Motivation

Emerging mobile networks and devices present unique challenges for high quality multimedia delivery

**Challenge: Heterogeneity**
- Differing access technologies
- Differing network characteristics
- Differing device capabilities
- Java performance issues

**Challenge: Network Congestion**
- Shared network scenarios
- Unpredictable join / leave
- Fluctuating network load

**Challenge: Radio Access**
- Signal interference
- Propagation problems
- Uneven network coverage
- Network handoff
Solution

*Seamless* automatic media adaptation allows sophisticated responses to mobile delivery challenges

**Automatic Media Adaptation**
- User QoS Policy
- Monitoring Results
- Movement Detection
- Resource Availability

**Seamless Chain Switching**
- Media Processing Chains consist of Codecs, Filter, Packetizer etc.
- Switching between Chains usually imposes additional degradations
- Our approach: *Seamless switching*
Seamless vs. Traditional Adaptation

Traditional Adaptation Approach

1. Chain teardown
   - Data loss during chain teardown & rebuilding

2. Chain rebuild
   - Long adaptation time interrupts stream

Seamless Adaptation Approach

1. Parallel chain build
   - No loss during chain reconstruction

2. RTP reconnect
   - Reduced adaptation time

Advantages:
- No loss during chain reconstruction
- Reduced adaptation time

Problems:
- Data loss during chain teardown & rebuilding
- Long adaptation time interrupts stream
Traditional Adaptation

Conventional media adaptation through sequential chain rebuilding

Loss Time \[ t_\lambda = \left[ \frac{\delta_1 + \phi_1 + \sigma_2}{\Delta t} \right] \cdot \Delta t \]

Lost Frames \[ \lambda = \left[ \frac{\delta_1 + \phi_1 + \sigma_2}{\Delta t} \right] \]

Gap Time \[ t_\gamma = t_\lambda + (\delta_2 - \delta_1) \]

\( \sigma_2 \): New Chain Set-up time
\( \phi_1 \): Old Chain Tear-down time
\( \delta_1, \delta_2 \): Intrinsic Chain Delays
Seamless Adaptation

Loss Time \( t_\lambda = 0 \)
Lost Frames \( \lambda = 0 \)

Gap Time \( t_\gamma = \delta_2 - \delta_1 \)  
Just codec-intrinsic delay! 
Overall adaptation time also reduced!

No loss at all!
Demo Scenario

- RTP Streaming
- Adaptation based on RTCP feedback

Adaptation Algorithm

Video Server

Nistnet Emulator

Hotspot Technologies

GSM/GPRS/UMTS

Adaptive Mobile Device

RTCP Monitor Event

User Policy

User changed his Policy

Waiting for events

Calculate smoothed packet loss value and estimate NETWORK STATUS Actual adaptation values in the user specified MinMax intervals?

Yes

Update user policy parameters

No

ADAPT VIDEO STREAM WITH USER min OR max VALUES

NETWORK STATUS = CONGESTED

SET NEW ACCESS NETWORK

ADAPT VIDEO STREAM WITH PRE-DREFINED ADAPTATION VALUES

SET ACTUAL Q and framerate value

INCREASE QUALITY or FRAME RATE

Estimate USERS PREFERENCES

DECREASE QUALITY or FRAME RATE

Actual adaptation values in the user specified MinMax intervals?

ADAPT VIDEO STREAM TONEW VALUE

ADAPT VIDEO STREAM TONEW VALUE

Yes

Set actual Q and framerate value

Yes

Set actual Q and framerate value
Prototype Implementation

- Java based adaptive multimedia API
- Based on Java Media Framework (JMF)
- Pluggable Adaptation Modules (Frame Filter, Quality, Datarate, Codec Switch)
- Reference implementation of Seamless Codec Switcher as a plug-in module
- Enhanced high-speed switching datasource to enable on-the-fly format switching to JMF RTP mechanisms.
- MPEG-4 Packetizer / Depacketizer / Frame Filter (DivX4.12)
- Optimized Resource Management (multi-threading, event-driven)
- JMF Native Encoder extensions to allow DivX on-the-fly parameter changes
- Transcoding Support
- NistNet Emulator remote control (Java/C++)
Preliminary Results

Sequential Mode
- CaptureDataSource reading audio from microphone

<table>
<thead>
<tr>
<th>Loss Time [ms]</th>
<th>DVI</th>
<th>GSM</th>
<th>G.711</th>
<th>G.723</th>
<th>MP3</th>
</tr>
</thead>
<tbody>
<tr>
<td>DVI</td>
<td>174.4</td>
<td>586.8</td>
<td>667.0</td>
<td>583.0</td>
<td>140.0</td>
</tr>
<tr>
<td>GSM</td>
<td>178.2</td>
<td>586.6</td>
<td>600.6</td>
<td>589.0</td>
<td>140.4</td>
</tr>
<tr>
<td>G.711</td>
<td>188.2</td>
<td>598.8</td>
<td>678.8</td>
<td>556.6</td>
<td>142.2</td>
</tr>
<tr>
<td>G.723</td>
<td>182.4</td>
<td>550.4</td>
<td>671.0</td>
<td>588.8</td>
<td>148.8</td>
</tr>
<tr>
<td>MP3</td>
<td>234.2</td>
<td>585.0</td>
<td>654.8</td>
<td>603.0</td>
<td>148.2</td>
</tr>
</tbody>
</table>

- Results with FileDataSources are even worse (worst case 3 seconds!)

Seamless Mode
- Loss Time is 0ms
- Chain Switching Time below 1ms (measurement accuracy)

Proposed Approach improves Codec Changes significantly
- Zero Loss at Sender (verified with Packet Sniffer)
- Reduced Overall Adaptation Time
- Codec Independent
- Media Type Independent

Average of 10 adaptation cycles (Athlon Thunderbird 750MHz, 256MByte Memory, Windows 2000)