

# Network-Integrated Multimedia Middleware (NMM)

Marco Lohse  
Motama GmbH  
Saarbrücken  
Germany  
mlohse@motama.com

Michael Repplinger  
Computer Graphics Lab  
Saarland University  
Saarbrücken, Germany  
repplinger@cs.uni-sb.de

Florian Winter  
Motama GmbH  
Saarbrücken  
Germany  
fw@motama.com

Philipp Slusallek  
Computer Graphics Lab  
Saarland University  
Saarbrücken, Germany  
slusallek@cs.uni-sb.de

## ABSTRACT

Today's multimedia infrastructures adopt a centralized approach, where all multimedia processing takes place within a single system. The network is, at best, used for streaming data transmission. Since there is a strong trend towards networked systems, these traditional approaches are becoming obsolete. In contrast, the Network-Integrated Multimedia Middleware (NMM) offers a multimedia architecture, which considers the network as an integral part and enables the intelligent use of devices distributed across the network.

## Categories and Subject Descriptors

C.2.4 [Computer Systems Organization]: COMPUTER-COMMUNICATION NETWORKS—*Distributed Systems*

## General Terms

Design

## Keywords

Multimedia, middleware, streaming, home networking

## 1. INTRODUCTION

Besides the PC, an increasing number of multimedia devices – such as set-top boxes, PDAs, and mobile phones – already provide networking capabilities. However, today's multimedia infrastructures adopt a centralized approach: All multimedia processing takes place within a single system and the network is only used for streaming predefined content from a server to clients.

Conceptually, such approaches consist of two isolated applications, a server and a client (see Figure 1). The realization of advanced scenarios is therefore complicated and

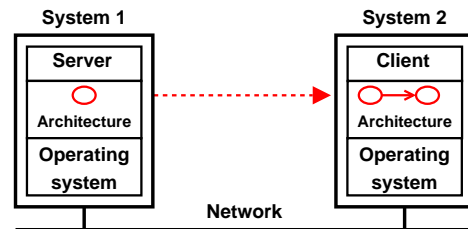


Figure 1: Client/server streaming consists of two isolated applications that do not provide fine-grained control or extensibility.

error-prone – especially since the client has typically no or only limited control of the server, and vice versa.

The Network-Integrated Multimedia Middleware (NMM) presented in this paper overcomes these limitations by enabling access to all resources within the network [1, 2]: distributed multimedia devices and software components can be transparently controlled and integrated into an application. In contrast to all other multimedia architectures available, NMM is a true *middleware*, i.e. a *distributed* software layer running in between distributed systems and applications (see Figure 2).

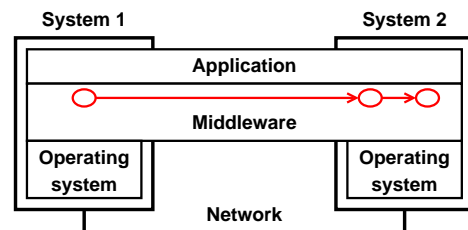


Figure 2: A multimedia middleware is a distributed software layer that eases application development by providing transparency.

The unified architecture of NMM offers a simple and easy to use interface for applications to integrate multimedia functionality. Therefore, it can be used as enabling technology for locally operating multimedia applications, but more im-

importantly for all kinds of networked and distributed multimedia systems – spanning from embedded and mobile systems, to PCs, to large-scale computing clusters.

NMM is implemented in C++ and NMM-IDL, an interface definition language. For developing new plug-ins or applications on top of NMM, you need a set of fundamental and optional NMM libraries. The NMM Software Developer Kit (NMM-SDK) available online [9] greatly simplifies this process.

NMM runs on a large number of operating systems, such as Linux, Windows XP/Vista, and Mac OS X, and supports a large number of platforms, such as 32/64 bit and little/big endian CPUs, Cell processor of Playstation 3, embedded systems, etc. Being a true middleware, NMM allows to create multimedia applications that span across all possible combinations of all supported operating systems and platforms.

The NMM project started in 2000. It is a joint research and development effort of Motama, a spin-off company of Saarland University, Germany, and the Computer Graphics Lab at Saarland University [9]. NMM is released under a dual-license model. While the result of our work is available as Open Source under the terms and conditions of the GPL, commercial licenses are offered by Motama.

## 2. GENERAL DESIGN APPROACH

The general design approach of the NMM architecture is similar to other multimedia architectures – but is extended to a truly network transparent approach as described in the following.

### 2.1 Nodes, Jacks, and Flow Graphs

Within NMM, all hardware devices (e.g. a TV board) and software components (e.g. decoders) are represented by so called *nodes*. A node has properties that include its input and output ports, called *jacks*, together with their supported multimedia *formats*. A format precisely defines the multimedia stream provided by specifying a type, such as “audio/raw” for uncompressed audio streams, plus additional parameters, such as the sampling rate.

Depending on the specific kind of a node, its innermost loop produces data, performs a certain operation on the data, or consumes data. Our system distinguishes between different types of nodes: *source*, *sink*, *converter*, *filter*, *multiplexer*, *demultiplexer*, and a generic *mux-demux* node.

These nodes can be connected to create a *flow graph*, where every two connected jacks need to support a “matching” format. The structure of this graph then specifies the operation to be performed, e.g. reading, decoding, and playback of an MP3 file.

Together, more than 60 nodes are already available for NMM, which allows for integrating various input and output devices, codecs, or specific filters into an application. A complete list of available nodes is available online [9].

### 2.2 Messaging System

The NMM architecture uses a uniform messaging system for all communication. There are two types of messages. On the one hand, multimedia data is placed into *buffers*. On the other hand, *events* contain control information, such as a change of speaker volume. Events are identified by a name and can include arbitrary typed parameters.

There are two different types of interaction paradigms used within NMM. First, messages are streamed one-way

along connected jacks. This type of interaction is called *instream* and is most often performed in *downstream* direction, i.e. from sources to sinks; but NMM also allows for sending messages in *upstream* direction.

Notice that both buffers and events can be sent instream. For instream communication, so called *composite events* are used that internally contain a number of events to be handled within a single step of execution. Instream events are very important for multimedia flow graphs. For example, the end of a stream (e.g. the end of a file) can be signaled by inserting a specific event at the end of a stream of buffers. External listener objects can be registered to be notified when certain events occur at a node (e.g. for updating the GUI upon the end of a file or for selecting a new file).

Events are also employed for the second type of interaction called *out-of-band*, i.e. request-reply interaction between the application and NMM objects, such as nodes or jacks. Events are used to control objects or for sending notifications from objects to registered listeners.

### 2.3 Interfaces

In addition to manually sending events, object-oriented *interfaces* allow to control objects by simply invoking methods, which is more type-safe and convenient than sending events. These interfaces are described in NMM Interface Definition Language, which is similar to CORBA-IDL, but was extended to allow for specifying out-of-band as well as instream interaction.

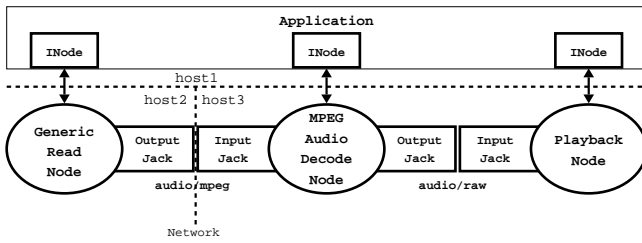
For each description, an IDL compiler creates an interface and implementation class in C++. While an implementation class is used for implementing specific functionality within a node, an interface class is used within applications for interacting with objects, or within nodes for creating instream events.

### 2.4 Distributed Flow Graphs

What is special about NMM is the fact that NMM flow graphs can be distributed across the network: local and remote multimedia devices or software components encapsulated within nodes can be controlled and integrated into a common multimedia processing flow graph, a *distributed flow graph*. While this distribution is transparent for application developers, no overhead is added to all locally operating parts of the graph: In such cases, references to already allocated messages are simply forwarded – no networking is performed at all.

For all distributed parts of a flow graph, suitable networking technologies, protocols, and serialization formats are either negotiated automatically or set up manually (i.e. scalable transparency [1]). While out-of-band communication requires reliable handling of events, instream interaction can use arbitrary combinations of protocols especially suited for the parallel yet ordered transmission of time-sensitive multimedia data and control events [10]. This, for example, allows employing RTP/RTCP for multimedia data in parallel with TCP for instream events.

Figure 3 shows an example for a distributed flow graph for playing back MP3 files. A source node for reading data from the local file system (GenericReadNode) is connected to a node for decoding MPEG audio streams (MPEGAudioDecodeNode). This decoder is connected to a sink node for rendering uncompressed audio using a sound board (PlaybackNode). Once the graph is started, the source nodes



**Figure 3: A distributed flow graph for playing back MP3 files.**

reads a certain amount of data from a given file, encapsulates it into buffers, e.g. of size 1024 bytes, and forwards these buffers to its successor. After being decoded to uncompressed “raw” audio, the converter node forwards data buffers to the sink node.

The application controls all parts of this flow graph using interfaces, e.g. `INode` for controlling the generic aspects of all instantiated nodes. Notice that three different hosts are present in our example. The application itself runs on `host1`, the source node on `host2`, and the decoder and sink node on `host3`. Therefore, NMM automatically creates networking connections between the application and the three distributed nodes (out-of-band interaction), but also between the source and the decoder node (instream interaction). As a result, compressed MPEG audio data is transmitted over the network.

Notice that such a simple but distributed flow graph already provides many benefits. First, it allows an application to access files stored on distributed systems without the need for a distributed file system, such as NFS. Second, the data streaming between connected distributed nodes is handled automatically by NMM. Third, the application acts as “remote control” for all distributed nodes. As an example, this allows for transparently changing the output volume of the remote sound board by a simple method invocation on a specific interface provided by the `PlaybackNode`, e.g. `IAudioDevice`.

## 2.5 Distributed Synchronization

Since NMM flow graphs can be distributed, they allow for rendering audio and video on different systems. For example, the video stream of an MPEG2 file can be presented on a large screen connected to a PC while the corresponding audio is played on a mobile device. In addition, media presentations can also be performed on several systems simultaneously. A common application is the playback of the same audio stream using different systems located in different rooms of a household – a home-wide music system. To realize synchronous playback of nodes distributed across the network, NMM provides a generic *distributed synchronization architecture* [1].

## 2.6 Registry Service

The *registry service* of NMM allows for the discovery, reservation, and instantiation of nodes available on local and remote hosts [1]. On each host, a unique *registry server* (included within the NMM application called “`serverregistry`”) administrates all NMM nodes available on this particular system. For each node available on the local system, the

server registry stores a complete *node description* that includes the specific type of a node (e.g. “sink”), its name (e.g. “`PlaybackNode`”), the provided interfaces (e.g. “`IAudioDevice`”), and the supported input and output formats.

The application uses a *registry client* to request nodes by using node descriptions, or complete flow graphs by using *graph descriptions*. Requests for distributed nodes are forwarded using a peer-to-peer approach. After successfully processing the request, the registry service creates nodes on the local host and remote hosts, respectively, and connects nodes, if a graph description was given.

## 3. APPLICATION DEVELOPMENT

A main goal of NMM is to ease the development of (distributed) multimedia applications in C++ [1]. An extensive tutorial showing several “Hello-World!” examples is available online [9]. In particular, we also demonstrate how easy it is to develop NMM applications employing distributed flow graphs: By simply adding a single line, a node within a flow graph can be distributed to a remote host, while usage and control of the node remains fully network transparent.

## 4. TOOLS

The tool “`clic`” allows for setting up distributed flow graphs from a simple textual description. A simple MP3 player can be created by using “`clic`” with following description stored in a file named “`mp3play.gd`”.

```
GenericReadNode
! MPEGAudioDecodeNode
! PlaybackNode
```

The tool “`clic`” can now be started to play “`song.mp3`” using following command:

```
user@host1> clic mp3play.gd -i song.mp3
```

If reading, decoding, and rendering of the audio stream is to be performed on remote hosts as in Figure 3, the description needs to be changed only slightly:

```
GenericReadNode      #setLocation("host2")
! MPEGAudioDecodeNode #setLocation("host3")
! PlaybackNode        #setLocation("host3")
```

In addition to starting the tool “`clic`” on `host1`, the NMM application “`serverregistry`” needs to be running on `host2` and `host3`.

For synchronously playing the audio file read from `host2` on `host3` and the local host (i.e. `host1`), following description is used.

```
GenericReadNode #setLocation("host2")
{
  { ["default"] ! MPEGAudioDecodeNode
                    #setLocation("host3")
                    ! PlaybackNode
                    #setLocation("host3")
  }
  { ["default"] ! MPEGAudioDecodeNode
                    ! PlaybackNode
  }
}
```

## 5. ADVANCED MIDDLEWARE SERVICES

Based on the basic facilities provided by NMM and its registry service, several new and advanced middleware services have been proposed and implemented:

A *graph building* [7] and *format negotiation service* [8] demonstrate new techniques that allow an application to only specify the most relevant parts of an intended functionality; the distribution of the resulting configuration and the connection of originally incompatible nodes are handled automatically and guided by a user-centric quality measurement.

In order to increase the sense of collaboration among users, the *session sharing service* [4] allows to automatically map requests sent to the registry service to already running flow graphs. This allows to simultaneously and synchronously access the same content on different devices controlled by different users. At the same time, this helps to reduce resource requirements by reusing (parts of) already active flow graphs and enables shared access to a single hardware device.

To enable mobile users to explore the capabilities of their surrounding environment, a further advanced middleware service is realized, which allows for *seamless reconfigurations* of active flow graphs [3]. As an application, we demonstrate the *seamless handover* of audio playback between systems, e.g. from a mobile system to a stationary system with high-quality audio output capabilities. Although audio rendering is the most timing sensitive part of media handling, NMM allows for performing such a handover seamless and without any loss or interruptions.

A *Quality of Service (QoS) service* within NMM automatically adapts the quality of video and audio streams to “optimally” use the bandwidth currently available in the network in order to always deliver the best possible user experience. This is achieved by combining different monitoring and adaptation strategies, e.g. for adapting bitrate, framerate, image size or codecs of audio/video streams based on observed packet loss, signal strength, and other criteria.

## 6. APPLICATIONS

We are especially focusing on the development of networked *home entertainment* systems, which employ various mobile and stationary devices in combination, e.g. for watching TV throughout a household [1, 5, 6].

In the area of *building technologies*, NMM is used as “translator” between proprietary systems available on analog busses, and standardized technologies of today’s digital networks, such as VoIP technologies.

Distributed *content processing* is another important application area for NMM. Since NMM is fully network transparent, time consuming tasks such as transcoding or content analysis can be distributed easily to the computing power of clusters of commodity PCs. While NMM handles all aspects of the networking involved, applications can concentrate on implementing optimized distribution and load-balancing strategies.

Another important field of application for NMM are large-scale *multimedia installations*, e.g. for digital signage. In particular, NMM allows for creating video walls. Such a video wall consists of several video screens that are placed on top of each other or side by side, where each screen only shows the section of a video frame that corresponds to its position within the array of screens. In contrast to expensive and proprietary hardware solutions, NMM offers scalability, full flexibility and programmability of a software driven solution, and allows for employing cheap commodity PC hardware.

## 7. CONCLUSIONS

The Network-Integrated Multimedia Middleware (NMM) offers a new approach to networked multimedia by providing a fully transparent view on distributed systems and thereby removes the artificial boundaries of traditional client/server streaming applications. As a result, NMM is the only full-featured multimedia middleware available.

NMM is used as fundamental software layer within various commercial products in the areas of home entertainment, building technologies, content processing and distribution, and multimedia installations. The dual-licensing model of NMM offers commercial licenses and services for industrial partners. Still, NMM is available as Open source and offers all its benefits for other Open Source projects and research efforts in academia.

## 8. REFERENCES

- [1] M. Lohse. *Network-Integrated Multimedia Middleware, Services, and Applications*. VDM Verlag, 2007.
- [2] M. Lohse, M. Repplinger, and P. Slusallek. An Open Middleware Architecture for Network-Integrated Multimedia. In *Proceedings of the Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems (IDMS/PROMS)*, 2002.
- [3] M. Lohse, M. Repplinger, and P. Slusallek. Dynamic Distributed Multimedia: Seamless Sharing and Reconfiguration of Multimedia Flow Graphs. In *Proceedings of the 2nd International Conference on Mobile and Ubiquitous Multimedia (MUM)*, 2003.
- [4] M. Lohse, M. Repplinger, and P. Slusallek. Session Sharing as Middleware Service for Distributed Multimedia Applications. In *Proceedings of the First International Workshop on Multimedia Interactive Protocols and Systems (MIPS)*, 2003.
- [5] M. Lohse, M. Repplinger, and P. Slusallek. Dynamic Media Routing in Multi-User Home Entertainment Systems. In *Proceedings of the Eleventh International Conference on Distributed Multimedia Systems (DMS)*, 2005.
- [6] M. Lohse and P. Slusallek. Middleware Support for Seamless Multimedia Home Entertainment for Mobile Users and Heterogeneous Environments. In *Proceedings of the 7th IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA)*, 2003.
- [7] M. Lohse and P. Slusallek. Towards Automatic Setup of Distributed Multimedia Applications. In *Proceedings of the 9th IASTED International Conference on Internet and Multimedia Systems and Applications (IMSA)*, 2005.
- [8] M. Lohse, P. Slusallek, and P. Wambach. Extended Format Definition and Quality-driven Format Negotiation in Multimedia Systems. In *Proceedings of the Eurographics Multimedia Workshop*, 2001.
- [9] Motama GmbH. NMM homepage. <http://www.motama.com/nmm.html>.
- [10] M. Repplinger, F. Winter, M. Lohse, and P. Slusallek. Parallel Bindings in Distributed Multimedia Systems. In *Proceedings of the 7th International Workshop on Multimedia Network Systems and Applications (MNSA)*, 2005.