CS 414 – Multimedia Systems Design

Lecture 25 – Client-Server Buffer Management

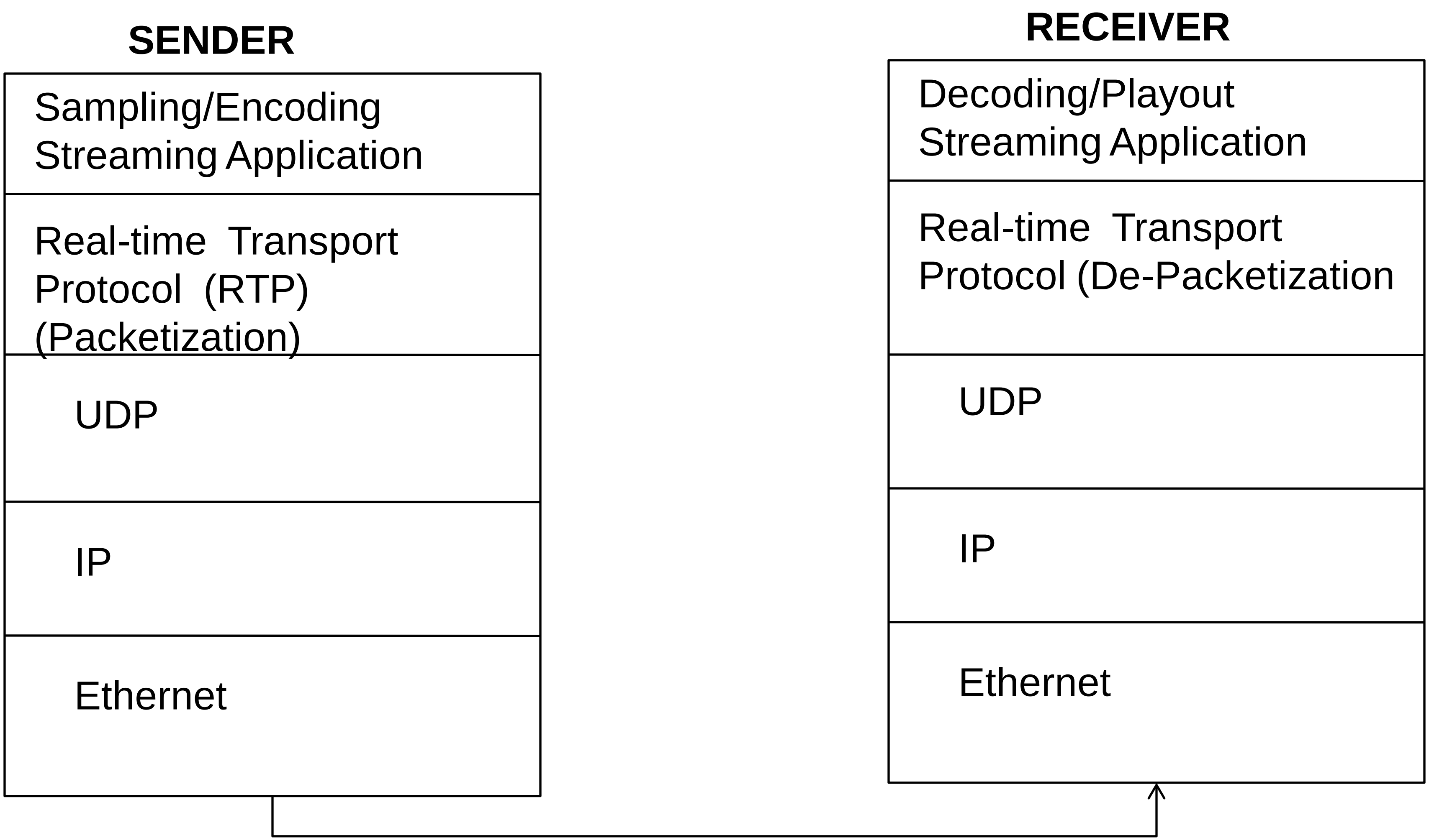
Klara Nahrstedt

Spring 2009

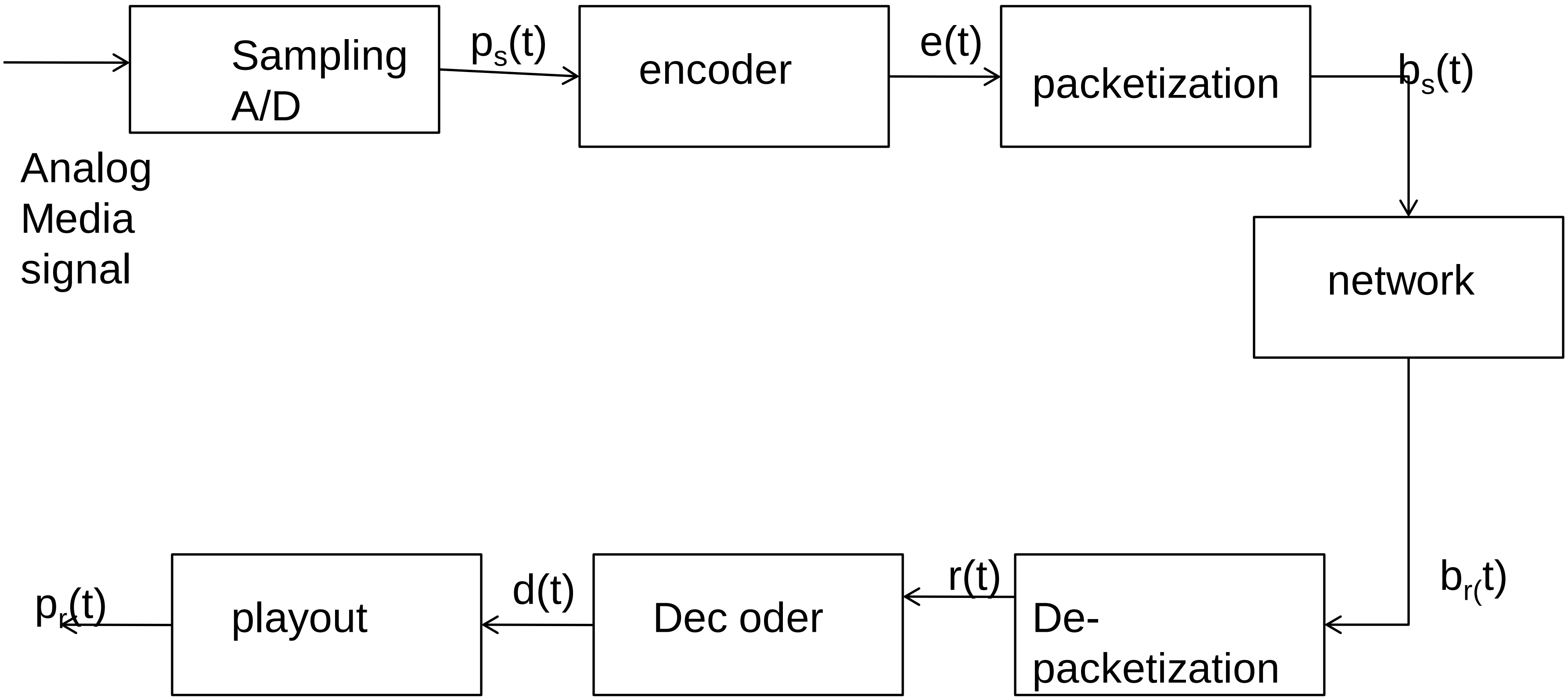
Administrative

- **MP3** is out – deadline April 6
- **MP3 Discussion Session** – Tuesday, March 31 at 7pm in 3401 Siebel Center

Layered Code in Multimedia Systems

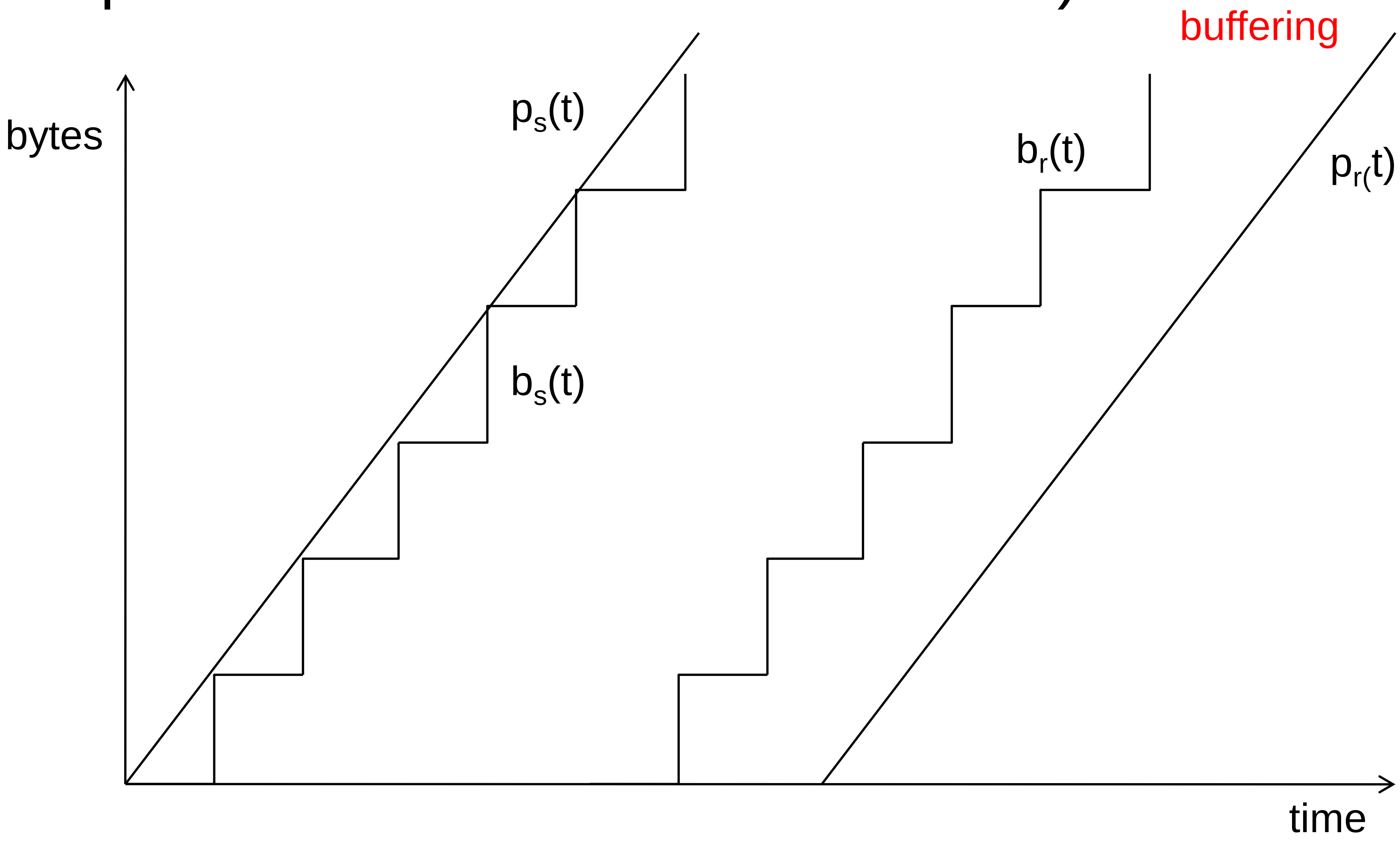


End-to-end Processing and Transmission of Digital Media Signals

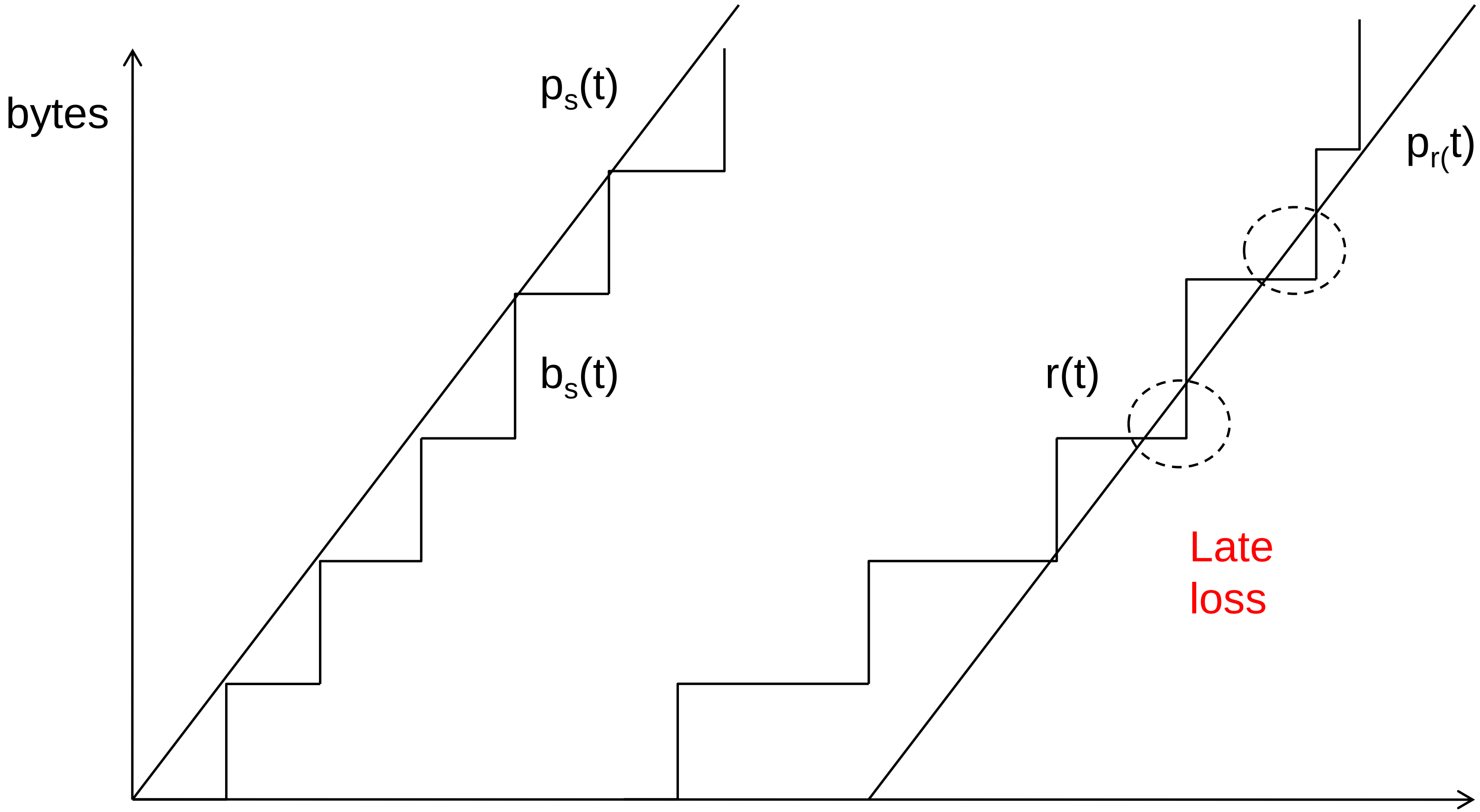


p(t) – amount of captured/playout bits its at time ***t***; ***e(t), d(t)*** amount of encoded/decoded bits at time ***t; b(t)*** - amount of clustered bits at time **t**

Example: Sender and Receiver Curves for transmission of CBR signal (digital speech with 64 kbps over a circuit-switched network)



Sender/Receiver Curves for Transmission of Voice over Packet-switched network



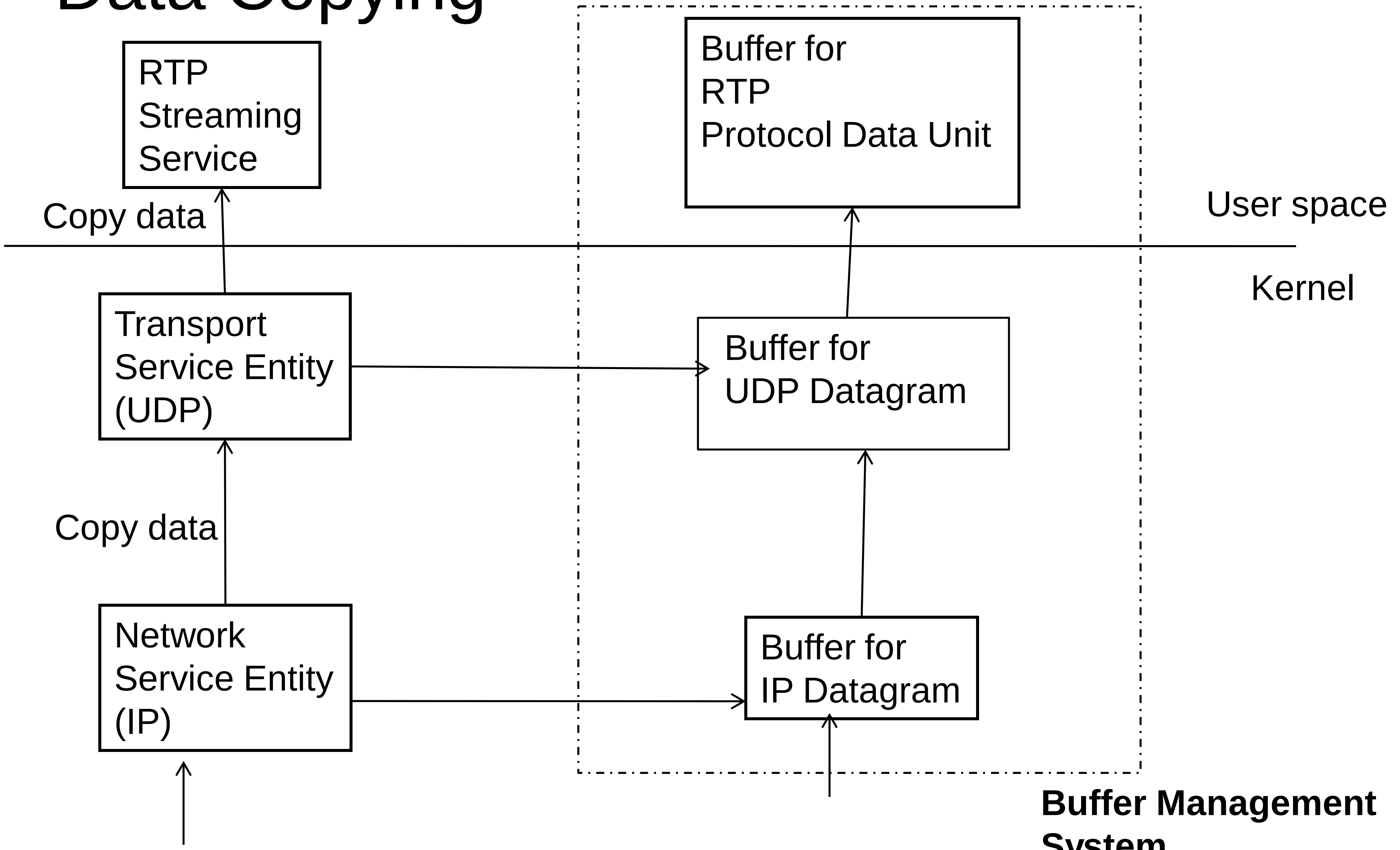
Outline

- Protocol Requirements on Buffer Management
- Buffer Management
  - Data Copying
  - Offset Management
  - Scatter/Gather System
- Buffering Strategies
  - Minbuf
  - Maxbuf

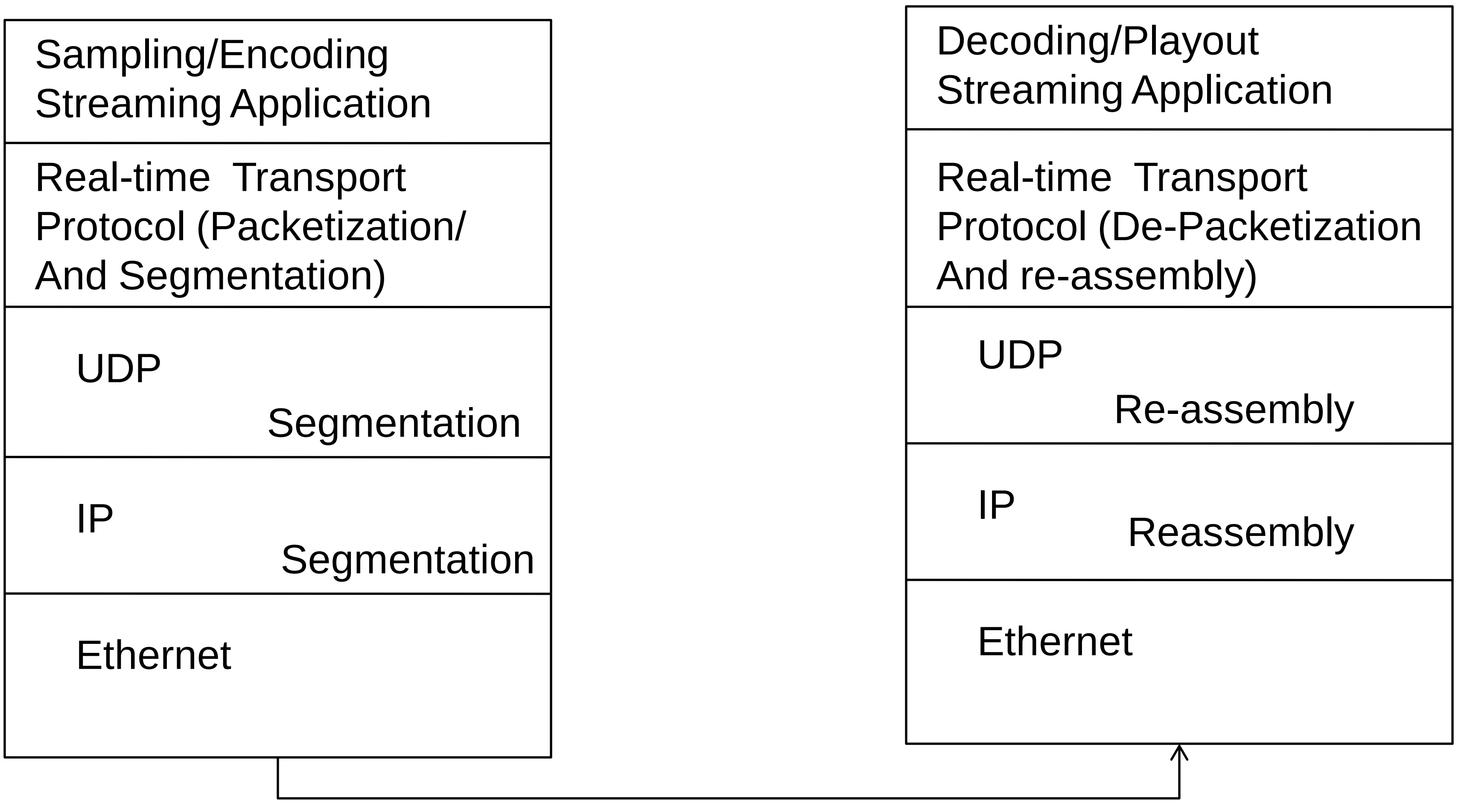
Buffer Management

- Buffers can be viewed as spatial representation of time
- Buffer plays very important role in smoothing traffic
- Network protocols buffer their service data units (SDUs) and use data copying when going from one protocol layer to another
- Moving data using data copying is very expensive

Data Copying

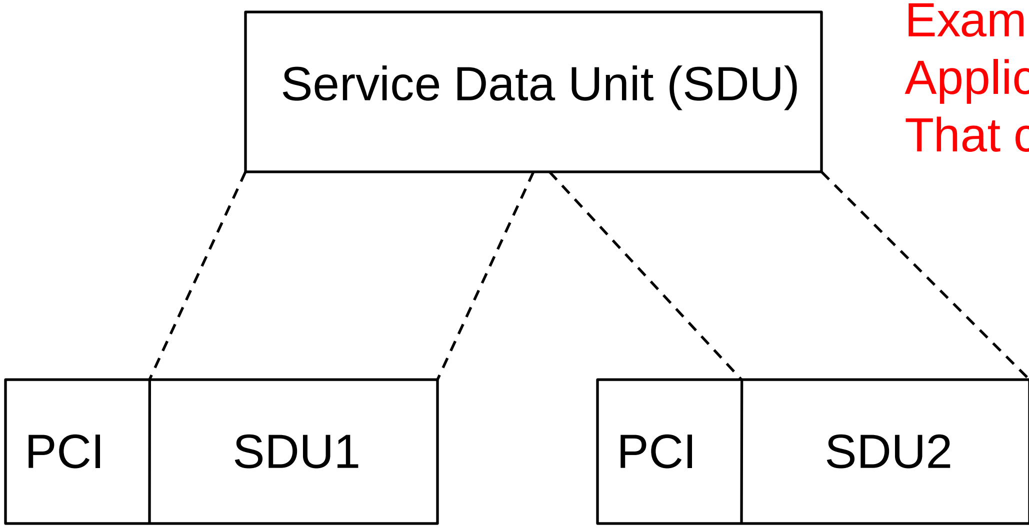


Layered Code in Multimedia Systems



Protocol Requirements on Buffer Management and Segmentation

Protocol Requirements



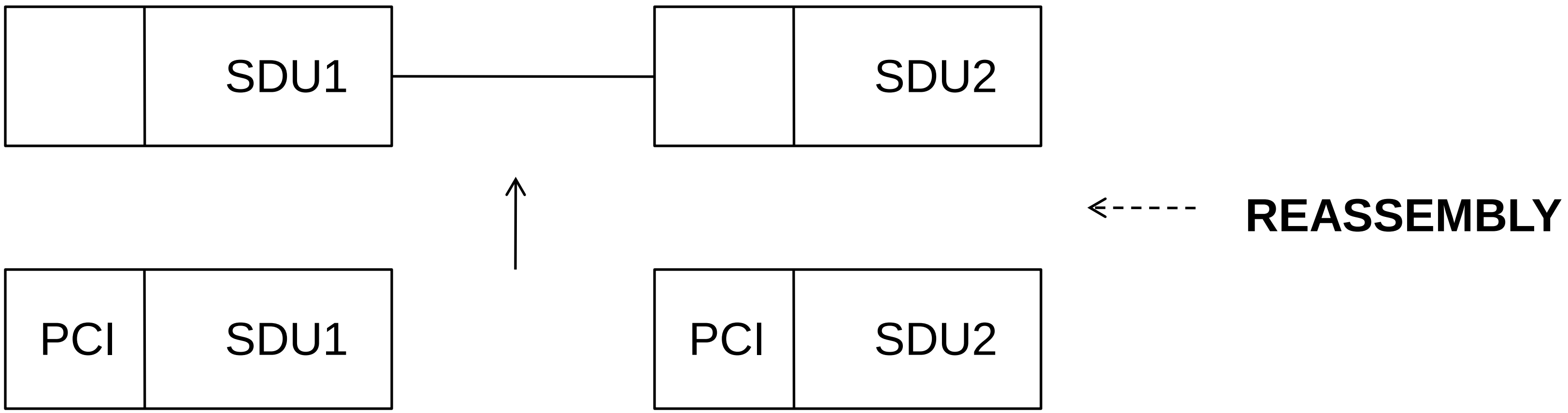
Example:
Application Data Unit – Video I frame
That comes out of encoder

← **SEGMENTATION**

Example:
I frame gets split into two
RTP datagrams

Buffer Management:
Keep identification (PCI) To which application Data unit the segment belong

Reassembly and Retransmission Requirements on Buffer Management



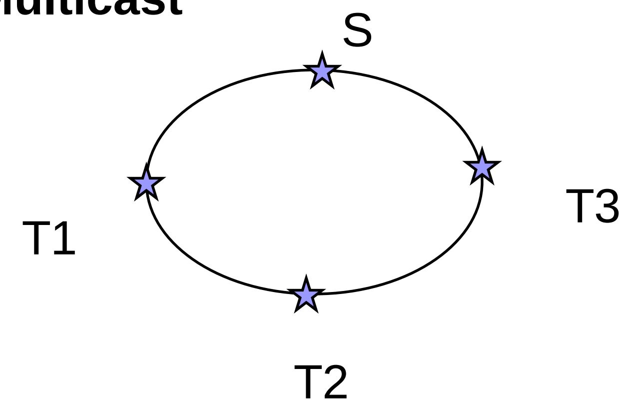
Buffer Management: support linking of memory to form one buffer

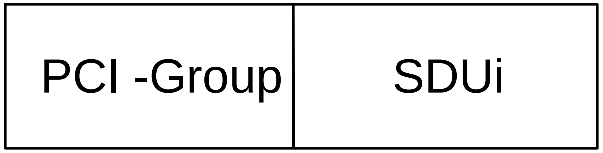
RETRANSMISSION

Buffer Management : logical copy of buffer must exist to store SDUs for Possible retransmission

Multi-cast and Multi-target requirements on Buffer Management

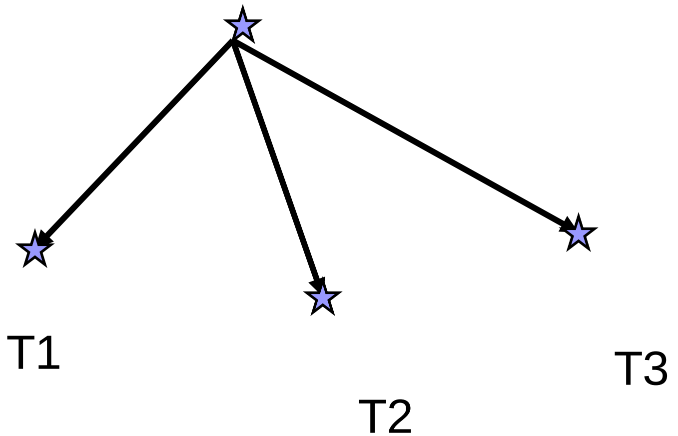
Multicast



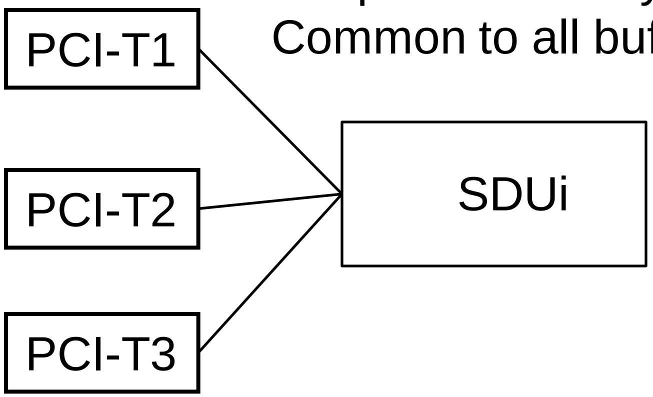


Buffer Management: Keep only
One buffer for all recipients
Use Multicast Group address (PCI-Group)

Multi-Traget

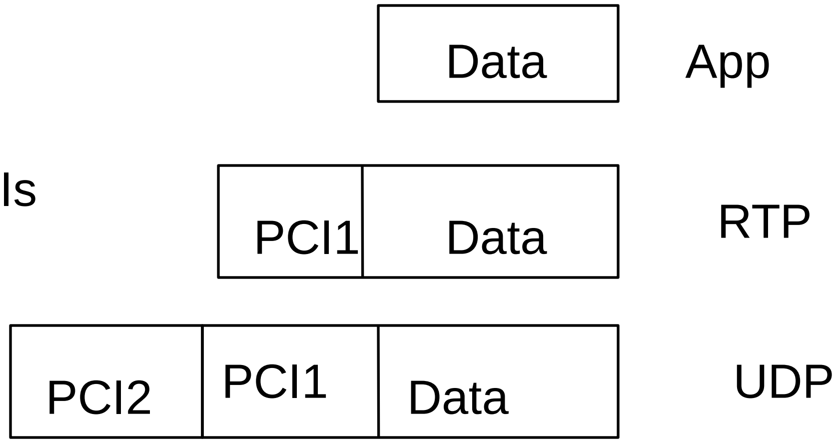


Buffer Management:
Keep one memory segment
Common to all buffers



Buffer Management Techniques

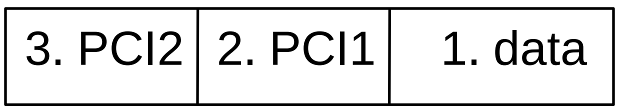
DATA COPYING



Buffer Management: Copy data and PCIs between protocol layers

Problems:???

OFFSET MANAGMENT

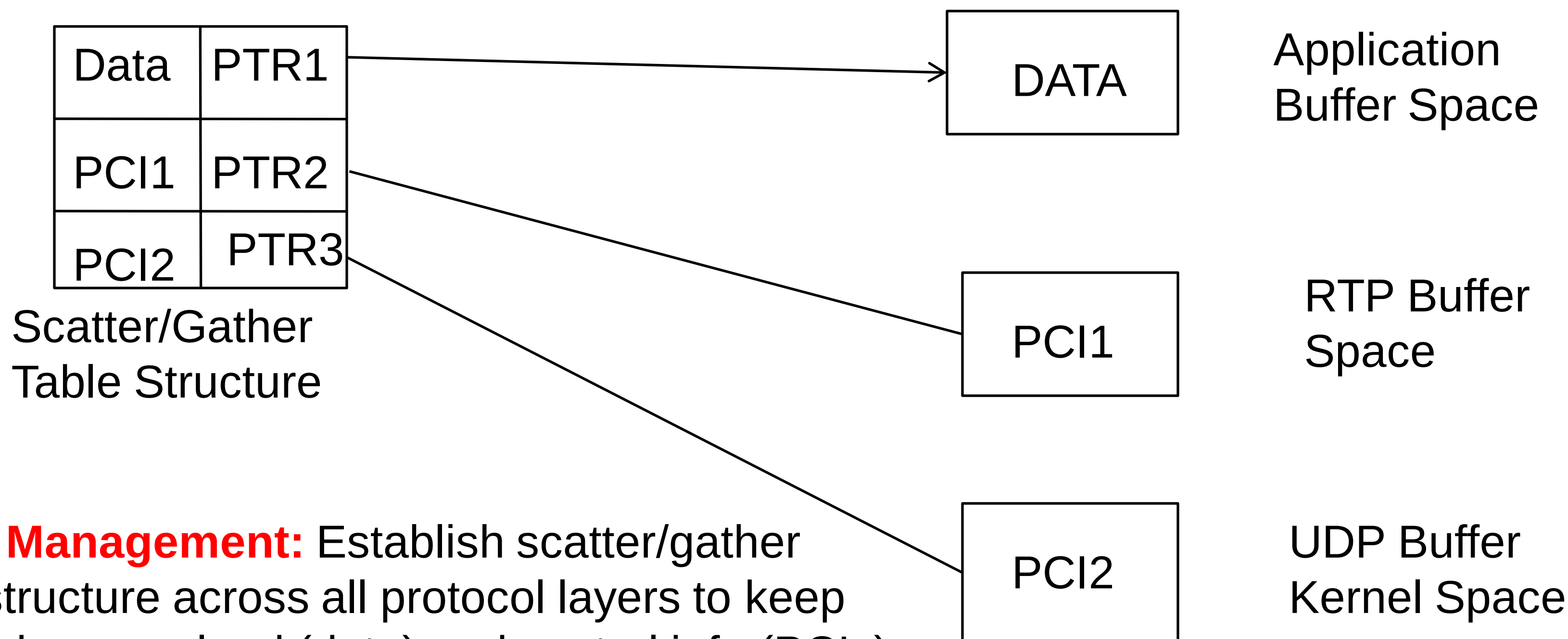


1. app
2. PCI1 added by RTP
3. PCI2 added by UDP

Buffer Management: Assign as large buffer as data + all headers of the protocols require

Problems:???

Buffer Management Techniques (Scatter-Gather)



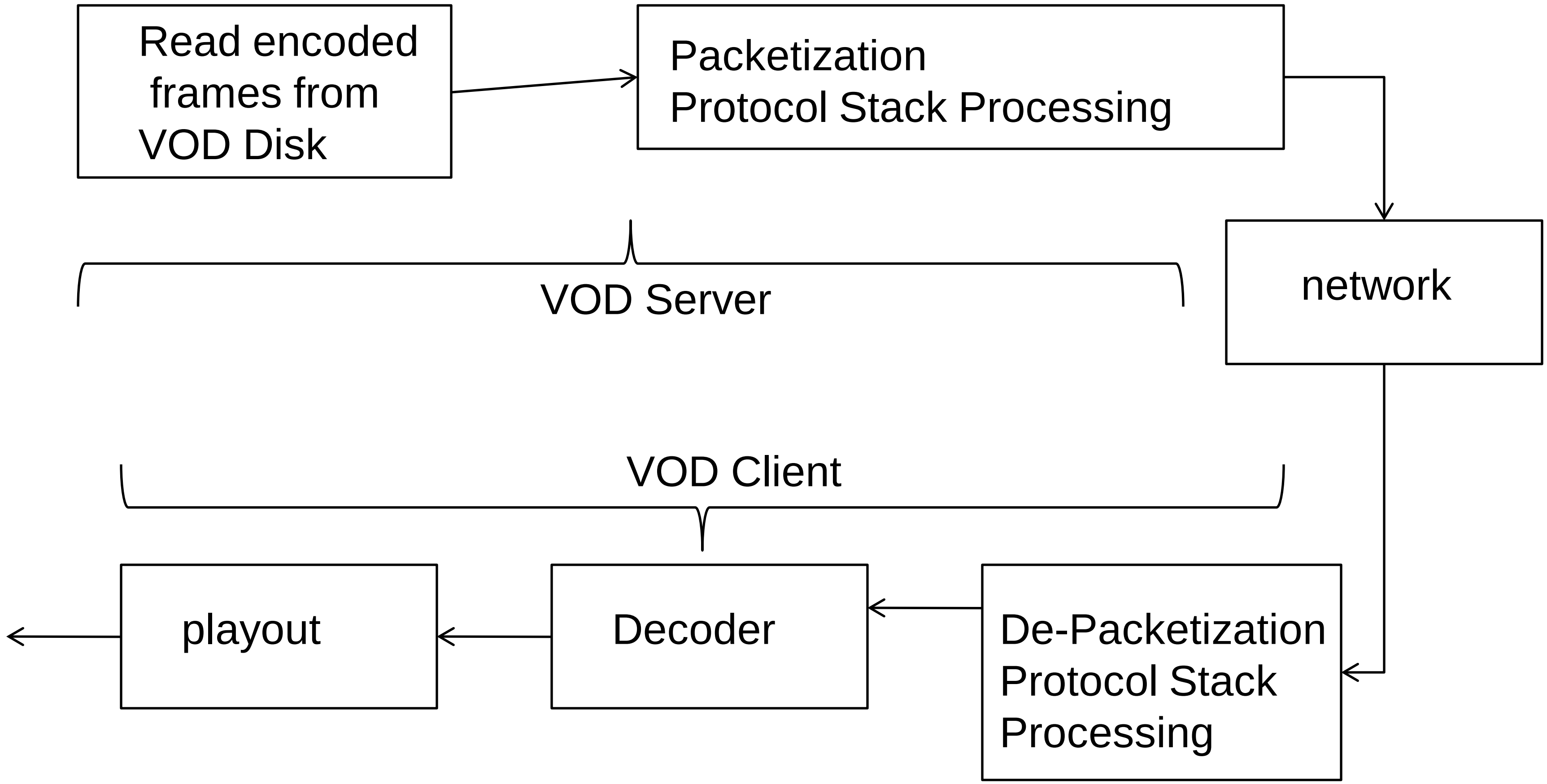
Buffer Management: Establish scatter/gather Table structure across all protocol layers to keep Track where payload (data) and control info (PCIs) Are located

Problems: ???

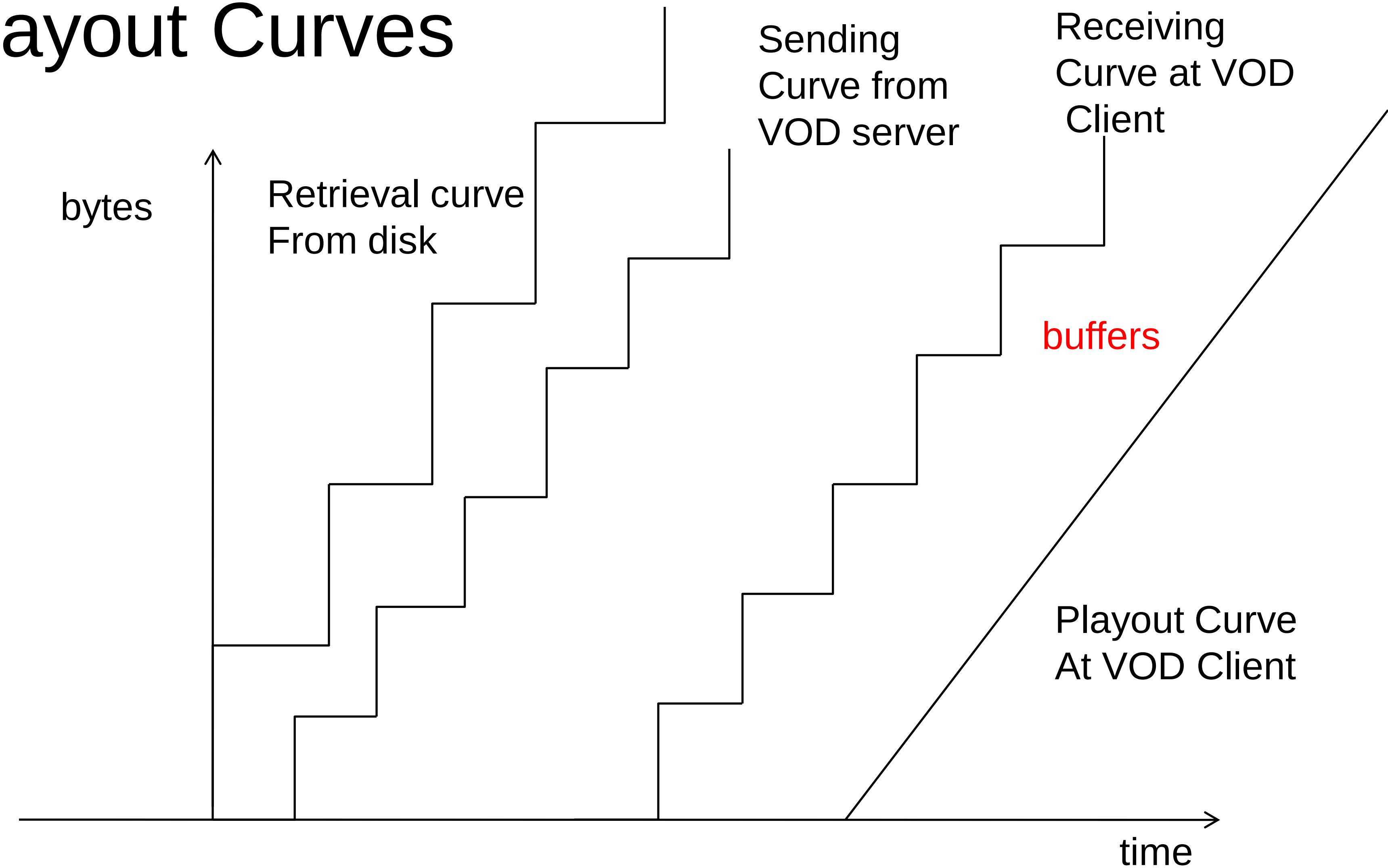
Comparison

	Data Copying	Offset	Scatter/Gather
Memory BW	High	Low	Low
CPU BW	High	Low	Low
Memory Usage per Layer	Optimal for individual protocol layer because exact amount of space will be allocated	High for application protocol because it must allocate space more than it needs	Compromise, segments are sized depending on requirements
Are Protocol Requirements without copying satisfied?	NA	No, segmentation needs copying	Mostly yes (one copy and segmentation must be done when data leaves node)

Buffering Strategies in Client-Server Systems



VOD Retrieval Transmission and Playout Curves



Buffer Management Strategies

- Clients need buffers for VOD (Video-on-Demand) application to smooth out traffic jitters
- Buffer management strategies balance bits in transit (buffer size and bits in channel)
- Fixed Strategy (non-adaptive) and dynamic (adaptive)

Buffer Management Strategies

- Fixed buffer strategy
  - Static buffer allocation during multimedia call setup phase
  - Static buffer allocation does not change during run-time
- Dynamic buffer strategy
  - Elastic buffer allocation , i.e., allocate buffers during multimedia call setup, but change them during run-time

Buffering Strategies at VOD Client

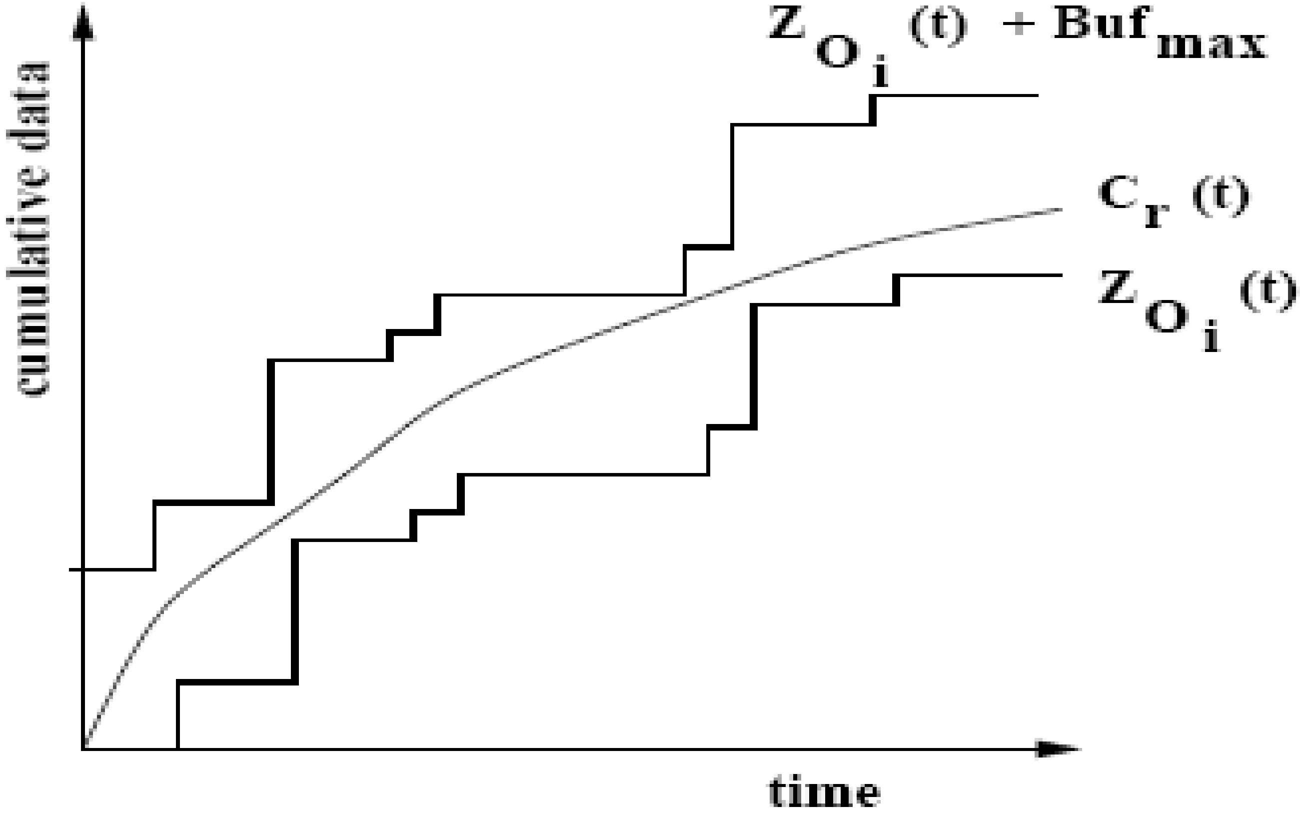
- Minbuf - minimum buffering strategy
  - Minbuf minimizes buffering requirements at VOD client, but makes more demands on network (throughput and delay guarantees)
- Maxbuf – maximum buffering strategy
  - Maxbuf buffers more than one unit of information and eases QoS guarantees demands on network
  - Buffering only up to a limit (Buf_{max})

Buffering Model

- $Z_{Oi}(t)$ – amount of bits in multimedia object (e.g., Video frame) O_i displayed at time t
- $C_r(t)$ – amount of bits received at receiver at time t
- Two buffer states in each buffering strategy:
  - Starvation if $C_r(t) \le Z_{Oi}(t)$
    - If $C_r(t) > Z_{Oi}(t)$, no starvation at VOD client
  - Overflow if $C_r(t) \ge Z_{Oi}(t) + Buf_{max}$

Client Buffer Constraints

(Received amount of data $C_r(t)$ should always be between $Z_{Oi}(t)$ and $Z_{Oi}(t) + Buf_{max}$)



Implications of Minbuf Strategy

- With minbuf strategy – delivery time of a unit of information is the display time of previous unit
- Minimum Throughput (*Th*) in bits per time unit required is

$$Th_{\min}^{\min buf} \geq \frac{Z_{Oi}}{t_i - t_{i-1}}$$

- Delivery time instant of the first unit before start of display: $t_x = \dfrac{Z_{Oi}}{Th_{\min}^{\min buf}}$

Implication of Maxbuf Strategy

- Delivery schedule of VOD server and network may cause that Buf_{max} bits will be delivered every K seconds, where
  - $K = Buf_{max}/Th$, Th – throughput of network
- Minimum Throughput required
  - $\sum Z_{Oi}$ – total size in bits of all objects that will be presented in stream:

$$Th_{min}^{max\,buf} \geq \frac{\sum_{i=1}^{n} Z_{Oi} - Buf_{max}}{StreamDisplayDuration}$$

Implementation Issues

- Buffers for Uncompressed Periodic Streams
  - Use **circular buffers** as prefetch buffers with maxbuf strategy
- Buffers for compresses periodic streams
  - Use **circular buffers**, but carefully consider the size of buffer unit (depending on MPJEG or MPEG) with maxbuf strategy
  - For MPEG may consider dynamic buffer allocation
- Buffers for control information
  - Use priority queues or FIFO
- Buffers for Non-RT data
  - Use maxbuf strategy with static buffer allocation

Memory Management

- Virtual memory versus real memory paradigm
- In VM – paging introduces unpredictable delays
- Multimedia timing requirements suggest that no paging is desired, however it might be difficult to do from user space
- Multimedia applications may want to pin pages into memory which include their time-sensitive code
- On-Chip Caching is desired
  - Intel Pentium Processor with MMX technology (SIMD Instruction Set) has on-chip cache of size 32KB for video processing only

Conclusion

- Need buffering at VOD client side
- Some buffering needed also at VOD Server side
  - Can use FIFO techniques
- Reservation memory schemes are possible
  - Implemented in system, called RK (CMU)
- **Will talk about VOD server in next lectures**

Memory Reservation Concept

- Possible
  - Done in hard real-time applications
  - Done in some multimedia applications – pin multimedia applications
- Be careful when you pin only the application since if other services are paged, the application will wait anyway
- Possible architecture
  - Memory broker and memory controller
  - Memory broker pins a certain size of memory – global reserve – shared among RT processes

Reservation Concept

- *Admitted storage* – memory size from the global reserve that has been already allocated to accepted processes
- *Available storage* – free memory from global reserve that is available for new RT processes
- *GlobalReserve:= AdmittedStorage + AvailableStorage;*
- Admission control:
  - test: RequestedStorage ≤ AvailableStorage
    - If yes, then
      - AdmittedStorage:= AdmittedStorage + RequestedStorage
      - AvailableStorage: = AvailableStorage – RequestedStorage
    - Else reject request for RequestedStorage

Reservation Protocol

- Step 1: SRT (Soft Real-Time) Process declares '*RequestedStorage*' to Memory Broker
- Step 2: Admission Control performed in broker
- Step 3: If Admission positive, broker builds memory reserve of size *'RequestedStorage',* allocated to SRT process during time critical run
- Step 4: Memory Broker
  - provides reservation ID to SRT process,
  - creates entry in reservation table, and
  - informs memory controller

Reservation Protocol

- Step 5: SRT process uses its reservation ID during execution to get memory allocated from memory controller
- Step 6. Memory controller
  - checks the validity of the reservation, and if positive,
  - allocates shared memory to SRT from global reserve pool;
  - Gives key of shared segment to application which then attaches the shared memory segment to its own address space

Reservation Protocol

- Step 7: After SRT finishes, it
  - Frees shared memory segment from its application space,
  - Sends an event to memory controller with key to shared memory
- Step 8: Memory controller
  - Frees shared memory
  - Increases available storage information for broker
    - AvailableStorage: = AvailableStorage + FreedStorage