Network-Integrated Multimedia Middleware
KDE aKademy 2004

Marco Lohse, Michael Repplinger, Philipp Slusallek

Computer Graphics Lab, Saarland University, Germany
Motivation

- PC-centric approach (back in 2000, today?)
  - Stand-alone multimedia-PC
  - Streaming predefined content from the Internet
  - Many other multimedia devices

⇒ Unexploited networking capabilities
Network-Integrated Multimedia Middleware (NMM)

- GNU/Linux (x86/ARM), LGPL/GPL, C++
- http://www.networkmultimedia.org
NMM

- Network-Integrated
  - Extend cooperation and control to network
- Multimedia (for GNU/Linux)
  - Uniform access to available drivers and libraries
- Middleware
  - Layer between applications and distributed systems
Application Scenarios

- Access distributed resources
  - Distributed transcoding
  - Watch TV using remote TV receiver
  - Control remote TV receiver

Stationary system

Processing power
I/O capabilities
TV receiver

Distributed processing power
Distributed I/O capabilities

KDE aKademy 2004 - NMM - p.5/46
Collaborative Scenarios

- Number of users simultaneously enjoys the same (or similar) content using different devices
- “Drive-in cinema”
  - Watch video of DVD on large screen
  - Synchronized playback on different PDAs

Mobile system 1
- Audio track 0

Mobile system 2
- Audio track 1

Stationary system
- Video
- DVD playback
Dynamic Scenarios

- User and device mobility
- Use capabilities of surrounding environment
- Handover of media playback
  - E.g. use high-fidelity audio output

Mobile system
- MP3 playback

Stationary system
Dynamic Scenarios

- User and device mobility
- Use capabilities of surrounding environment
- Handover of media playback
  - E.g. use high-fidelity audio output
Beyond Client/Server

- Traditional client/server streaming $\Rightarrow$ Black-box
  - Locate server? Activate server? Provided functionality? Connection setup? Streamed data format? Control server, e.g. switch the TV channel? Complex application scenarios, e.g. transcode content? Distribute workload? Synchronization? Shared applications?
Beyond Client/Server

- Traditional client/server streaming ⇒ Black-box
  - Locate server? Activate server? Provided functionality? Connection setup? Streamed data format? Control server, e.g. switch the TV channel? Complex application scenarios, e.g. transcode content? Distribute workload? Synchronization? Shared applications?

⇒ It’s not a question of “if” – it’s a question of “how”
⇒ Generally solve challenges within middleware
⇒ Provide suitable abstractions and programming model
⇒ Easy, efficient development of distributed applications
Overview

• Motivation
• Architecture of NMM
• Developing plug-ins for NMM
• Basic middleware services
  • Registry service, Synchronization, GraphBuilder
• Distributed multimedia application
  • helloworld, clic, Multimedia-Box, ...
• Advanced Middleware Services
  • Application sharing, seamless handover
• Summary, future work & discussion
Flow Graph

- Nodes as smallest processing unit
- Jacks to connect nodes
- Formats to type connections
- Interfaces to control objects
- Messages to forward media data or control information

Generic Read Node

Output Jack

Input Jack

audio/mpeg

MPEG Audio Decode Node

Output Jack

Input Jack

audio/raw

Playback Node

Application
Flow Graph

- Nodes as smallest processing unit
- Jacks to connect nodes
- Formats to type connections
- Interfaces to control objects
- Messages to forward media data or control information
Flow Graph

- Nodes as smallest processing unit
- Jacks to connect nodes
- Formats to type connections
- Interfaces to control objects
- Messages to forward media data or control information
Flow Graph

- Nodes as smallest processing unit
- Jacks to connect nodes
- Formats to type connections
- Interfaces to control objects
- Messages to forward media data or control information
Messaging System

- **Message**
  - Header information such as timestamps
- **Buffer**
  - Multimedia data
  - Efficiently managed by buffer pools
- **Composite event (CEvent)**
  - Stores list of events
- **Event**
  - Key, list of values, return value, exception
  - Represents single method call for dispatcher
  - Both, out-of-band and instream interaction
Flow Graph

- NMM flow graphs are distributed
- Local and remote nodes can be controlled and integrated into a common flow graph
- Transparent for developers, no overhead for all co-located parts of flow graph

Diagram:
- Generic Read Node
- Output Jack
- Input Jack
- MPEG Audio Decode Node
- Output Jack
- Input Jack
- Playback Node
- Application
Flow Graph

- NMM flow graphs are distributed
- Local and remote nodes can be controlled and integrated into a common flow graph
- Transparent for developers, no overhead for all co-located parts of flow graph
Interfaces

• NMM Interface Definition Language (NMM IDL)
  • Similar to CORBA IDL
  • Interface class and implementation skeleton
  • Supports inheritance, exceptions, arbitrary data types, state machine, ...
  • Both, out-of-band and instream interaction
module NMM {
    interface IFileHandler {
        void setFilename(in string name);

        Result endTrack() instream;
    }
}
Nodes

• Different types of nodes & generic base classes

- Source Node
- Filter / Converter Node
- Mux Node
- Demux Node
- Mux Demux Node
- Sink Node

• State machine controls life-cycle

CONSTRUCTED
INITIALIZED
OUTPUT_INITIALIZED
ACTIVATED
STARTED

Constructor();
init();
doInit();
initOutput();
doInitOutput();
activate();
doActivate();
start();
doStart();
deinit();
doDeInit();
deinitOutput();
doDeInitOutput();
deactivate();
doDeactivate();
stop();
doStop();
60+ Plug-ins

- **Source**
  - CDDA, DVD/menus, WinTV PVR, KFIR, DVB, Firewire, VISCA cameras, audio devices, WAV, AVI
- **Encoder/decoder/converter/filter**
  - MPEG audio, MPEG video, DivX, Ogg/Vorbis, SPU, PNG, JPEG, RTJPEG
  - Color space, video scaler, on-screen menus, deinterlacer
- **Multiplexer/demultiplexer**
  - AVI, MPEG, OGM
- **Sink**
  - Video output via X or OpenGL, audio devices, WAV, AVI
Communication Channels

Application

Interface

Node

Output Jack

Input Jack

Communication Channel

Network

Buffer

Composite event

Event

KDE aKademy 2004 - NMM - p.17/46
Communication Channels

- Serialization and transport of objects
  - E.g. Buffers, composite events, events
- Scalable transparency
  - Automatic setup vs. manual configuration
- Serialization strategies
  - Magic number, XML, ...
- Transport strategies
  - Pointer forwarding, TCP, UDP, RTP, ...
- Local optimization
  - Pointer forwarding or elimination of communication channel
Developing Plug-ins

- Specification
  - Supported functionality? Features?
- Granularity and base classes
  - How many nodes? What kind of node?
- Implementation of (some) state transitions
  - doInit() : generally supported formats
  - doInitOutput() : currently supported formats
- Definition and implementation of interfaces
  - Out-of-band and instream interaction
- Implementation of processBuffer()
- Register node with registry service
processBuffer()

Message* MyConverterNode::processBuffer(Buffer * in_buffer )
{
   // get new buffer with defined size
   Buffer* out_buffer = getNewBuffer( out_buffer_size ) ;
   // get data of buffer
   char* p = in_buffer -> getData();
   // some code ...

   // release in_buffer since it is no longer needed
   in_buffer -> release();
   return out_buffer ;
}
processBuffer()

Message* MySinkNode::processBuffer(Buffer * in_buffer )
{
    // release in_buffer since it is no longer needed
    in_buffer -> release();

    // create composite event including single event
    CEvent* cevent = new CEvent(IExample::create_foo());
    // set direction to UPSTREAM
    cevent -> setDirection(Message::UPSTREAM);

    return cevent;
}
Event Handling

- Automatic registration and dispatching due to implementation of interface
  - Out-of-band and instream interaction

```c++
// Implementation of IFileHandlerImpl::endTrack()
Result MySinkNode::endTrack()
{
    // handle endTrack, e.g. close file
    return SUCCESS;
}
```
Developing Plug-ins

- Modify, add, delete, replace events
- Handling multiple input and output jacks
- Handling arbitrary patterns of incoming and outgoing messages
  - Upstream and downstream traffic
  - working-flag and producing-flag
- ...

⇒ See "Developing Plug-ins for NMM"
// MP3 player: the NMM application for the example
NMMApplication* app =
    ProxyApplication::getApplication(argc, argv);

// create the node descriptions
NodeDescription readfile("GenericReadNode");
NodeDescription decoder("MPEGAudioDecodeNode");
NodeDescription audioplay("PlaybackNode");

// create a query for a flow graph
GraphDescription graph;
graph.addEdges(&readfile, &decoder, &audioplay);
// request complete graph from registry
ClientRegistry & registry = app->getRegistry();
registry.requestGraph(graph);

// set the filename by requesting appropriate interface
INode* inode = graph.getInode(readfile);

IFileHandler_var filehandler (inode->getParentObject() 
   ->getCheckedInterface<IFileHandler>();

filehandler->setFilename(argv[1]);
// realize and start graph
graph.realizeGraph();
graph.startGraph();

// wait ...

// stop complete graph
graph.stopGraph();

// release complete graph
ergyistry.releaseGraph(graph);
// Distributed MP3 player:
// decoder and audio sink on remote host argv[2]
NodeDescription readfile ("GenericReadNode");
NodeDescription decoder("MPEGAudioDecodeNode");
NodeDescription audioplay("PlaybackNode");

// NEW: set location for decoder and audio sink
dercoder.setLocation(argv[2]);
audioplay.setLocation(argv[2]);

GraphDescription graph;
graph.addEdges(&readfile, &decoder, &audioplay);
ClientRegistry & registry = app->getRegistry();

// NEW: add remote host to registry
// (. / serverregistry is running on remote host )
registry.addRegistry(TCPAddress(argv[2], ClientRegistry::default_port), argv[2]);

// same code as before ...
registry.requestGraph(graph);

graph.realizeGraph();
graph.startGraph();
Further Options

- Node descriptions
  - Types, I/O formats, interfaces, sharing policy
- Manual creation of flow graphs
  - `connect(node1, node2);`
- Configuration of communication channels
  - `c_connect(node1, node2);`
- Listener notification
- ...

⇒ See “Hello World! Welcome to NMM Application Development :)”
Registry Service

- Administrates locally available nodes
- Queries
  - Single (subset) of a node description, or
  - Graph description
- Peer-to-peer approach for distributed flow graphs
Registry Service

- Administrates locally available nodes
- Queries
  - Single (subset) of a node description, or
  - Graph description
- Peer-to-peer approach for distributed flow graphs

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Interfaces</th>
<th>Input format</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generic</td>
<td>SOURCE</td>
<td>IFileSource</td>
<td>audio/raw</td>
<td>Host 2</td>
</tr>
<tr>
<td>Read Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPEG</td>
<td>CONVERTER</td>
<td>audio/mpeg</td>
<td>audio/raw</td>
<td>Host 3</td>
</tr>
<tr>
<td>Audio</td>
<td>SINK</td>
<td></td>
<td></td>
<td>Host 3</td>
</tr>
<tr>
<td>Decode Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Playback</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Node</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Network:
- Registry
  - Host 1
  - Host 2
  - Host 3
• **Command line interaction and configuration**

• **Setting up distributed flow graphs from textual descriptions**
  - `.clic <.gd file> [-i <input file>] [-o <output file>]`

• **Examples**

```plaintext
mp3play.gd:  
GenericReadNode ! MPEGAudioDecodeNode ! PlaybackNode
./clic mp3play.gd -i /home/bob/music/song.mp3
```
% Watch TV via DVB receiver in remote host

DVBRadNode #Event "setChannel(1)" ACTIVATED
#Host "host.domain.org"

! MPEGDemuxNode
{
  { ["mpeg_audio0"] ! MPEGAudioDecodeNode
    ! PlaybackNode
  }
  { ["mpeg_video0"] ! MPEGVideoDecodeNode
    ! XDisplayNode
  }
}
GraphBuilder

- Automatically creates distributed flow graph for media playback
  - Distributed source, audio sink, and video sink
- Supported URLs
  - file, audiocd, dvd, tv, dvbtv, ivtv, mpegtv
- Examples

  ```
  ./clic -u file:///home/bob/mp3/song.mp3
  ./clic -u "dvd:///dev/cdrom?title=1&chapter=4&angle=1"
  ./clic -u file://host1/movies/movie.mpeg -A host2 -V host3
  ```

  Demo
Distributed Synchronization

- Strict separation
  - Locally running controllers
  - Synchronizer globally adjusts latencies
- Global clock via NTP
Applications

- Virtual Courseroom Environment (VCORE)
  - Record and broadcast all aspects of a talk
  - Audio and video stream of the speaker, slides, written text or annotations (SMIL)
  - Successfully used at Saarland University since winter 2002
Applications

- amaroK – the audio player for KDE
- NMM engine employs GraphBuilder (next NMM release)

⇒ Demo
Goal (in April 2001)

- Networked multimedia home entertainment
- Developed completely on top of NMM
- E.g. plug-in for on-screen display
- Seamless integration of mobile devices
Multimedia-Box

- CD-Player with CDDB support
- CD-Grabber and transcoding to MP3/OggVorbis
- DVD-Player with support for menus
- DVD-Grabber and distributed transcoding
- TV, time-shifting, access to remote receiver
- Video recorder and Electronic Program Guide (EPG)
- Media player with playlist
- Multi-tasking: watch TV while transcoding a DVD
- Controllable with a remote-control
- Extensible and configurable application framework (XML)
Demo

Playlist
- Gladiator vob
- spedream1.mpeg
- Pro-7 2003-02-12 18:05.mpg

Selection /video
Disk
- DIR: Go Up
- Georg Lucas - Love.mpeg
- Pixar - For the Birds.avi
- Scrat - Gone Nutty.avi
- Shakira - Objection.mpeg
- Shakira - Where... wherever.avi
- Sheryl Crow - ever dies.mpeg

CD
- Shakira - Objection live from the 2002 MTV

Extended Information
07.01.2003
- Monday: Die Freunde treffen Mr. Steven
- Tuesday: Der Vater von Elizabeth
- Wednesday: Statt Rose war der Schwieger
- Thursday: In der Tat steht bitte
- Friday: Rose: seine Freunde, die es
- Saturday: Mich zu erzählen. Doch
- Sunday: Freundeviel versteht die Echte
- Im Falsch und schon nicht

Channel Schedule
Sat.1
- 11:58 - 12:58: ALENZIO: Schoner leben
- 12:59 - 13:59: SAIL
- 13:59 - 14:57: Arabella
- 14:57 - 15:58: Die Jugend der anderen
- 15:58 - 16:54: DDR Casting Agentur
- 16:54 - 17:00: Do It Yourself - S.O.S.
- 17:00 - 18:02: taff
- 18:02 - 19:02: Private
Session Sharing

- Service that allows to “overlap” running flow graph with query
  - E.g. shared access to DVD, different audio tracks for mobile devices

![Diagram of session sharing](image)

**Running session 1**

- DVDRead Node
- MPEG Demux Node
- Audio Decode Node
- Playback Node
- Video Decode Node
- Display Node
- Synchronizer

**Query**

- DVDRead Node
- MPEG Demux Node
- Audio Decode Node
- Playback Node
- "video"
- "audio0"
- "audio1"
- localhost
- localhost
Session Sharing

- Service that allows to “overlap” running flow graph with query
- E.g. shared access to DVD, different audio tracks for mobile devices
Seamless Handover

- Playback session running on mobile system
- Handed over to nearby stationary system
- Seamless and synchronized handover
  - No loss, no duplicates, no interruption
Summary

- Network-Integrated Multimedia Middleware (NMM)
  - Research project and Open Source project
- Simple, clear, and unified design
  - Nodes connected by jacks that stream messages
  - Unified out-of-band and instream interaction
- Object-oriented design
  - Inheritance for interfaces
- Integrating micro-core architecture
  - Middleware core plus plug-ins, services, transport and serialization strategies
- No overhead for all co-located components
Summary

• Scalable transparency
  • Network as black-box vs. tuning of network parameters
• Unique features
  • Transparently distributed flow graphs
  • Distributed GraphBuilder
  • Distributed synchronization
  • Session sharing service
  • Seamless handover
• Research project with real world applications
  • Multimedia-Box, VCORE
Future Work

- Update plug-ins
- deb/rpm packages
- Optimization of current implementation
- Support for standard protocols like SIP
- Distributed streaming server with overlay network
- Intelligent distribution of workload, e.g., for transcoding
- (Help with) support for other platforms
- (Help with) integration into KDE (and other projects)
Acknowledgments

NMM contributors: Andreas Meyer, Benjamin Deutsch, Christian Gerstner, Christoph Wellner, David Maass, David Philippi, Eric Peters, Florian Winter, Marc Klein, Markus Sand, Patrick Becker, Patrick Cernko, Patrick Wambach, Robert Wruck, Roger Dostert, Stephan Diudas, Wolfgang Enderlein, and Wolfram von Funck; Georg Demme (who supervises the VCORE project and designed the NMM logo), Andreas Pomi (helps managing the technical infrastructure at our lab).