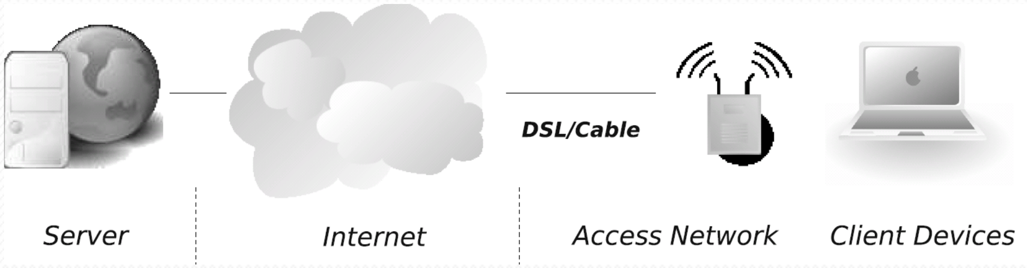
Evaluation of HTTP-based Request-Response Streams for Internet Video Streaming

Robert Kuschnig, Ingo Kofler, Hermann Hellwagner
Institute of Information Technology (ITEC)
Klagenfurt University, Austria

Use case



- Streaming video via Internet
- TCP friendliness required
- High round-trip-times (RTT)
- Available bandwidth unknown/changing
- Random packet loss in access networks

Classical TCP streaming

- Transport video data within single TCP connection
- Good performance in low-latency networks
- Performance problems on packet loss (AIMD)
- Throughput model:
  - Packet loss p: after every 1/p packets, one is lost
  - Upper bound for throughput r_{tcp}:

$$r_{tcp} = \frac{MSS}{\sqrt{p}} \cdot \frac{1}{RTT}$$

Parallel TCP-based Request-Response Streams

- Request-Response (RR): short-lived TCP connection
- Connection-less
- More reliable in error-prone networks
- May experience unfairness from infinite-source TCP flows (cf. download of large file vs. web browsing)
- Idea to aggregate multiple submissive RR streams with the same TCP-friendliness as a single TCP connection
- Introduce inter-request gap to tune TCP-friendliness

Model for RR Streams

- Upper bound of throughput for n_c parallel RR streams using chunks of size l_{ch}:

$$r_{rrsimple} = n_c \left(\frac{l_{ch}}{RTT + t_{gap}} \right)$$

- If we additionally assume to known the bottleneck link and the random packet loss on the network
- Model of throughput r_{rrloss} for n_c RR streams:

$$r_{rrloss} = n_c \left(\frac{l_{ch}}{t_{chloss} + t_{gap}} \right)$$

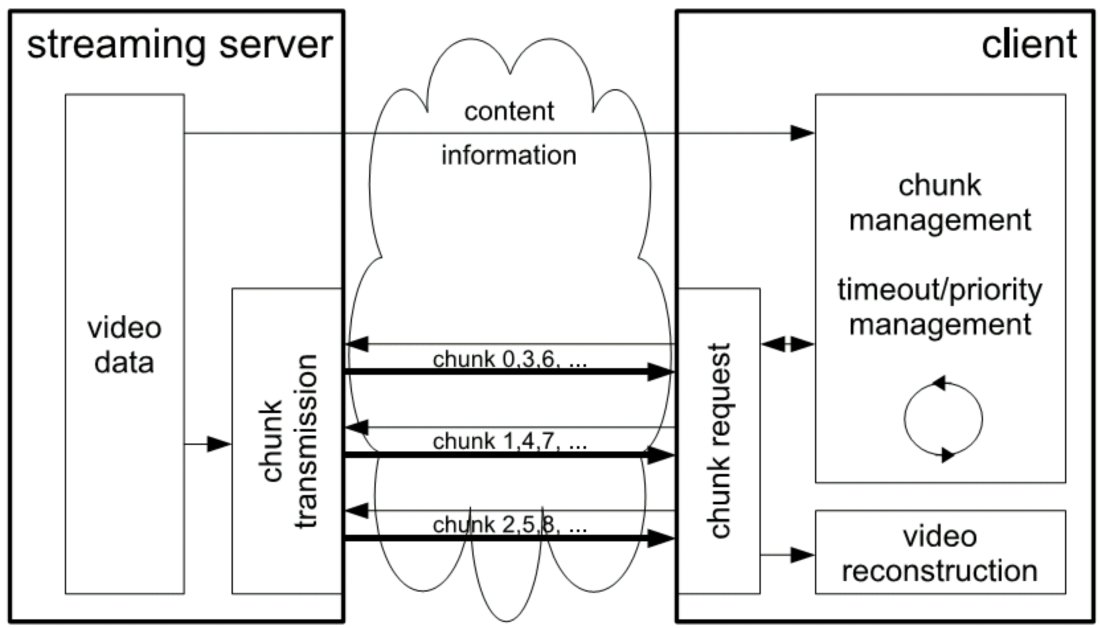
$$t_{chloss} = \lceil n_{rtloss} \rceil (RTT + t_{qavloss})$$

System & Network Parameters

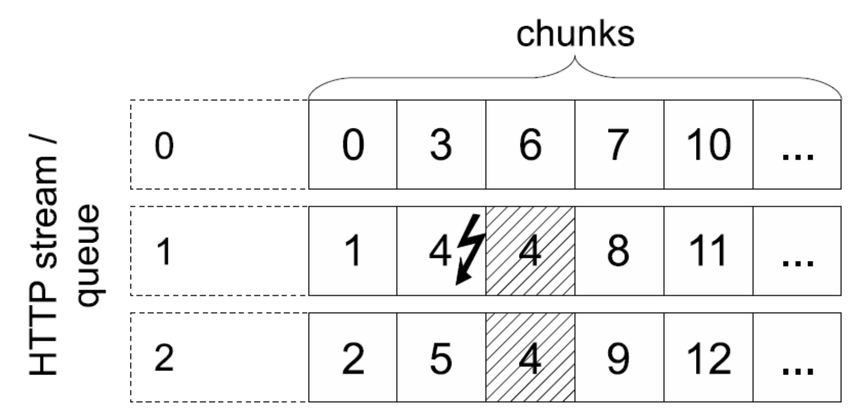
- System parameters of a RR streaming system
  - Number of parallel RR streams n_c
  - Chunk size l_{ch}
  - Inter-request gap t_{gap}
- Network parameters considered in the model
  - Bandwidth of bottleneck router BW
  - Maximum queuing delay allowed on the router t_q
  - Network Round-Trip-Time RTT
  - Random packet loss on the network

Client-driven Video Streaming based on RR Streams

- Transport based on multiple RR streams (HTTP)
- No feedback loop between client and server
- HTTP enables easy deployment
- HTTP/1.1 connection reuse
- H.264/SVC Priority Streaming (video reordering)



Timeout and Priority Management



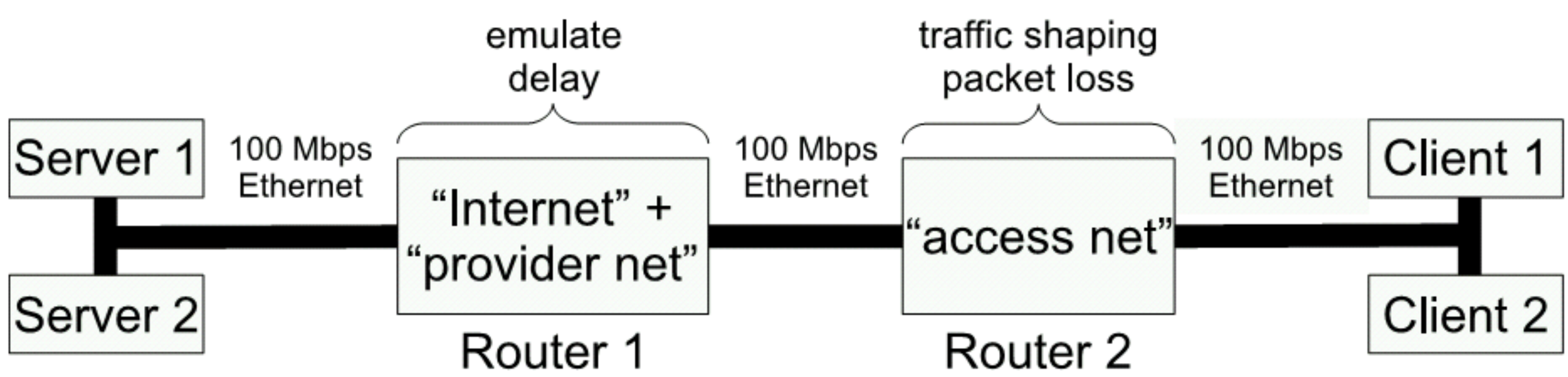
- Manage chunks in queues
- Each queue is assigned to a HTTP stream
- Timeout Management:
  - Monitor transmission time of chunks
  - If transmission is stalled, abort transmission
- Priority Management:
  - Prioritize chunks needed in the near future
  - If a chunk is stalled, it is fetched by two streams again to increase the probability of success

Evaluation of Streaming System

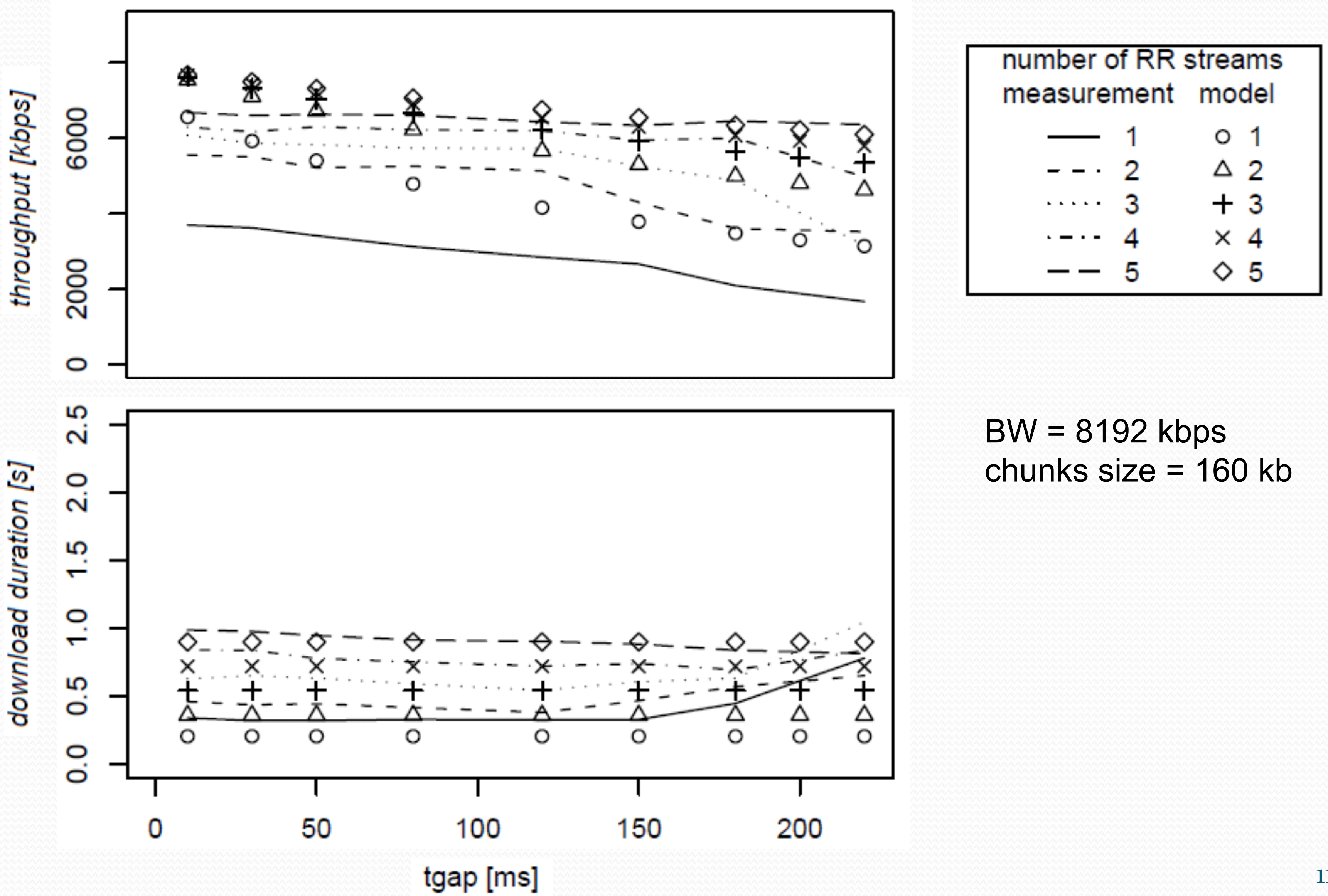
- Vary system and network parameters
- Measure:
  - Throughput
  - Download duration of single chunks
  - Fairness ratio to concurrent TCP connections
  - PSNR of received video
- Goal:
  - Gain insights on streaming performance and TCP friendliness with respect to the system parameters

Test setup

- Ubuntu with Linux kernel 2.6.27
- Network emulation with netem
- Apache web server
- Prototype software with Python

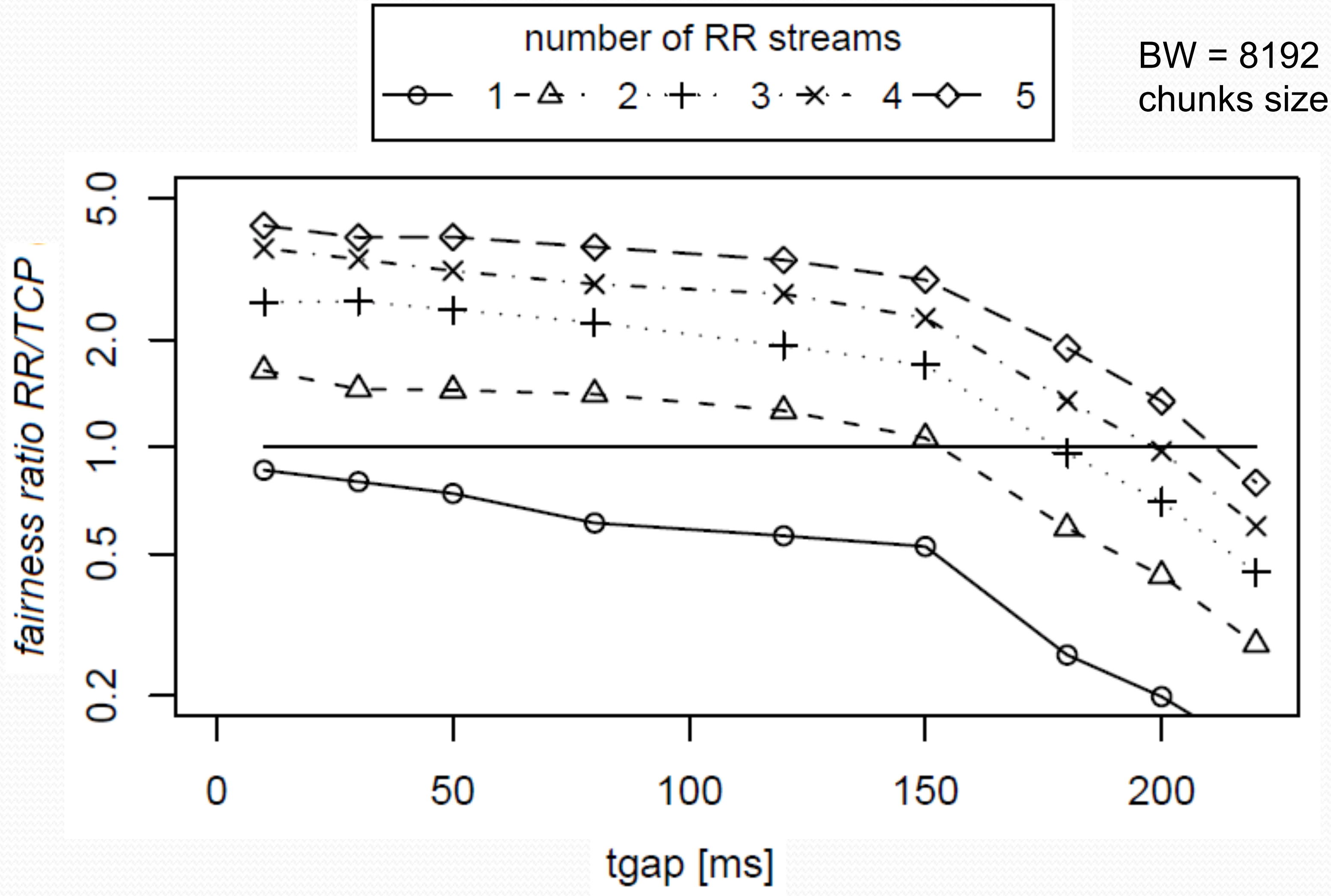


Throughput and Download Dur.



BW = 8192 kbps
chunks size = 160 kb

TCP Friendliness

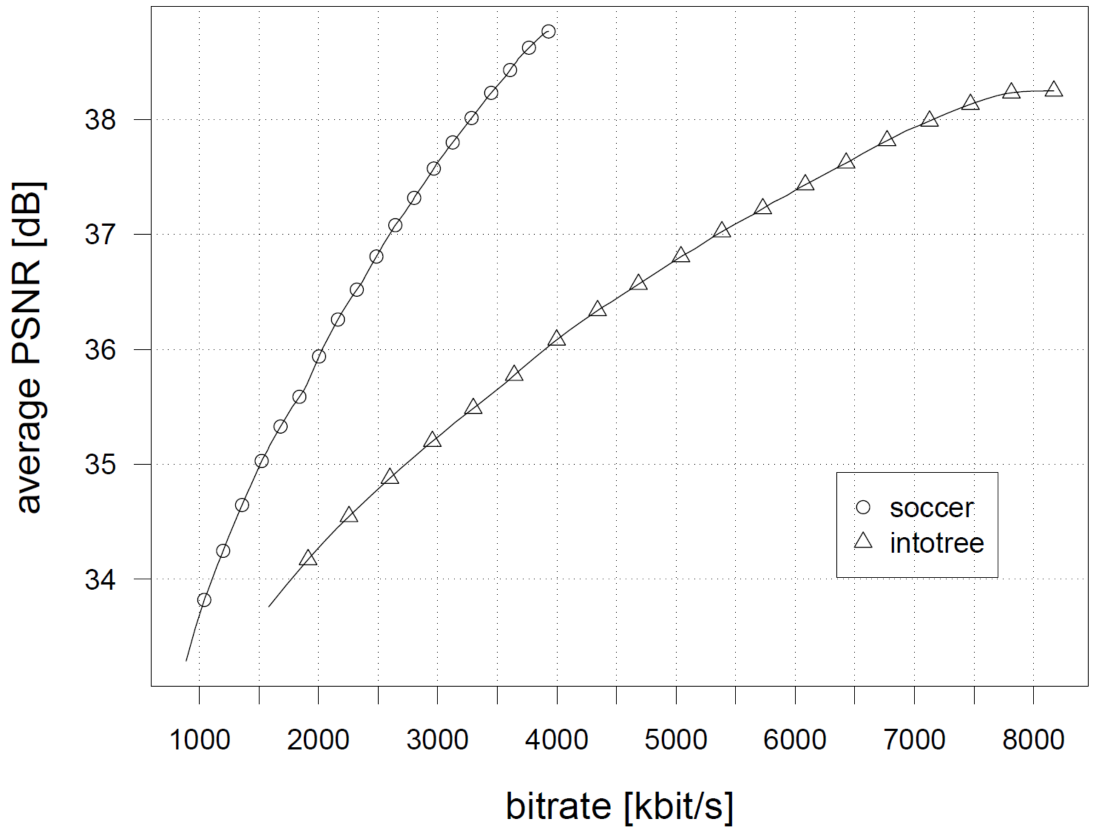


BW = 8192 kbps
chunks size = 160 kb

Test video sequences

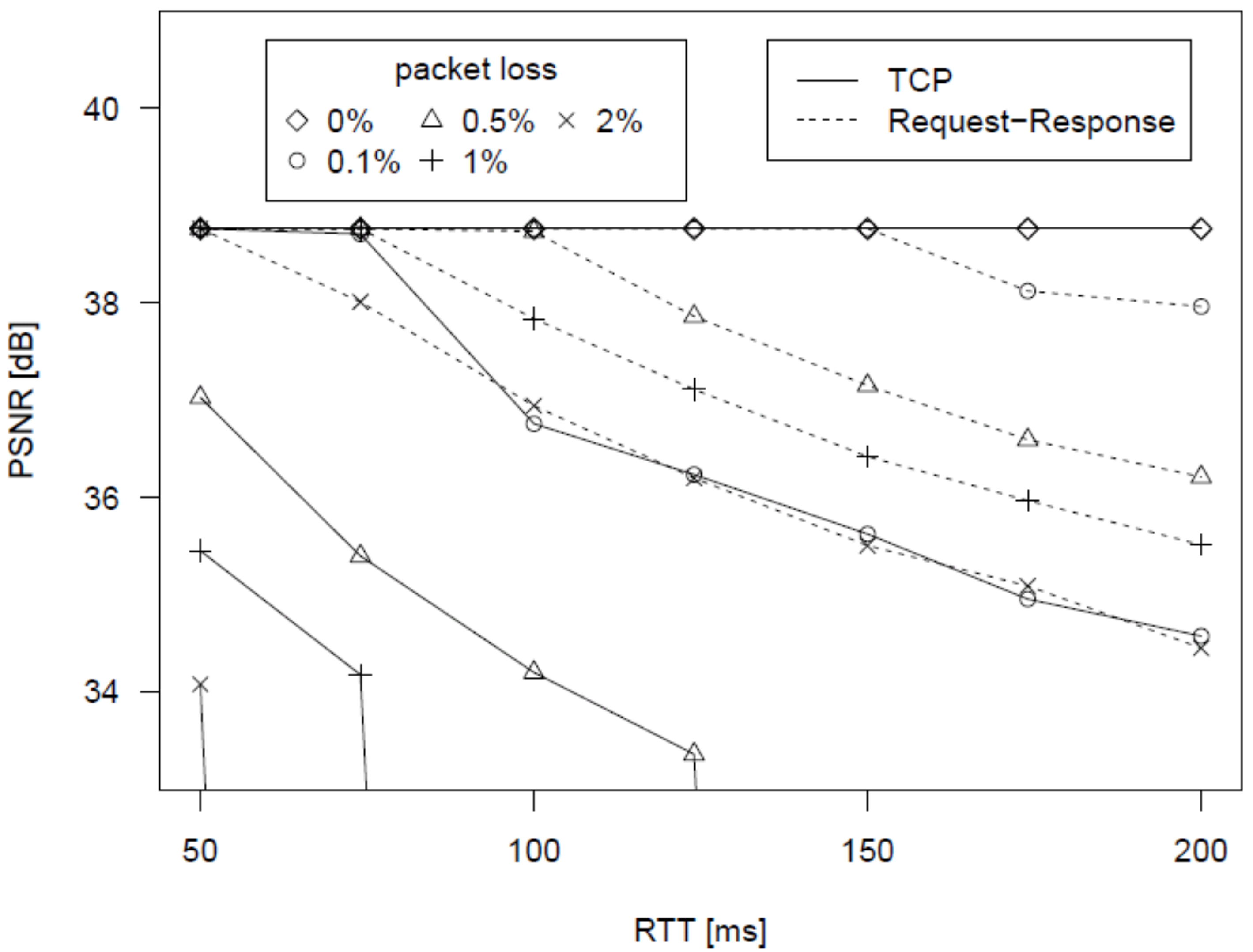
- Soccer: 4CIF@30
- In-to-tree: 720p@50

- H.264/SVC
- Single MGS layer with 4 MGS vectors
- PSNR-optimal Priority ID (PID) assignment



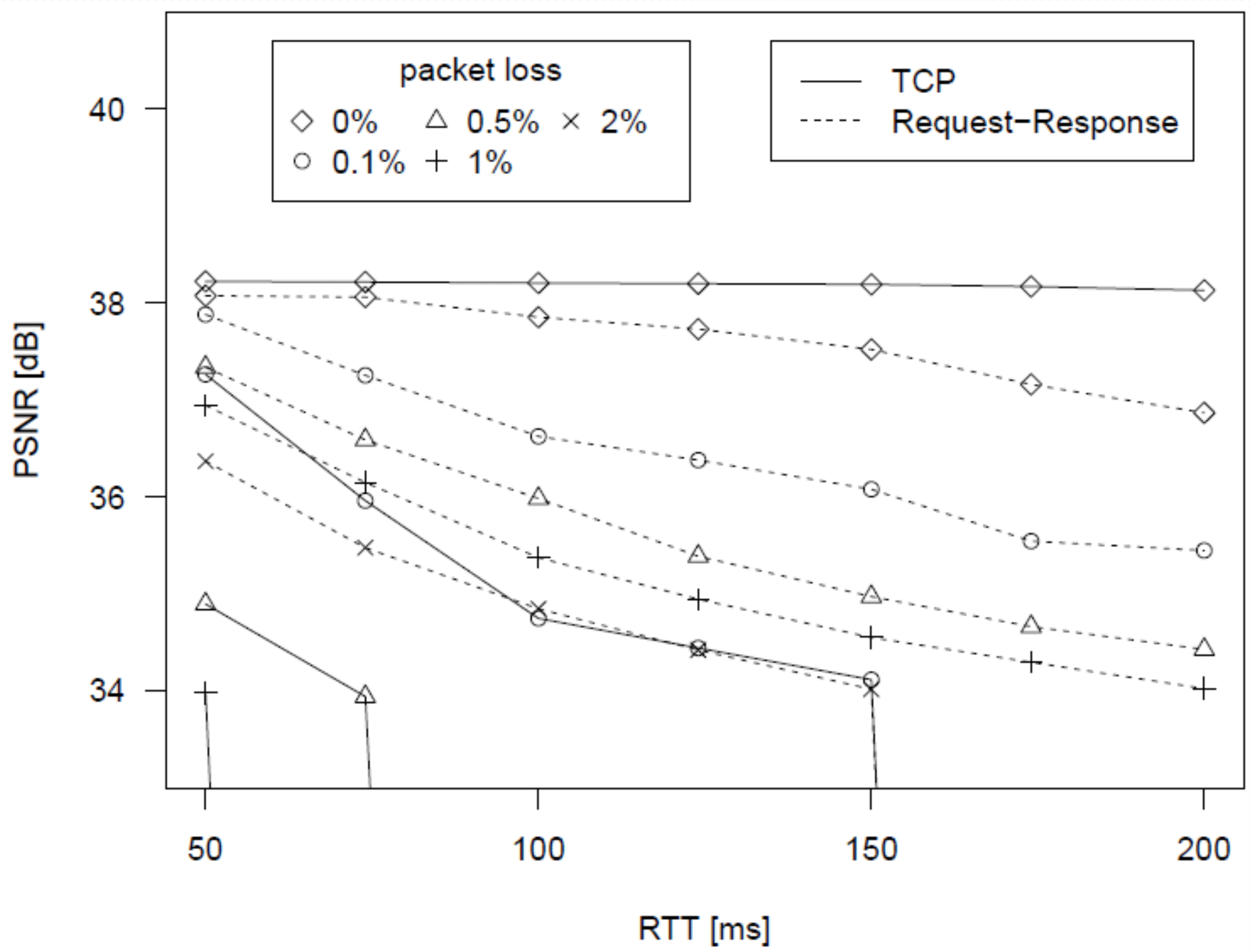
- Video is reordered before transmission according to PID (priority streaming)

Video Streaming – Soccer



BW = 8192 kbps
n_c = 5
l_{ch}= 160 kb
t_{gap} = 210 ms

Video Streaming – intotree



BW = 8192 kbps

n_c = 5

l_{ch}= 160 kb

t_{gap} = 210 ms

Conclusion

- Request-Response streams are connection-less, but more computational expensive than TCP
- A single RR-stream may not fully utilize the avail. BW
- RR-streams scale well with increasing l_{ch} or n_c
- RR-streams can achieve TCP-friendliness at good performance
- Advantage to TCP-streaming in case of packet loss