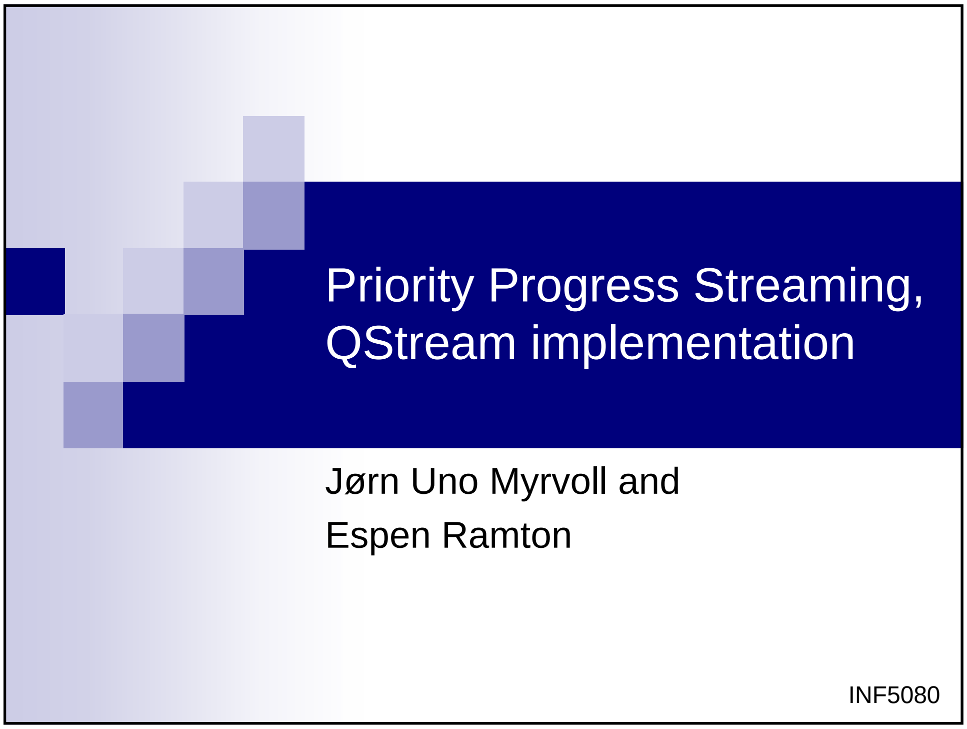
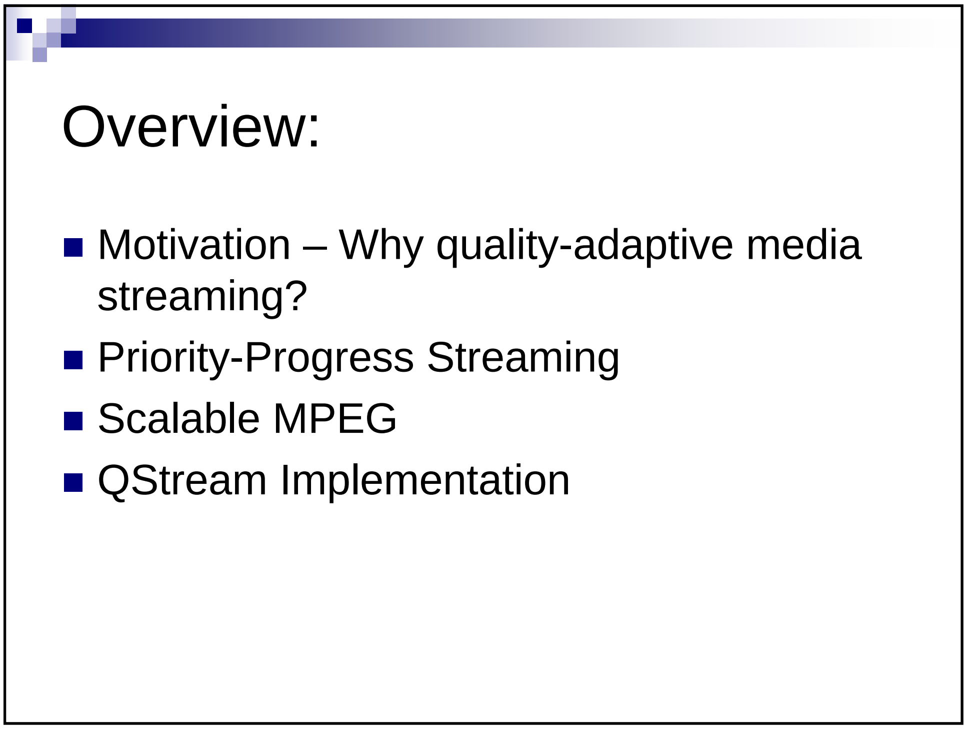


Priority Progress Streaming, QStream implementation

Jørn Uno Myrvoll and
Espen Ramton

INF5080



Overview:

- Motivation – Why quality-adaptive media streaming?
- Priority-Progress Streaming
- Scalable MPEG
- QStream Implementation

Why quality-adaptive media streaming?

- Both video bit rates and network bandwidth are variable over time.
- End users may have different needs.
  - □ Encode once, stream anywhere!
  - □ Big screen at home, mobile unit on the bus.

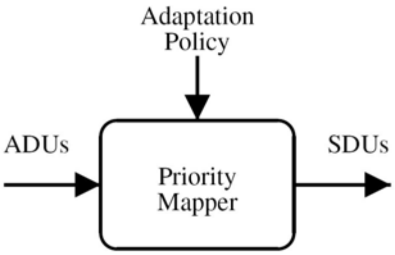
Priority Progress Streaming

Some problems that must be addressed

- Robustness
  - □ Ability to avoid interruptions in a best effort network.
- Utilization
  - □ Quality of video. Is the user experiencing the highest quality possible?
- Latency
  - □ Navigation Latency
    - Delay between user initiates a request and the result is displayed.
  - □ Communication latency
    - Delay between video entering the streaming process on the server side and being displayed on the client side.
- Consistency
  - □ Frequency and magnitude of changes in video.
  - □ Fewer and smaller changes leads to better experience of the video.

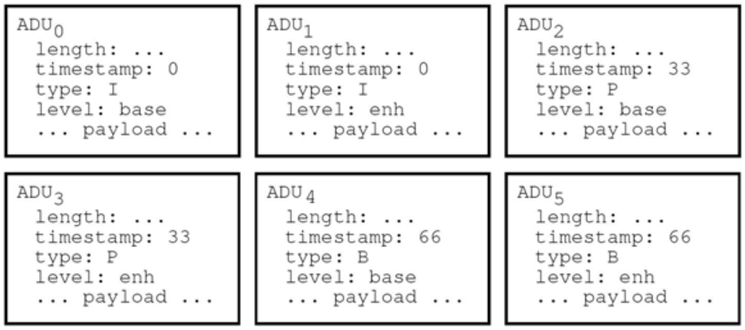
Priority Mapping

- Divides timeline in stream into mapping windows
- Prioritizes the ADUs in each window separately
- Applies adaptation policy
- Outputs prioritized sequence of streaming data units.



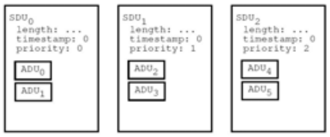
ADUs – Application Data Units

- ADU_1=>ADU_0, ADU_3=>ADU_2, ADU_5=>ADU_4
- ADU_4=>ADU_2=>ADU_0

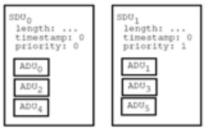


SDUs – Streaming Data Units

- Two different mappings of the six ADUs.
- a) Grouped for frame drop
- b) Grouped for spatial drop.

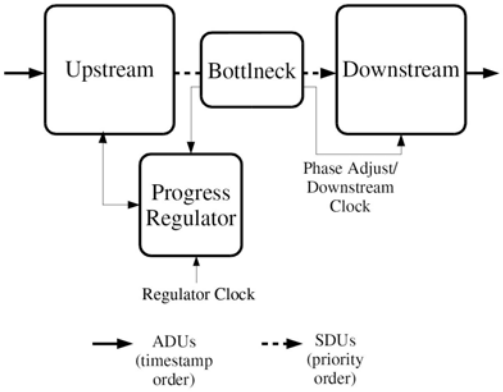


(a) Frame drop



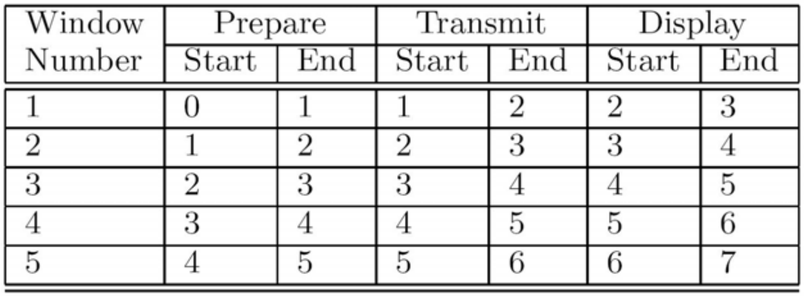
Priority Progress Streaming

- The PPS algorithm subdivides the timeline into intervals using the SDU timestamps.
- These intervals are called adaptation windows
- SDUs are sent in priority order.
- When the regulator clock advances to next window, unsent SDUs from the old window are discarded.



Priority Progress Example

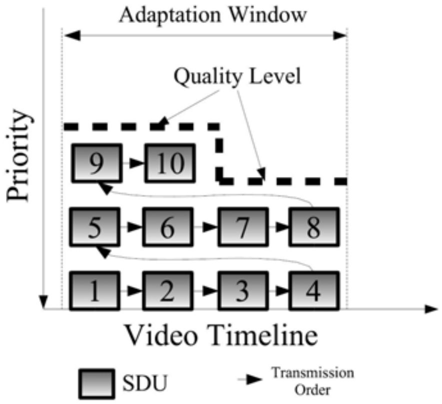
- An entire window must be transmitted before it is displayed.



Window Number	Prepare		Transmit		Display	
	Start	End	Start	End	Start	End
1	0	1	1	2	2	3
2	1	2	2	3	3	4
3	2	3	3	4	4	5
4	3	4	4	5	5	6
5	4	5	5	6	6	7

Adaption Window Transmission

- Small windows gives shorter startup delays
- Larger windows gives fewer changes in quality. (Maximum two quality levels within one window.)

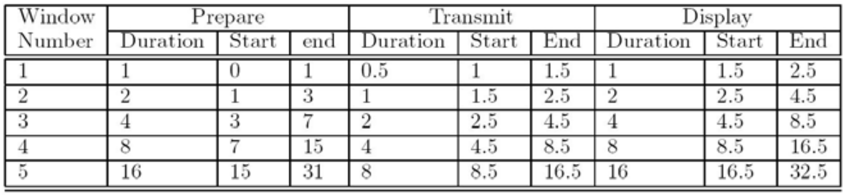


Window Scaling

- Starts with small adaptation windows.
- Gives short startup time.
- Growing window size gives less changes in quality.

Window Scaling Example

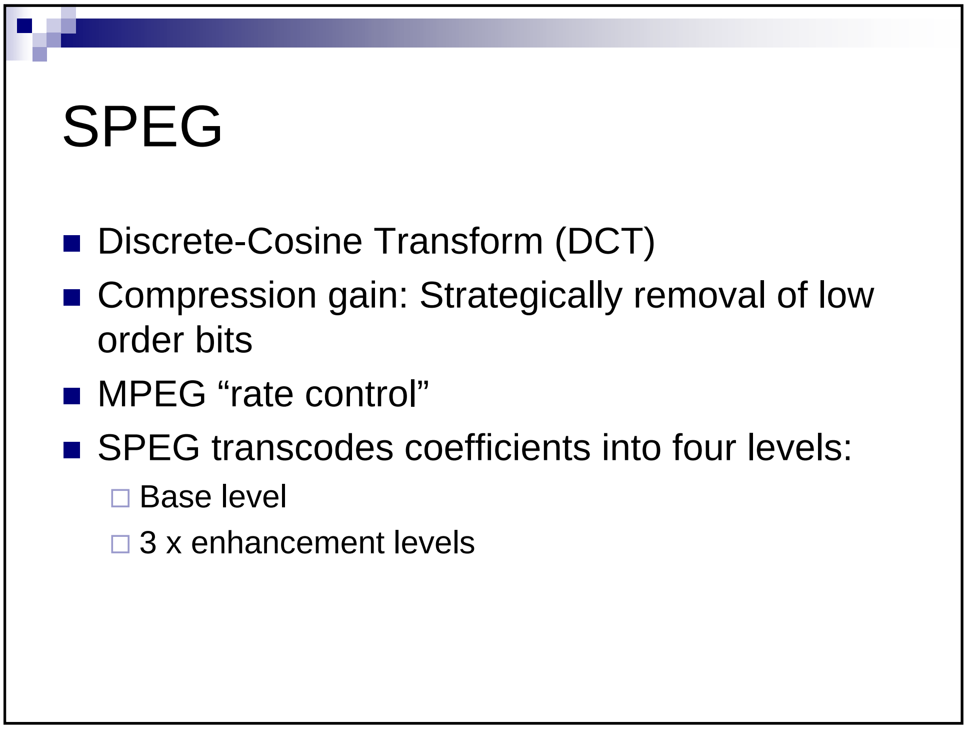
- An example of growing adaptation window.



Window Number	Prepare			Transmit			Display		
	Duration	Start	end	Duration	Start	End	Duration	Start	End
1	1	0	1	0.5	1	1.5	1	1.5	2.5
2	2	1	3	1	1.5	2.5	2	2.5	4.5
3	4	3	7	2	2.5	4.5	4	4.5	8.5
4	8	7	15	4	4.5	8.5	8	8.5	16.5
5	16	15	31	8	8.5	16.5	16	16.5	32.5

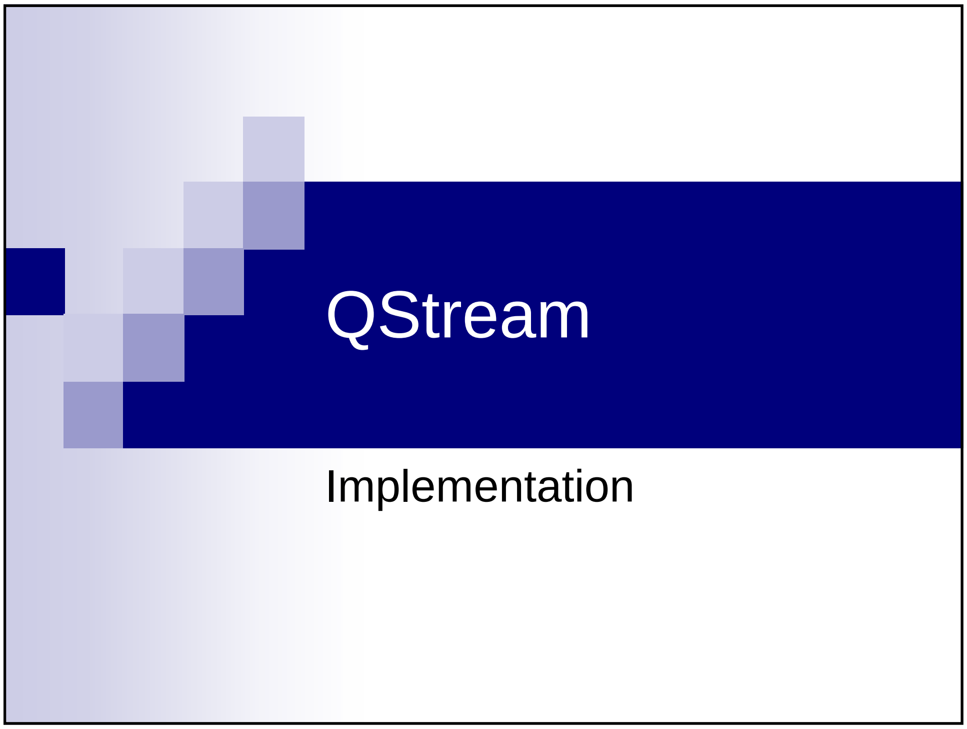
SPEG

Scaleable MPEG



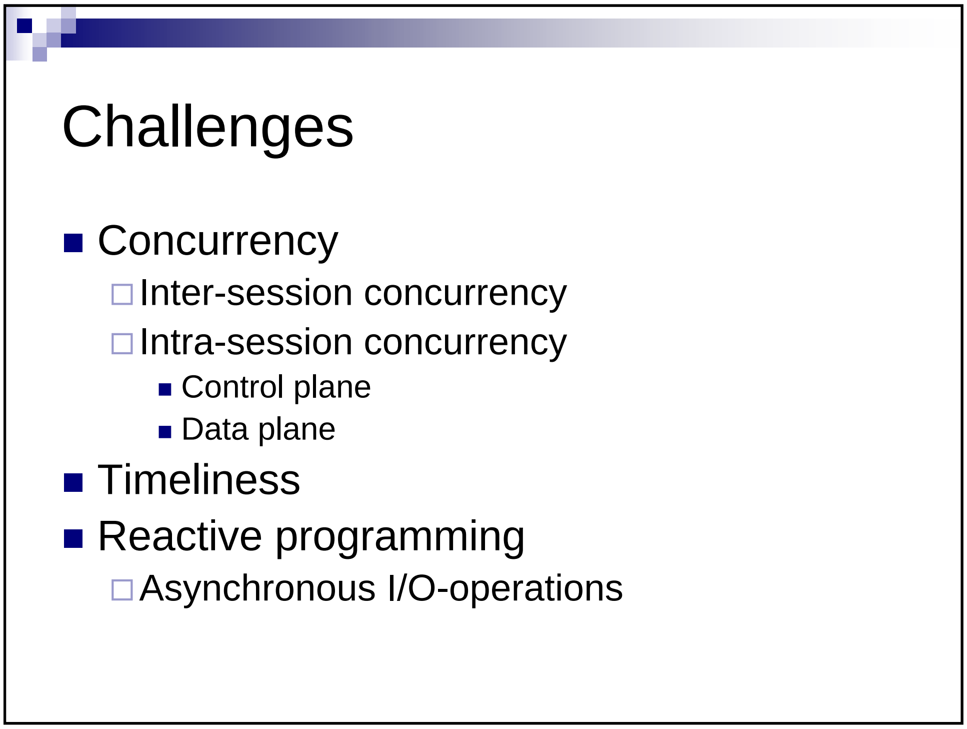
SPEG

- Discrete-Cosine Transform (DCT)
- Compression gain: Strategically removal of low order bits
- MPEG “rate control”
- SPEG transcodes coefficients into four levels:
  - □ Base level
  - □ 3 x enhancement levels



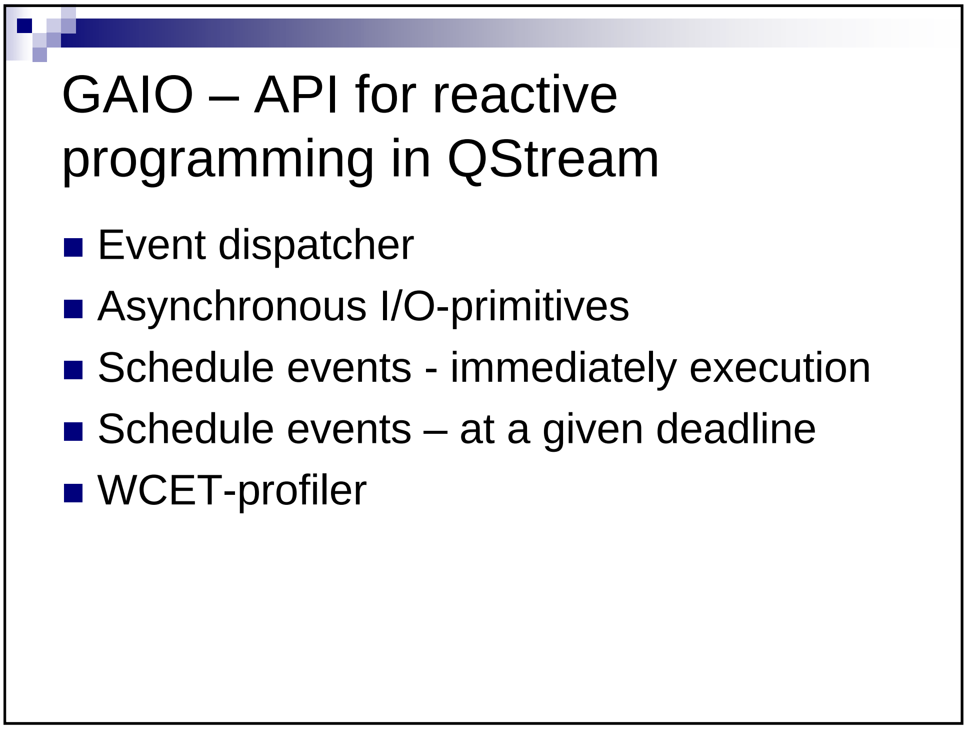
QStream

Implementation



Challenges

- Concurrency
  - □ Inter-session concurrency
  - □ Intra-session concurrency
    - Control plane
    - Data plane
- Timeliness
- Reactive programming
  - □ Asynchronous I/O-operations



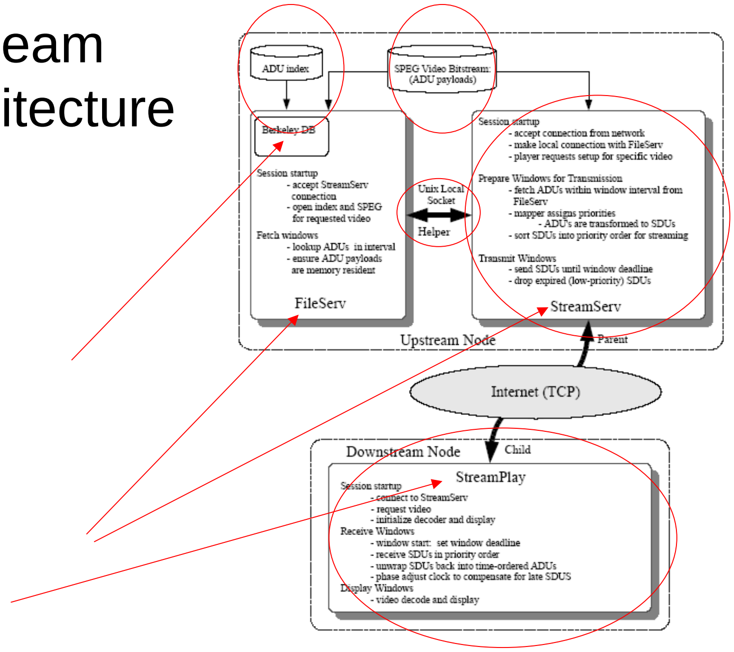
GAIO – API for reactive programming in QStream

- Event dispatcher
- Asynchronous I/O-primitives
- Schedule events - immediately execution
- Schedule events – at a given deadline
- WCET-profiler

Quasar Streaming Framework (QSF)

- Primitives for establishing network connections and message passing
- Logging and tracing
- A primitive to enable real-time OS scheduling

QStream Architecture



ADU index

SPEG Video Bitstream: (ADU payloads)

Berkeley DB

Session startup
- accept StreamServ connection
- open index and SPEG for requested video

Fetch windows
- lookup ADUs in interval
- ensure ADU payloads are memory resident

FileServ

Unix Local Socket

Helper

Session startup
- accept connection from network
- make local connection with FileServ
- player requests setup for specific video

Prepare Windows for Transmission
- fetch ADUs within window interval from FileServ
- mapper assigns priorities
  - ADU's are transformed to SDUs
- sort SDUs into priority order for streaming

Transmit Windows
- send SDUs until window deadline
- drop expired (low-priority) SDUs

StreamServ

Upstream Node

Parent

Internet (TCP)

Child

Downstream Node

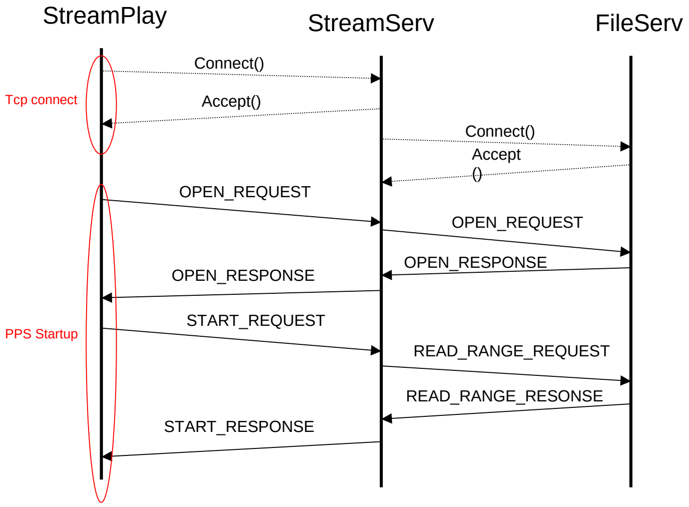
StreamPlay

Session startup
- connect to StreamServ
- request video
- initialize decoder and display

Receive Windows
- window start: set window deadline
- receive SDUs in priority order
- unwrap SDUs back into time-ordered ADUs
- phase adjust clock to compensate for late SDUS

Display Windows
- video decode and display

PPS messages



StreamPlay

StreamServ

FileServ

Connect()

Accept()

Tcp connect

Connect()

Accept
()

OPEN_REQUEST

OPEN_REQUEST

OPEN_RESPONSE

OPEN_RESPONSE

START_REQUEST

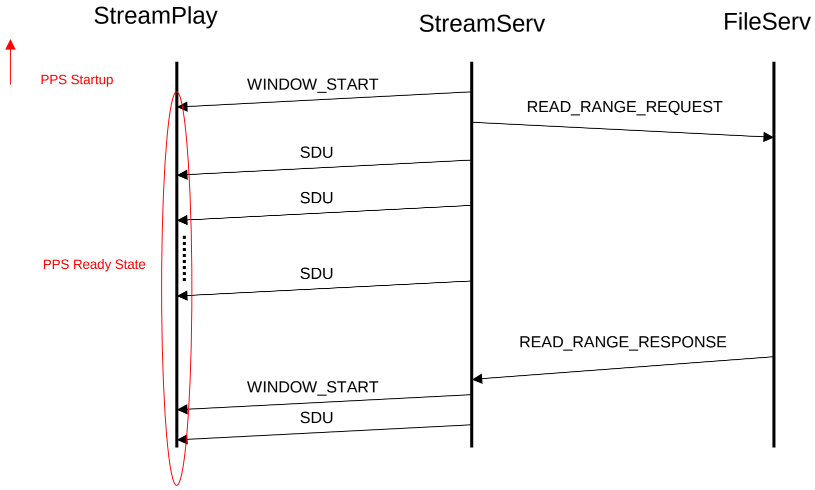
PPS Startup

READ_RANGE_REQUEST

READ_RANGE_RESONSE

START_RESPONSE

PPS Steady State



StreamPlay

StreamServ

FileServ

PPS Startup

WINDOW_START

READ_RANGE_REQUEST

SDU

SDU

PPS Ready State

SDU

READ_RANGE_RESPONSE

WINDOW_START

SDU

References

- C. Krasic. A Framework for Quality-Adaptive Media Streaming: Encode Once – Stream Anywhere
- C. Krasic, J. Walpole, W. Feng. Quality-Adaptive Media Streaming by Priority Drop
- C. Krasic, K. Li, J. Walpole. The Case for Streaming Multimedia with TCP
- C. Krasic, J. Walpole. Priority-Progress Sreaming for Quality-Adaptive Multimedia
- The QStream Framework – http://qstream.org