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.
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**

- ADU₁ => ADU₂, ADU₃ => ADU₂, ADU₅ => ADU₄
- ADU₄ => ADU₂ => ADU₀

**SDUs – Streaming Data Units**

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

<table>
<thead>
<tr>
<th>Window Number</th>
<th>Prepare Start</th>
<th>Prepare End</th>
<th>Transmit Start</th>
<th>Transmit End</th>
<th>Display Start</th>
<th>Display End</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>6</td>
<td>7</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Window Number</th>
<th>Prepare</th>
<th>Transmit</th>
<th>Display</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Duration Start End</td>
<td>Duration Start End</td>
<td>Duration Start End</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0 1 0.5</td>
<td>1 1.5 1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>1 3 1</td>
<td>1.5 2.5 2</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>3 7 2</td>
<td>2.5 4.5 4</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>7 15 4</td>
<td>4.5 8.5 8</td>
</tr>
<tr>
<td>5</td>
<td>16</td>
<td>15 31 8</td>
<td>8.5 16.5 16</td>
</tr>
</tbody>
</table>
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
PPS messages

StreamPlay

Tcp connect

Accept()

OPEN_REQUEST

OPEN_RESPONSE

START_REQUEST

START_RESPONSE

StreamServ

Connect()

Accept()

OPEN_REQUEST

OPEN_RESPONSE

READ_RANGE_REQUEST

READ_RANGE_RESPONSE

FileServ

Connect()

Accept()

OPEN_REQUEST

OPEN_RESPONSE

READ_RANGE_REQUEST

READ_RANGE_RESPONSE

PPS Steady State

StreamPlay

WINDOW_START

SDU

SDU

SDU

PPS Startup

PPS Ready State

STREAM_START

READ_RANGE_REQUEST

READ_RANGE_RESPONSE

FileServ

READ_RANGE_REQUEST

READ_RANGE_RESPONSE
References

- 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 Streaming for Quality-Adaptive Multimedia
- The QStream Framework – http://qstream.org