# TCP-Friendly Lossy Streaming to TCP Clients

GEMINI Seminar

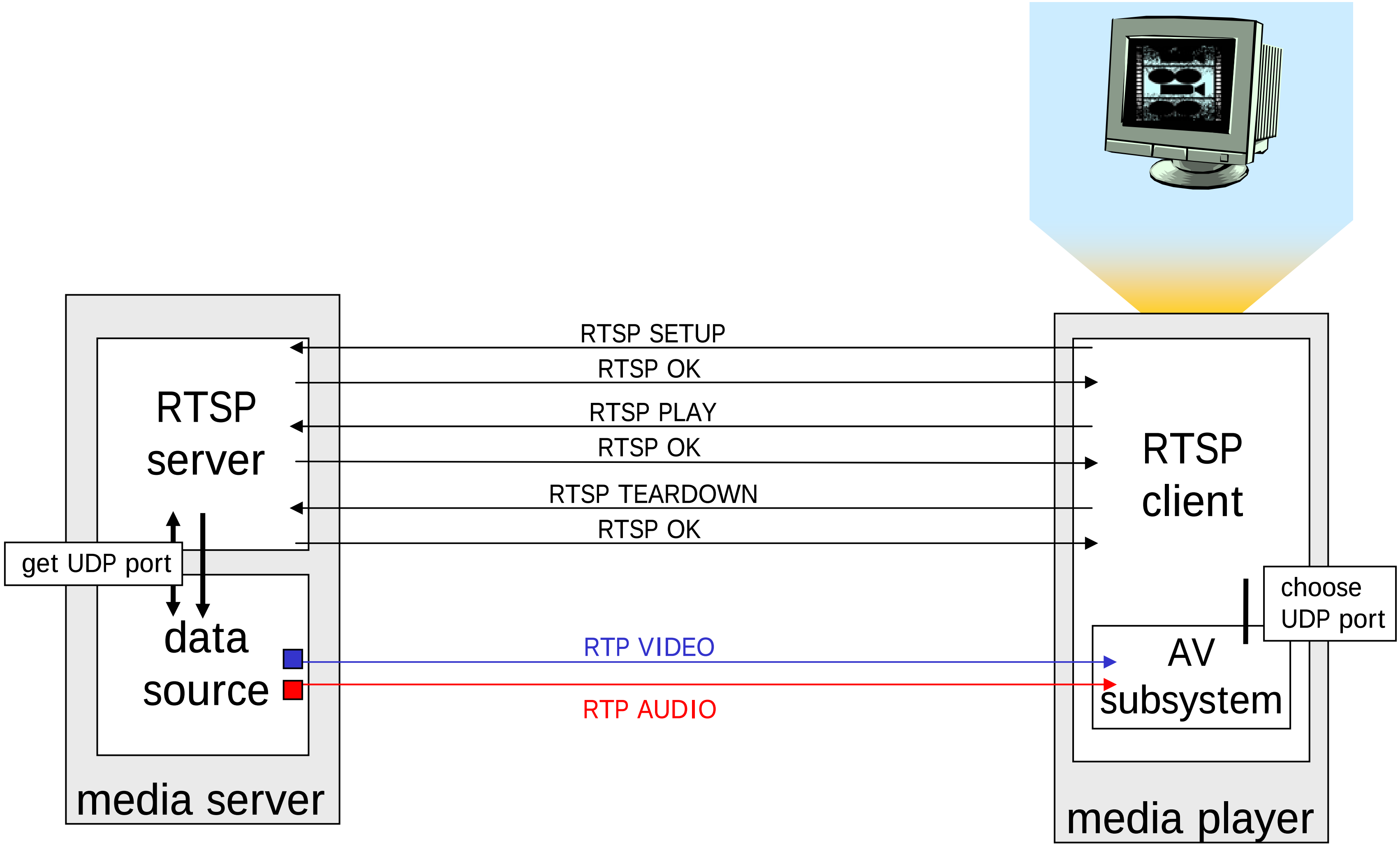
12./13. March 2007

# Motivation

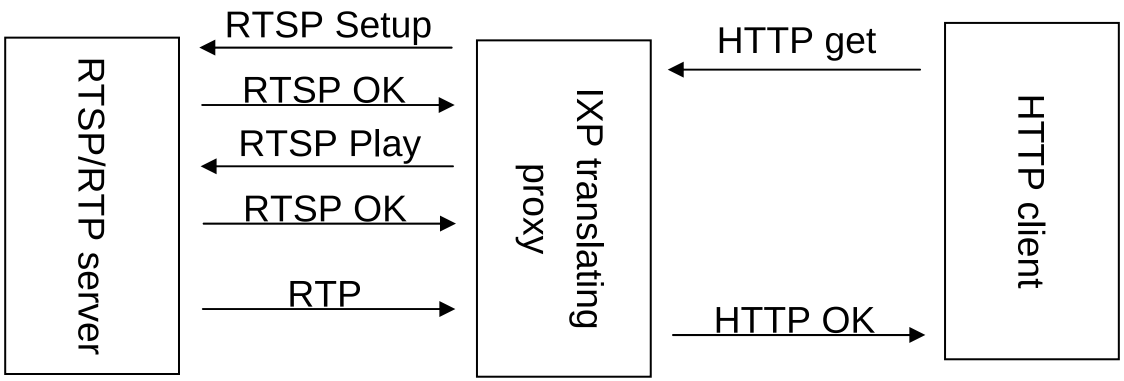
- Streaming video in the Internet
- TCP-Streaming is wide-spread
  - Recent log analysis of an MS Mediaserver shows roughly
    - 1/3 MMS/UDP
    - 1/3 MMS/TCP
    - 1/3 HTTP
  - I.e.: 2/3 of all streaming is over TCP
- Problems with TCP-Streaming
  - Congestion translates into delay instead of packet loss
  - Long buffering delays at the client
  - Hick-ups at the client
  - No exploitation of multicast at the server
  - Increased memory consumption for TCP buffer handling

# RTSP Operation

- Integration with other real-time and multimedia protocols



# HTTP/RTSP translation

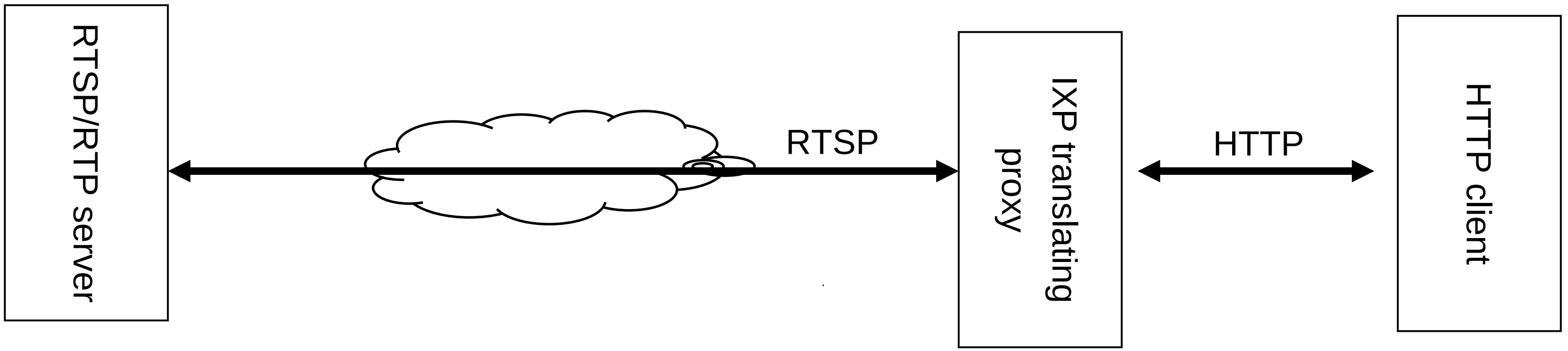



- Protocol translation
- Congestion control
  - But not AIMD
- Flow control
- Fake retransmissions
- No buffering

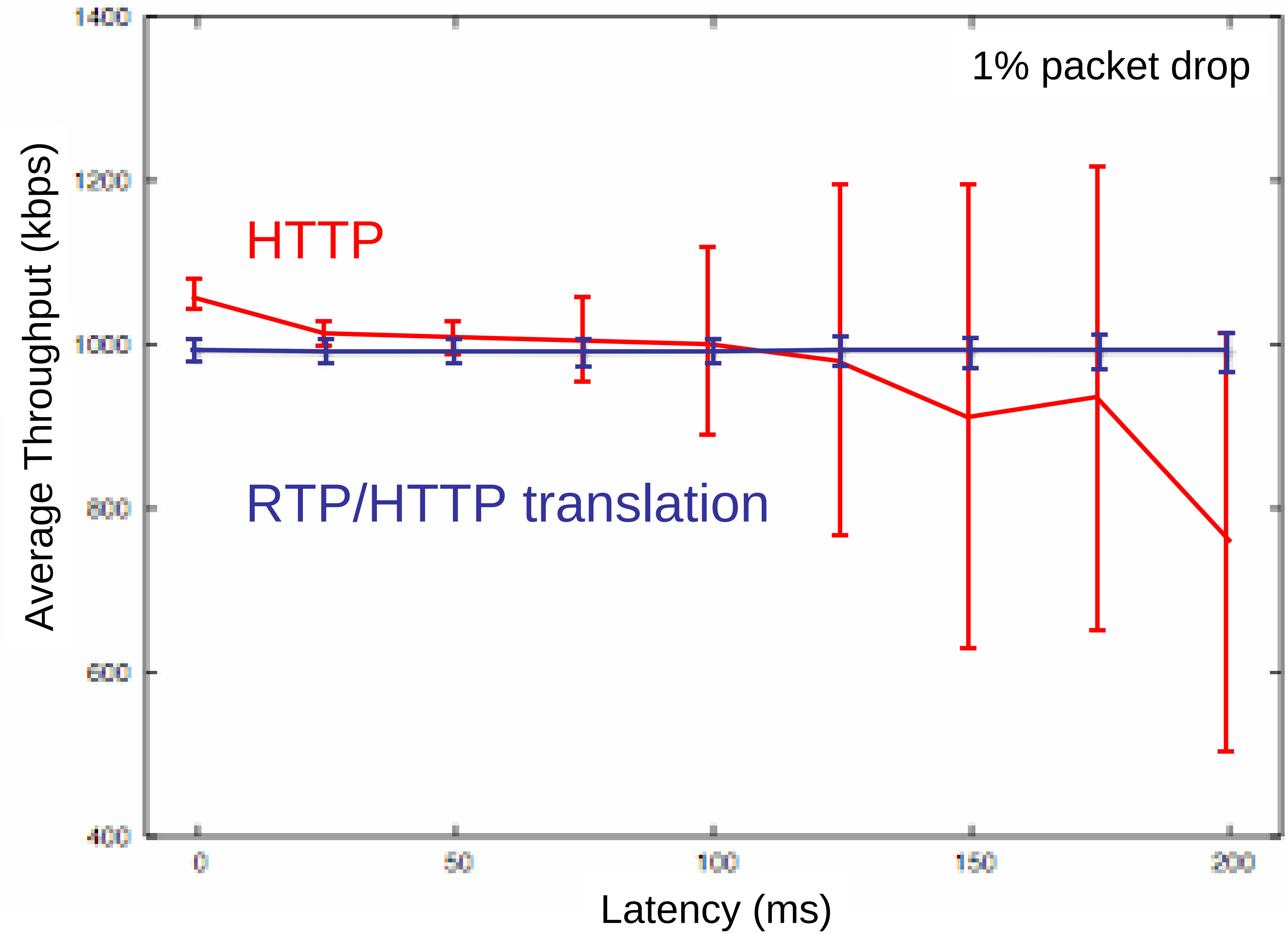
- Proxy drops randomly
- Maintain TCP sequence numbers
- Transmit 0-filled packets when retransmissions are required

# Adapt in a client-sided translator

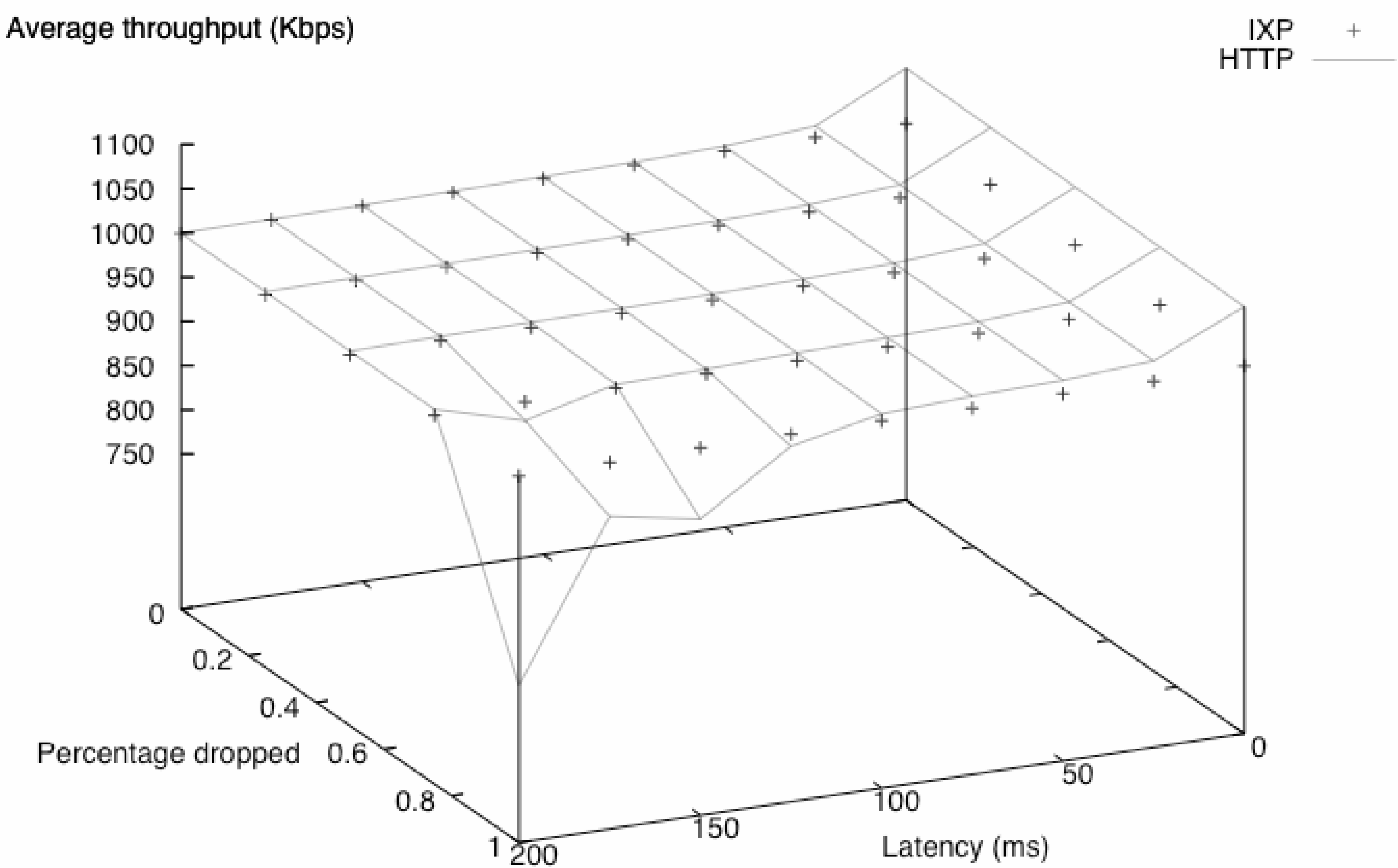
- RTP/UDP sending rate
  - Is the long-distance part
  - Must adhere to rules
  - Adapt to TCP or TFRC–given rates
- Advantages
  - Long-distance communication is RTP/UDP
  - Straight-forward translation
  - Short-distance TCP has quite stable rates on the timescale of video streaming
- Disadvantages
  - Translator installed on ISPs' premises



# Adapt in a client-sided translator



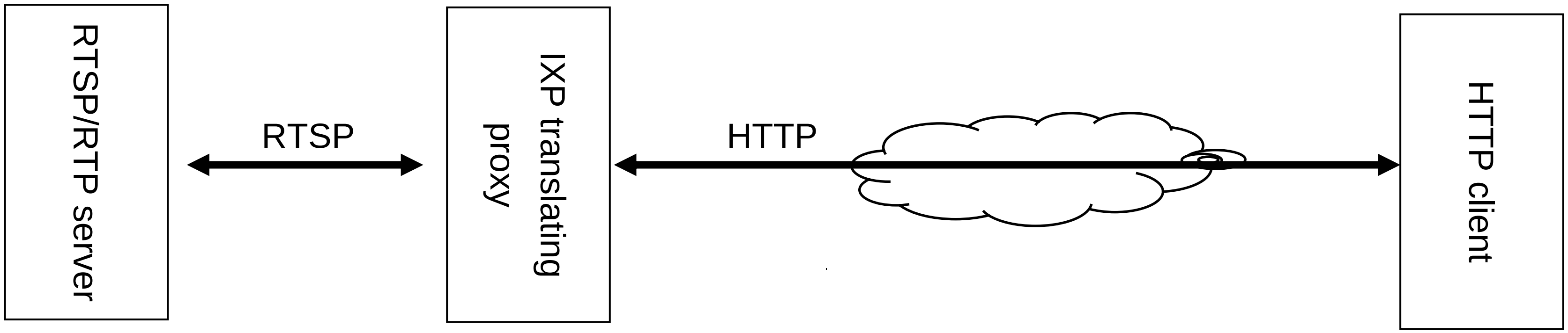
# Adapt in a client-sided translator




INC paper 2004, TCP Ulem

# Adapt in a server-sided translator

- RTP/UDP sending rate
  - Is the short part of the data path
  - Only on sender's premises
  - No need to adhere to rules
- Advantages
  - Reduce server load
  - Can use multicast functions inside the server
  - Does not require off-site proxy deployment
  - Can use TCP-friendly long-distances rates instead of TCP's AIMD
- Translation
  - Must drop actively to be TCP-friendly
  - Use TFRC–rate as envelope
- Disadvantages
  - Needs to handle TCP's credit window
  - Needs to handle retransmission semantics
  - Clients may be unable to accept gaps in a stream
  - Clients may not understand 0-filled packets



# Adapt in a server-sided translator

- TCP-Friendly Rate Control (TFRC)
  - Equation-based TCP-friendly congestion control
  - Receiver sends rate estimate and loss event history
  - Sender uses models of SACK TCP to compute send rate
  - One proposed mechanism for DCCP

$$T = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO}\min(1,3\sqrt{\frac{3bp}{8}})\,p(1+32p^2)}$$

Steady state TCP send rate

- Why should it work?
  - Because we drop packet-sized units
  - They are used in RTP/UDP -> this is a reasonable drop unit
- Effect
  - No re-buffering delays

by Padhye, UMass

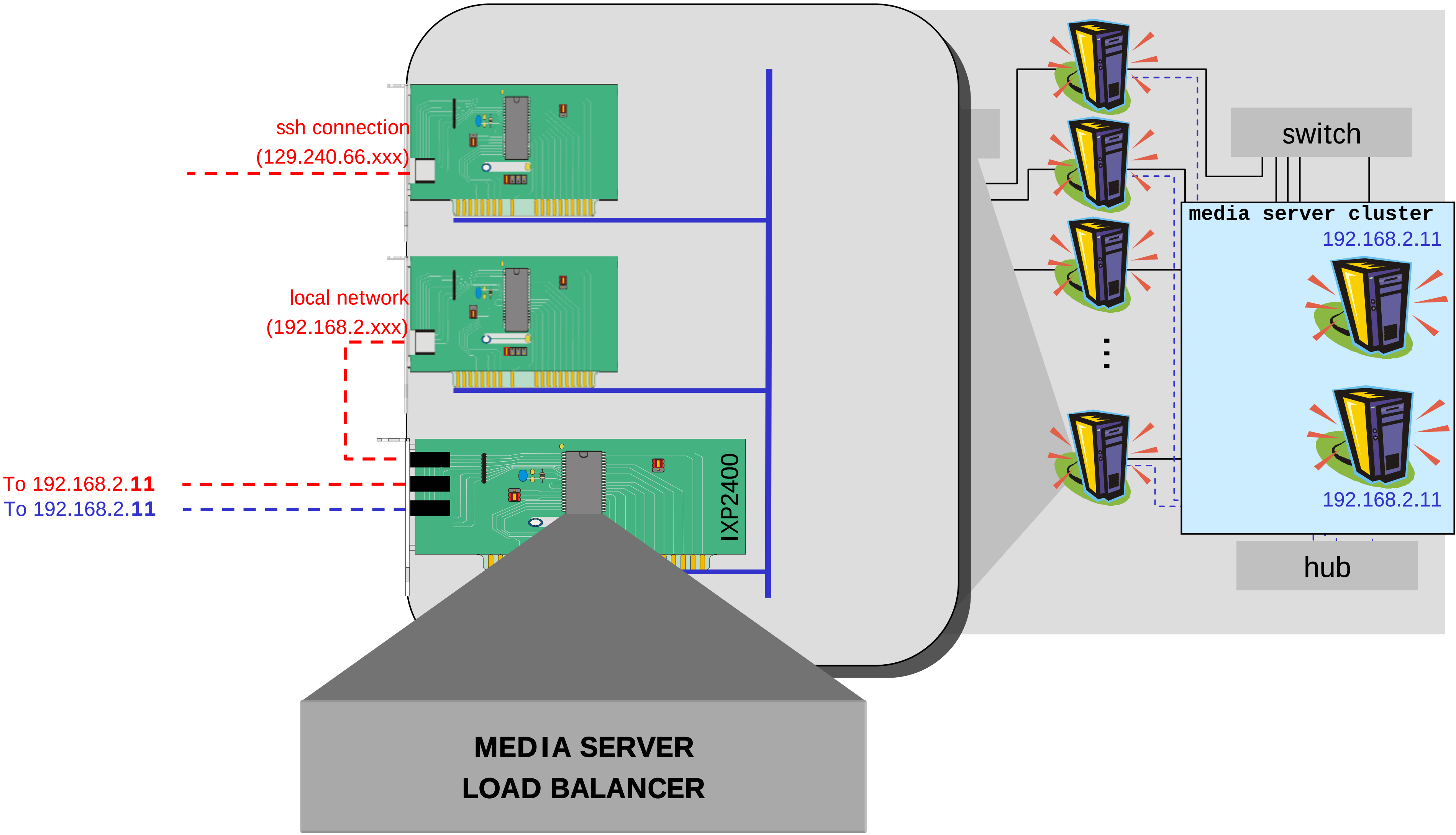
# Asymmetric multiprocessing

- New type of processors
  - Promises scalable applications
  - Applications are so far only optimized for exclusive use
  - Should allow more general use

- Existing asymmetric multicore processors
  - Intel IXP
  - Sony/Toshiba/Intel Cell
  - Nvidia Cuda
  - (Tandberg)

- Practical work so far on IXP
  - E.g. a protocol translating proxy

# IXP implementation

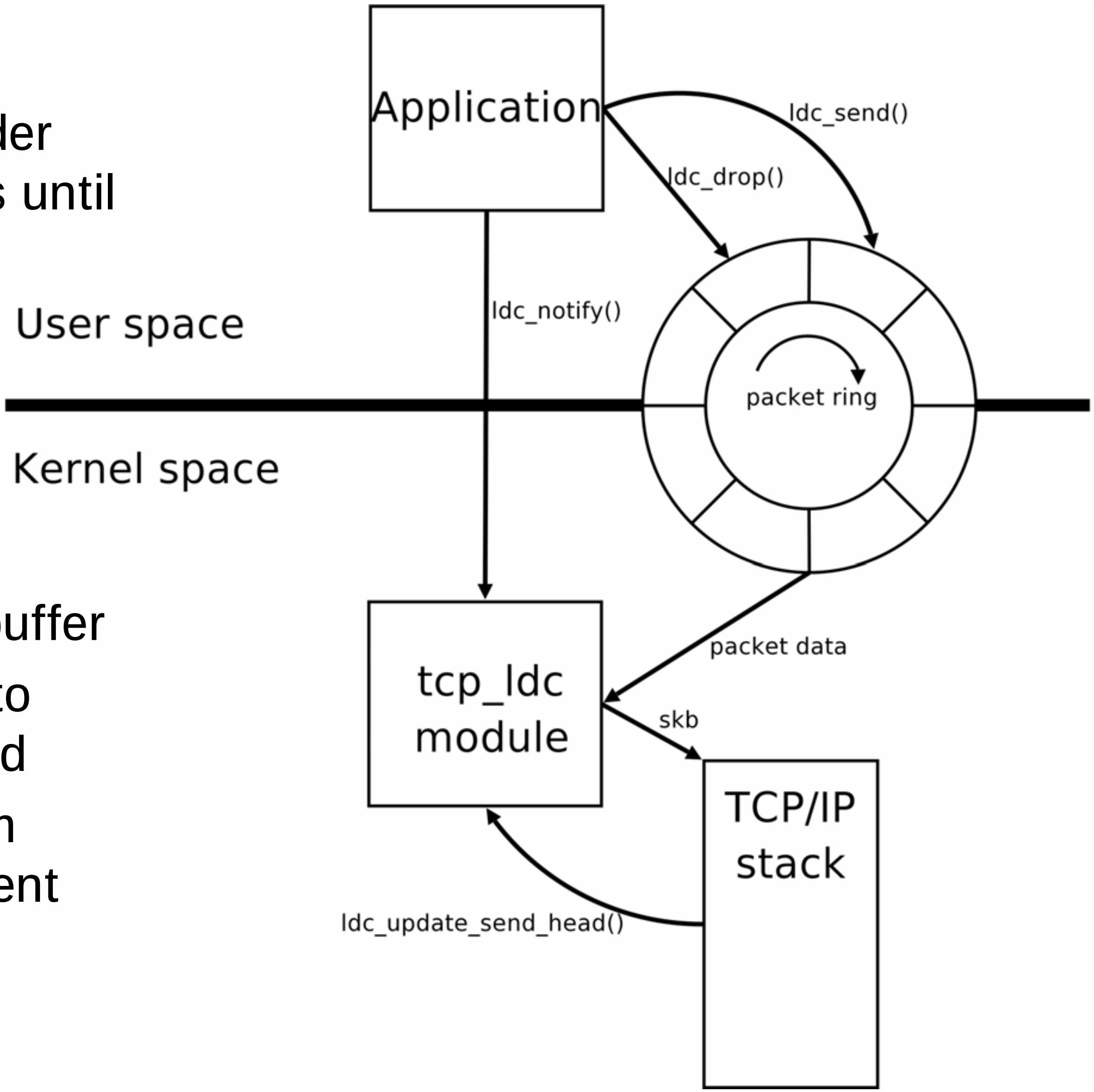


# Related ideas

- Transcoding proxies
  - All kinds of possibilities - including exactly this idea
- PRTP-ENC
  - Client-side change
  - Random loss up to a specified percentage is accepted
  - ENC-marks instead of loss reports allow congestion control to work correctly
- LDC
  - Sender-side change
  - Take late data out of the send buffer
- Kernel multicast
  - Sender-side change
  - Attach several destination addresses to one socket
  - Make in-kernel copies

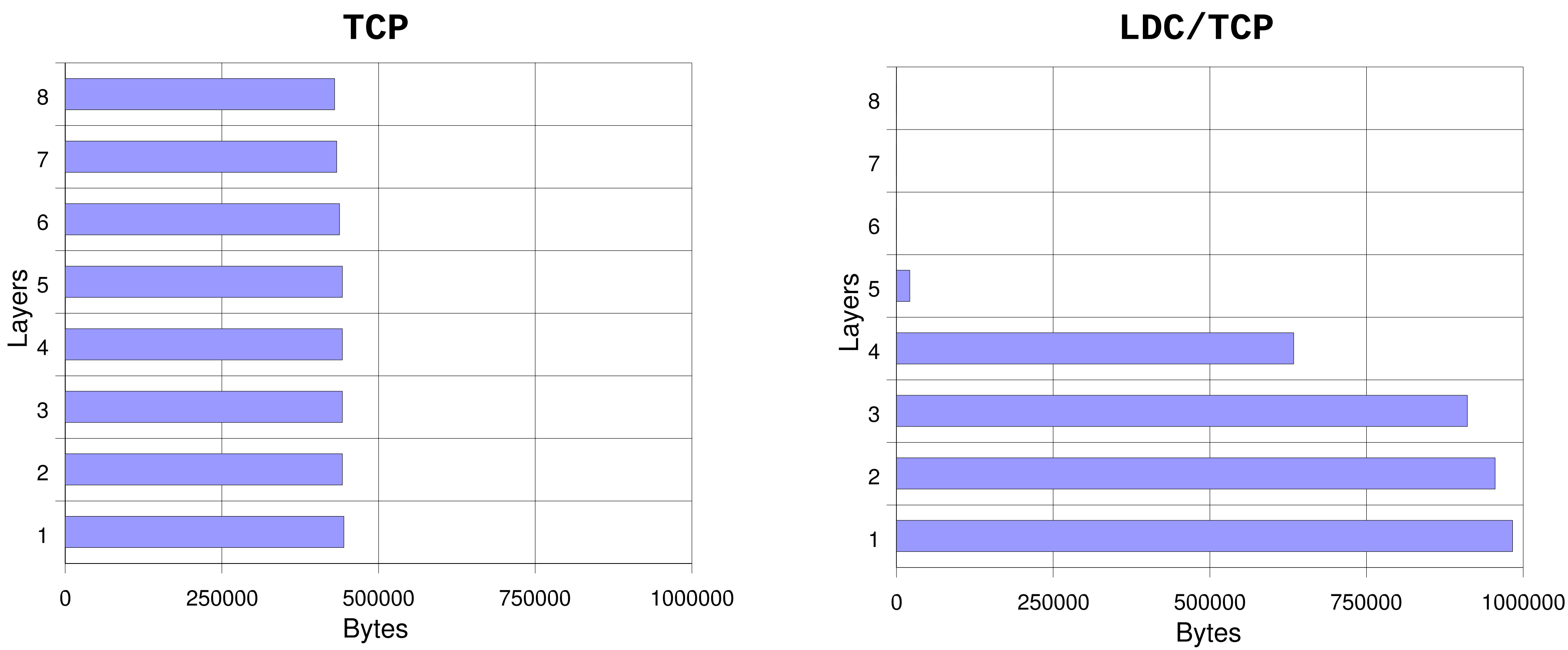
# LDC API for TCP

- Concept
  - Application can reconsider sending specific packets until they are sent once
- Approach
  - API extension
  - Create a shared buffer
  - Application sends to a buffer
  - Application drops (trys to drop) when reconsidered
  - TCP takes 1 packet from buffer when packet is sent
  - TCP hops over dropped packets



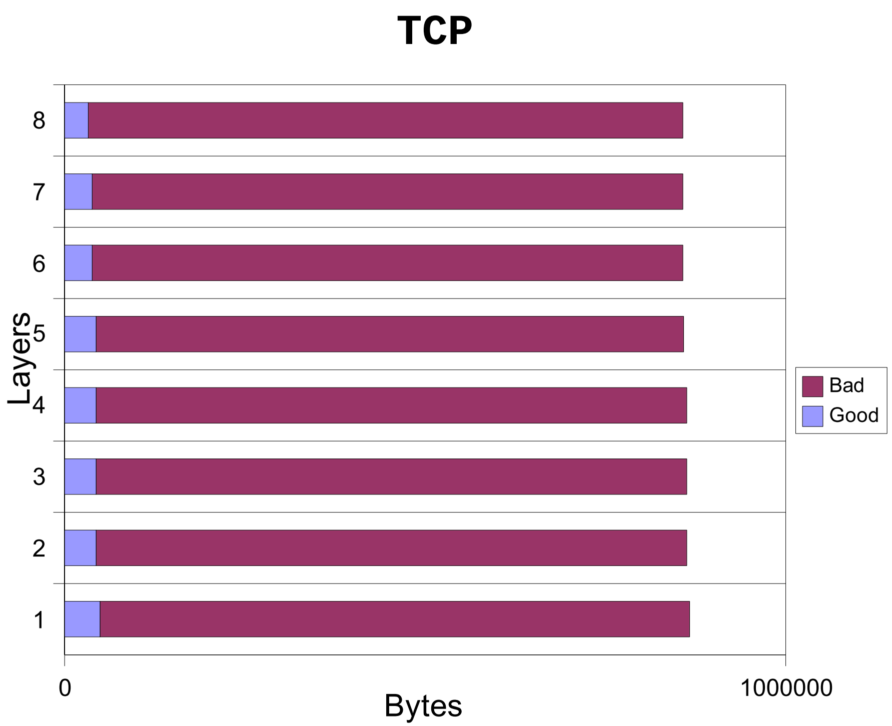
# LDC: Late data choice

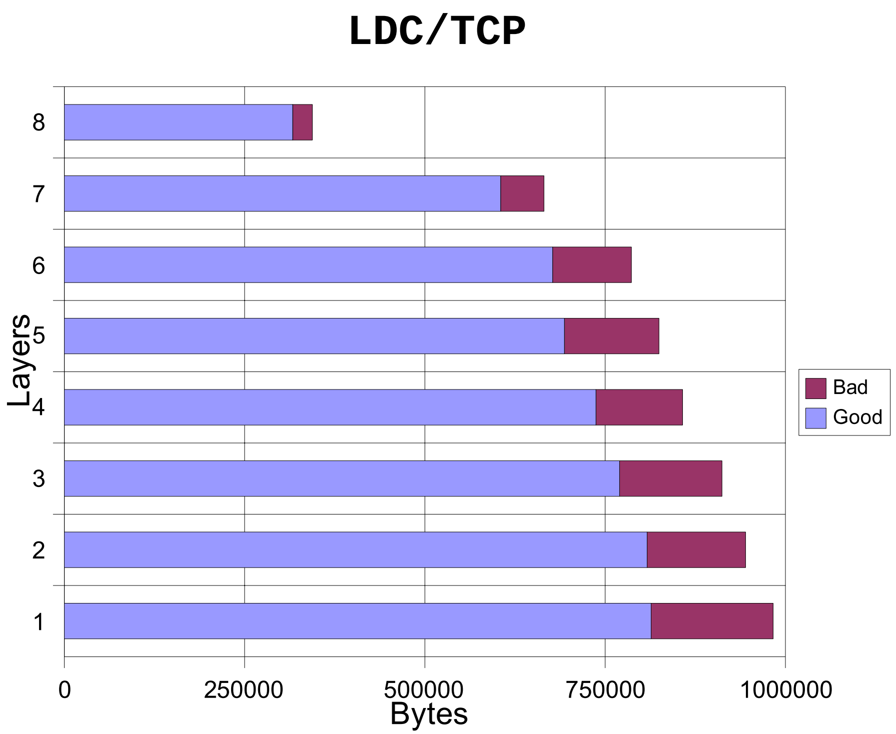
- amount of layered video data received in 60s
- 1 Mbps in 8 layers of 128 Kbps
- 5% loss



# LDC: Late data choice

- goodput achieved in 60s without client buffering
- 1 Mbps in 8 layers of 128 Kbps
- 5% loss



# Conclusion

- A means TCP-friendly lossy streaming to TCP receivers
  - Can send to normal HTTP streaming clients
  - Through firewalls
  - Works with TCP flow control
  - Has a TCP-friendly congestion control
  - ... Several TODOs

- An API for applications with second thoughts
  - LDC has been implemented with DCCP
  - It is less natural but similarly useful with TCP
  - Especially when you have a timeout event in AIMD