

Multimedia

– Introduction to NMM –

Marco Lohse
Philipp Slusallek

Overview

- Motivation & quick overview
 - NMM as middleware
 - NMM for the application programmer
 - „Hello World!“-program
 - NMM for the plug-in programmer
 - Step by step
 - Next steps & questions
-

Motivation



NMM

- Network-Integrated Multimedia Middleware
- Middleware
 - Layer between application and system
 - Service-oriented
- Network-Integrated
 - Transparently use in network available devices
 - Intelligent employment of available devices
- Multimedia for Linux ?
 - No multimedia middleware so far

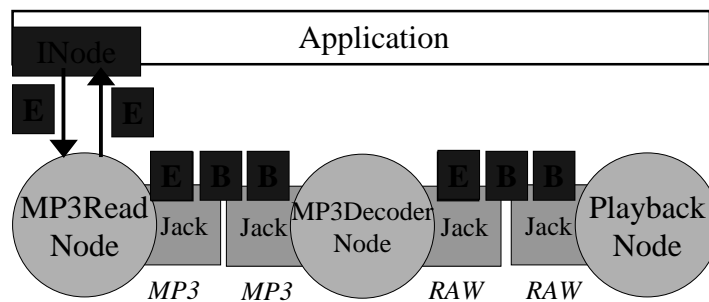
Aspects of NMM

- Application development
 - Semester project
 - Plug-in development
 - Weekly exercises and semester project
 - Core development
 - Not touched within exercises or project
 - Application and plug-in programmers still need background knowledge
-

Basics

- Application specifies a flow graph
 - **Nodes** are the processing elements
 - **Jacks** connect nodes, if **formats** match
 - **Messages** are streamed along the edges of the graph
 - Buffer
 - Chunk of multimedia data, e.g. one video frame, some audio samples, ...
 - In-stream events
 - Control information, e.g. start track / end track
 - **Interfaces** for controlling objects
 - Internally out-of-band events are sent between application and nodes
-

Flow Graph



COMPUTERGRAPHIK – UNIVERSITÄT DES SAARLANDES

Base system

- Kernel facilities
 - Thread management
 - Memory management
 - State machine
 - Format definition & negotiation
 - Synchronization
 - Event system
 - Generic nodes
 - Bi-directional communication

COMPUTERGRAPHIK – UNIVERSITÄT DES SAARLANDES

State Management

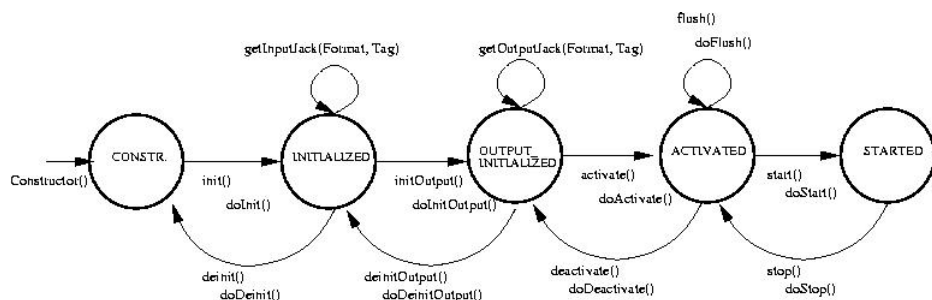
- Idea

- Some operations may only be performed in a certain state
 - E.g. filename can only be set when node not started
- Some operations have to be performed before reaching a new state
 - E.g. a filename has to be set before node can be started

State Management (2)

- do-Methods

- Plug-in specific functionality during state transitions
- Design pattern called Template Method
- E.g. `doInit()` will be called from `init()`

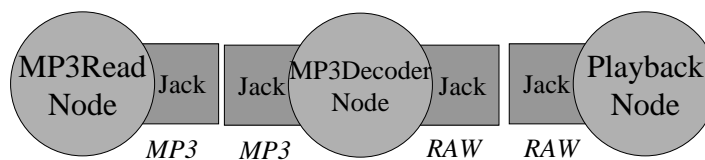


Registry

- Server registry
 - One per host
 - Administrates all nodes and resources of host
- Client registry
 - One per application
 - Requests nodes from server registry
- But ...
 - Server registry can be replaced by special object

Hello World!

Application



mp3play (1)

- Setup NMM application

```
NMMApplication* app = 0;
try {
    app = new ProxyApplication(argc, argv)
} catch (...) {
    delete app;
    app = new NMMApplication(argc, argv);
    IPluginManager& pm = app->getPluginManager();
    pm.loadLibrary("libnmmmpegread.so");
    pm.loadLibrary("libnmmmpegaudiodec.so");
    pm.loadLibrary("libnmmaudio.so");
}
```

COMPUTER GRAFIK - UNIVERSITÄT DES SAARLANDES

mp3play (2)

- Request nodes from registry

```
ClientRegistry& registry = app->getRegistry();
registry.initTransaction();

NodeDescription request;
request.setNodeName("PlaybackNode");

list<Response> response = registry.initRequest(request);
INode_var audio_play;
audio_play.reset(registry.requestNode(response.front()))

// ... more nodes

registry.releaseTransaction();
```

COMPUTER GRAFIK - UNIVERSITÄT DES SAARLANDES

mp3play (3)

- **Configure and connect nodes**
 - Order is important: successor might request data from predecessor

```
readfile->init();
mp3dec->init();
readfile->initOutput();

connect(readfile, mp3dec);

readfile->activate();
mp3dec->initOutput();
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

mp3play (4)

- **Start nodes**
- **Stop, flush and release nodes**

```
readfile->start();
mp3dec->start();
audio_play->start();

audio_play->stop();
audio_play->flush();
audio_play->deactivate();
audio_play->deinitOutput();
audio_play->deinit();
registry.releaseNode(*audio_play);
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

mp3play (5)

- Set filename via interface

```
ISourceFileHandler_var file_source  
  (readfile->getParentObject()  
   ->getCheckedInterface<ISourceFileHandler>());  
file_source->setFilename(filename);
```

- ... or (unchecked)

```
ISourceFileHandler_var file_source  
  (readfile->getParentObject()  
   ->getInterface<ISourceFileHandler>());  
if (file_source.get()) {  
  file_source->setFilename(filename);  
}
```

mp3play (6)

- Register method for specific event
 - Gets triggered when event occurs at node

```
class EventCatcher {  
public:  
  Result mySetFormat(Format& ) {  
    return SUCCESS;  
  }  
};  
  
EventCatcher ec;  
readfile->getParentObject()->registerEventListener  
  (IFormatChange::setFormat_event,  
   new TEDObject1<EventCatcher, Format>  
    (&ec, &EventCatcher::mySetFormat) );
```

Plug-in Development

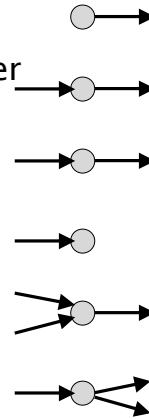
- a) Thinking
 - b) What kind of node? One node or many nodes?
 - c) What interfaces should be supported?
 - d) What formats will be supported?
 - e) What needs to be done during state transitions?
 - f) What functionality has to be performed?
 - g) How can a node be registered?
-

Step A

- Thinking
 - MP3-File-Reader
 - Reads MP3-file
 - Needs filename
 - Parses MP3-Header
 - Supports seeking operations
 - ...
-

Nodes

- Source, e.g. camera, file-reading node
 - Creates new data
- Converter, e.g. JPEG decoder, MP3 decoder
 - Converts from one format to another
- Filter, e.g. brightness or volume effect
 - Modifies data
- Sink, e.g. display, sound device
 - Consumes data
- Multiplexer, e.g. MPEG multiplexer
 - Two or more inputs
- Demultiplexer, e.g. MPEG demultiplexer
 - Two or more outputs



COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Step B

- What kind of node? One node or many nodes?
 - The more the better
 - Inherit from wanted super class

```
namespace NMM {  
    class MPEGAudioDecodeNode:  
        public GenericProcessorNode {  
  
    public:  
        MPEGAudioDecodeNode(const char* =  
                               "MPEGAudioDecodeNode");  
  
        virtual ~MPEGAudioDecodeNode();  
};  
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Step C

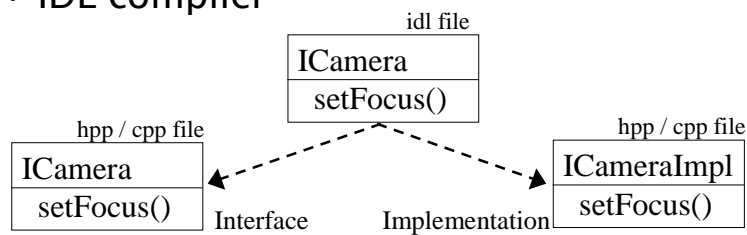
- What interfaces should be supported?
 - Find wanted interface in nmm/interfaces
 - Inherit from impl-class
 - Implement interface methods

```
#include "nmm/interfaces/video/camera/ICameraImpl.hpp"
namespace NMM{
    class MyCamera : public GenericSourceNode,
                    public ICameraImpl {
    protected: // interface methods
        virtual void setBrightness(const int& value);
        virtual int getBrightness() const;
        virtual bool hasBrightness() const;
        // some more ...
    };
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Interfaces

- Interfaces for specific functionality
 - Camera, FileHandler, ...
 - Instream and out-of-band control
- NMM Interface Definition Language (IDL)
 - Similar to CORBA IDL
- IDL compiler



COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Step C (2)

- What interfaces should be supported?
 - Write your own interface (.idl file)
 - Use IDL compiler

```
module NMM {  
    interface ICamera {  
        void setBrightness(in int value);  
        int getBrightness();  
        bool hasBrightness();  
    };  
}
```

Step C (3)

- Instream methods
 - Should return Result
 - A create_<method name> method is generated

```
module NMM {  
    interface IMyInstream {  
        Result foo(in int value) instream;  
    };  
}
```

generates ...

```
Event* IMyInstream::create_foo(const int& value);
```

Step D

- What formats will be supported?
 - Stored in properties of node
 - Static property stores generally supported formats
 - Specified in doInit() called by init()
 - Working property stores formats supported by this instance of node
 - E.g. depending on file to be read
 - Specified in doInitOutput() called by initOutput()

Step D (2)

```
Result JPEGDecodeNode::doInit() {
    Format *myInputFormat =
        getOwnInputProperty() ->addNewFormat("video/jpeg");
    myInputFormat->addWildcard(Format::x_resolution_param);
    myInputFormat->addWildcard(Format::y_resolution_param);
    getOwnInputProperty() ->setDefaultFormat(myInputFormat);
    Format *myOutputFormat =
        getOwnOutputProperty() ->addNewFormat("video/raw");
    myOutputFormat->addWildcard(Format::x_resolution_param);
    myOutputFormat->addWildcard(Format::y_resolution_param);
    getOwnOutputProperty() ->setDefaultFormat(myOutputFormat);

    myInputFormat->setIOPartner(myOutputFormat);
    myOutputFormat->setIOPartner(myInputFormat);
    return SUCCESS;
}
```

Step D (3)

```
Result JPEGDecodeNode::doInitOutput() {
    const Format* in_format = getInputFormat();

    int x_resolution =
        in_format->getIntValue(Format::x_resolution_param);
    int y_resolution =
        in_format->getIntValue(Format::y_resolution_param);

    Format* myInputFormat = new Format( *in_format );
    getOwnWorkingInputProperty() ->addNewFormat(myInputFormat);
    getOwnWorkingInputProperty() ->setDefaultFormat(myInputFormat)

    // ... some more code

    return SUCCESS;
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Step E

- What needs to be done during state transitions?
 - Implement corresponding do-method
 - E.g. doFlush() called by flush() resets library used internally

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Step F

- What functionality has to be performed?
 - Producing, processing, or consuming of buffers or instream events
 - Example
 - MP3ReadNode: produces buffers, generates instream events for each track
 - MP3DecodeNode: decodes MP3 audio in received buffers to raw audio
 - PlaybackNode: consumes raw audio
-

processBuffer()

```
Message* MyNode::processBuffer(Buffer *in_buffer)
{
    // check reference count of buffer
    in_buffer = in_buffer->getWriteableInstance();

    // get data of buffer
    char* p = in_buffer->getData();

    // modify data ...

    return in_buffer;
}
```

processBuffer() (2)

```
Message* MyNode::processBuffer(Buffer *in_buffer)
{
    // get new buffer with other size
    Buffer* out_buffer =
        getNewBuffer( in_buffer->getSize() * 2 );

    // get data of buffers
    char* p_in  = in_buffer->getData();
    char* p_out = out_buffer->getData();

    // some code ...

    // release in_buffer
    in_buffer->release();
    return out_buffer;
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Instream Events

- Composite event holds list of events

```
Message* MyNode::processBuffer(Buffer *in_buffer)
{
    in_buffer->release();

    // return new event
    return new CEvent
        ( ISomeInterface::create_foo("new Event" ) );
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Upstream Messages

- Upstream messages have higher priority

```
CEvent *e = new CEvent
  ( ISomeInterface::create_foo("some upstream
    event") );
e->setDirection(Message::UPSTREAM);
```

Instream Events (2)

- Handling, deleting, inserting, modifying, and replacing of instream events

```
Result MyNode::handleEventAndDelete()
{
  return DELETE;
}

Result MyNode::handleEventAndInsertNewEvent()
{
  Event* new_event =
    ISomeInterface::create_foo("Add a new Event");
  insertEvent(new_event);
  return SUCCESS;
}
```

Instream Events (3)

```
Result MyNode::handleEventAndReplace()
{
    Event* new_event =
        ISomeInterface::create_foo("Add a new Event");
    insertEvent(new_event);
    return DELETE;
}

Result
MyNode::handleEventAndManipulateParameters(int &i)
{
    ++i;
    return SUCCESS;
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

working-Flag

- If set to true, processBuffer() gets called with 0-message
- Flag can be set in processBuffer() or an interface method
- Purpose
 - Produces two or more output buffers from a single input buffer
 - Produce instream event(s)
 - ...

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

producing-Flag

- Similar to working-Flag
 - working-Flag == true, means no upstream messages are handled
 - producing-Flag == true
 - processBuffer() called with 0-message *unless* there is a message that can be dequeued
 - Set to true per default for source nodes
 - Set to false manually if node no longer produces output
-

Multiplexer

- More than one input, single output

```
Result MyNode::doInit()
{
    // some code ...

    // add two inputs, both are enabled per default
    addInputStream("input1");
    addInputStream("input2");

    return SUCCESS;
}
```

Multiplexer (2)

```
Message* MyNode::processBuffer(Buffer* in_buffer)
{
    if(!strcmp(getCurrentRecvInputStream(),
               "input1")) {
        // some code ...
    }
    if(!strcmp(getCurrentRecvInputStream(),
               "input2")) {
        // some code ...
    }
}
```

- Round robin dequeuing strategy if input is not disabled

```
- setRecvInputStreamEnabled("input1", false);
```

Demultiplexer

- More than one output, single input

```
Result MyNode::doInit()
{
    // some code ...

    // add two inputs, both are enabled per default
    addOutputStream("output1");
    addOutputStream("output2");

    return SUCCESS;
}
```

Demultiplexer (2)

```
Message* MyNode::processBuffer(Buffer* in_buffer)
{
    if( /* sent to output1 ? */ ) {
        setCurrentSendOutputStream("output1");
        // some code ...
        return my_output_message;
    }

    if( /* sent to output2 ? */ ) {
        setCurrentSendOutputStream("output2");
        // some code ...
        return my_output_message;
    }
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Synchronized Sinks

- **processBuffer** is split in **prepareBuffer** and **presentBuffer**
 - **presentBuffer** is called „in time“

```
void XDisplayNode::prepareBuffer(Buffer *in_buffer)
{
    // copy video frame to (non-visible) frame buffer
}
Message* XDisplayNode::presentBuffer(Buffer
    *in_buffer)
{
    // make (non-visible) frame buffer visible
}
```

COMPUTER GRAPHIK - UNIVERSITÄT DES SAARLANDES

Step G

- How can a node be registered?
 - Simple, if arbitrary number of instances can be created

```
#include "nmm/base/registry/Plugin.hpp"
#include "MyNode.hpp"

// instantiate a plugin object for each node
NMM::TPlugin< NMM::MyNode > plugin0("MyNode");

// instantiate the plugin array
NMM::Plugin* nmm_plugins[] = { &plugin0, 0 };
```

Next Steps

- Don't panic!
 - Download and install NMM
 - Or use installed version with CG account
 - Read the slides & documentation at www.networkmultimedia.org/Docs/
 - Try some of the examples
 - Read the code
 - Ask other students
 - Ask your teaching assistant
 - Mail to [mm03] or [nmm-dev]
-

Questions?

COMPUTER GRAPHIK – UNIVERSITÄT DES SAARLANDES