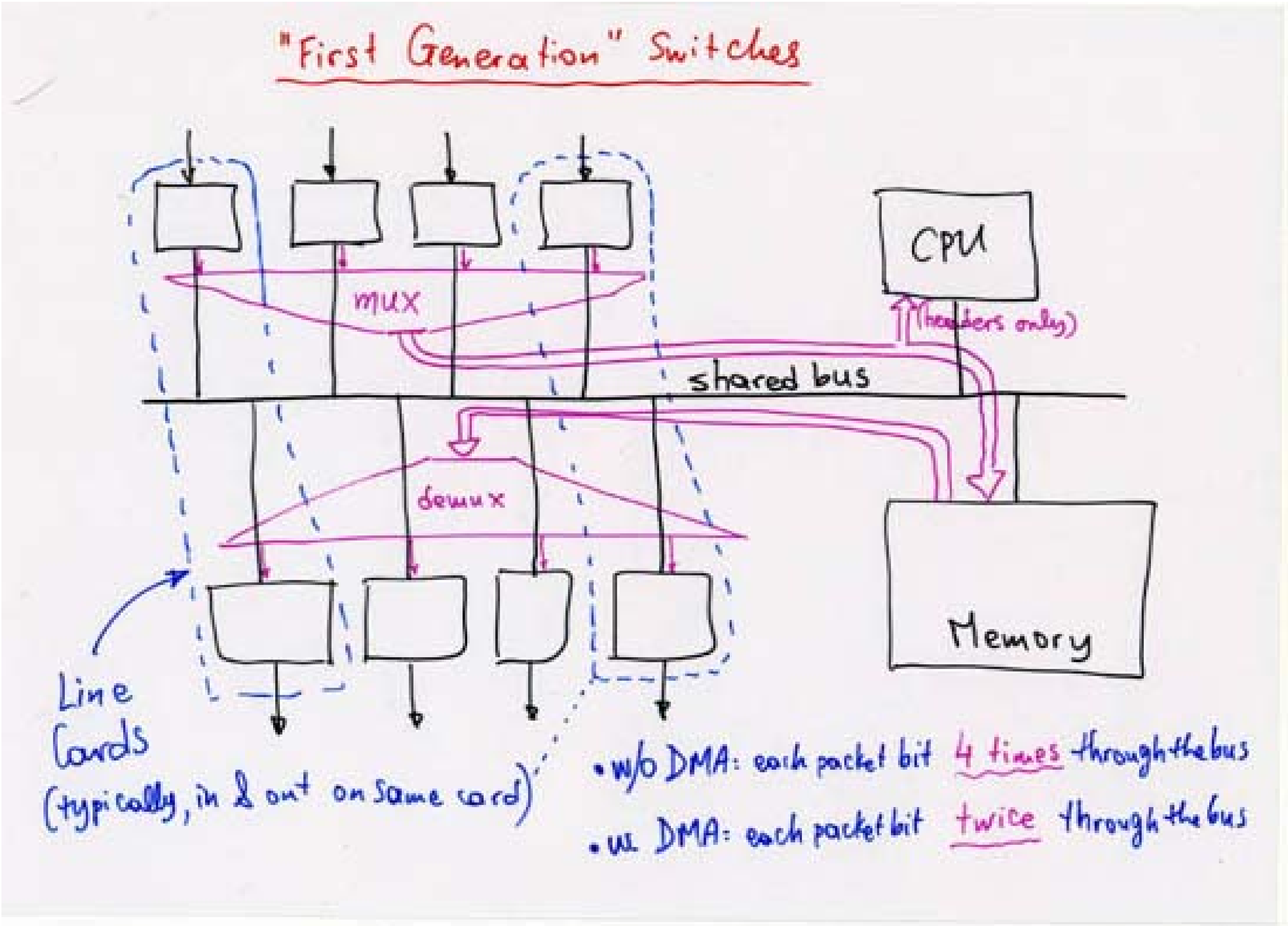
Issues:

- Granularity of transfers In→Mem, Mem→Out (multiplexing quantum):

  fine grain... (narrow word/block): ⊕ small buffers in I/O ⊖ narrow mem ⇒ small throughput
  ⊕ small packets OK ! ⊖ access rate/bus turn-around bound

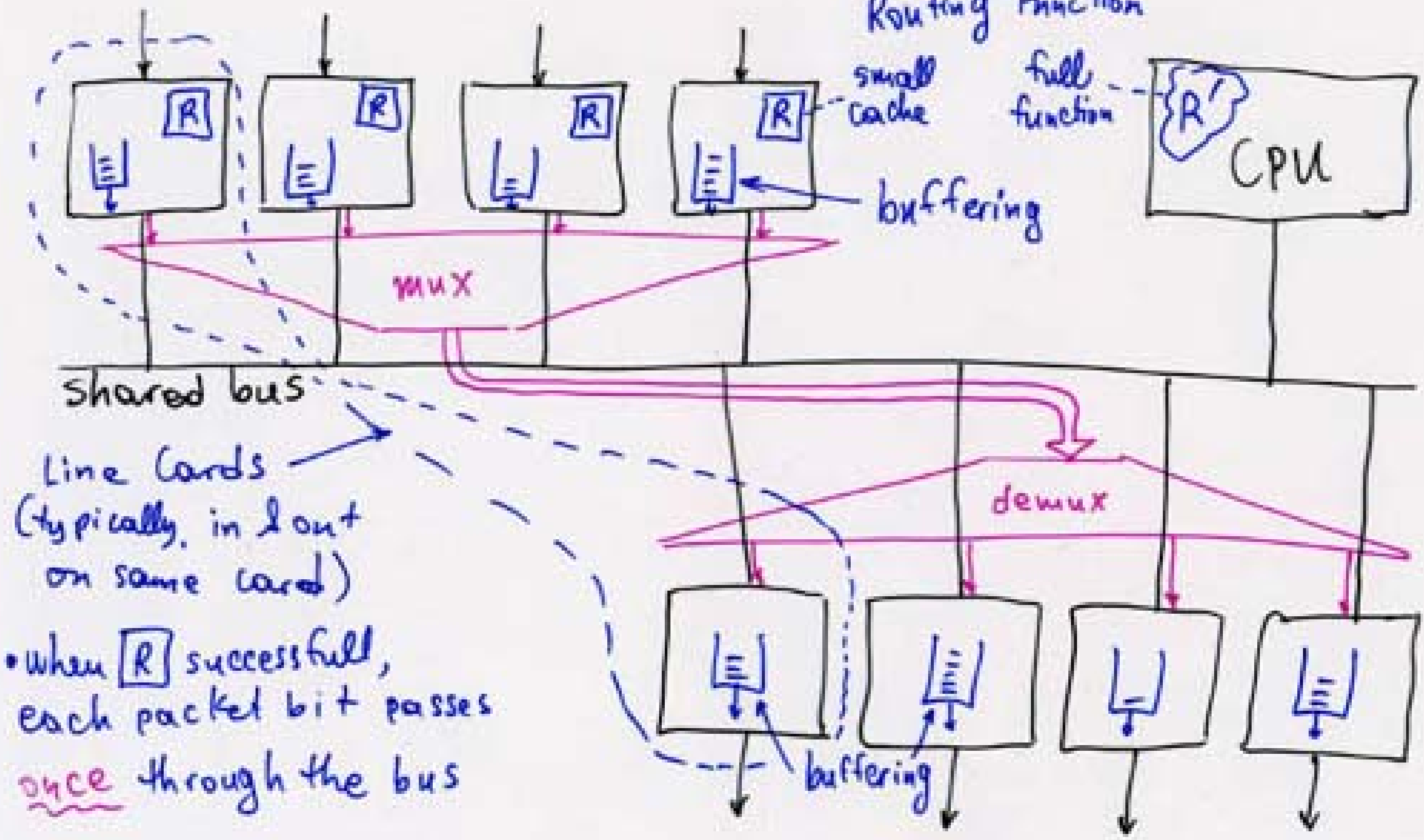
- Control structure & operation:
  - where to store the words or blocks of each packet? contiguous? scattered?
    can I mix multiple packets in one block?
  - where to store the packets going to a certain output link?
    (or from a certain input link?) (or of a same QoS class?)

"First Generation" Switches

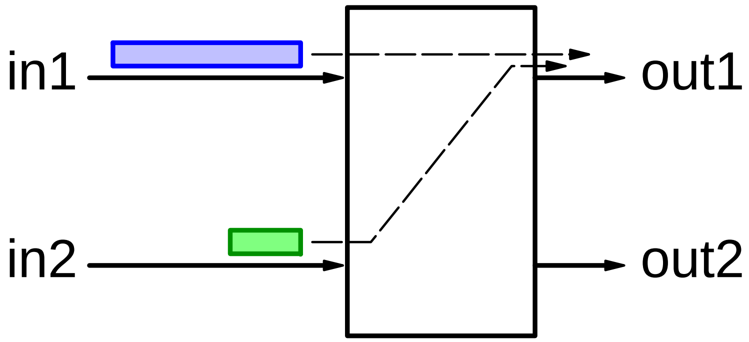


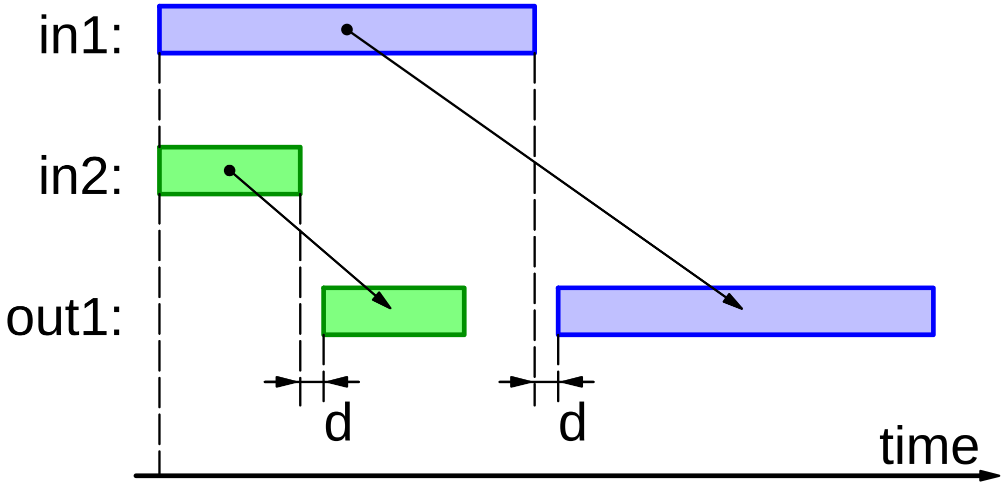
- w/o DMA: each packet bit 4 times through the bus
- w. DMA: each packet bit twice through the bus

"Second Generation" Switches



- when R successfull, each packet bit passes once through the bus

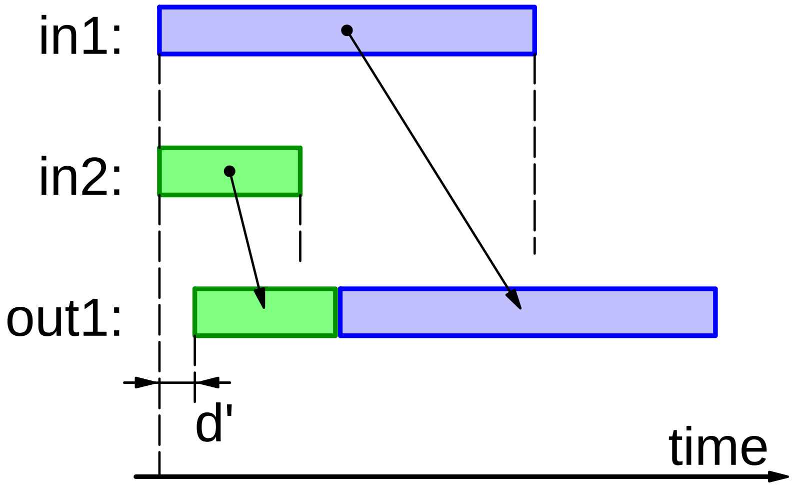




Store-and-Forward

versus

Cut-Through



Cut-through reduces delay.

Hiccup-less cut-through requires:

- hiccup-less incoming packets
- controlled rate difference between input and output