Dynamic Media Routing in Multi-User Home Entertainment Systems

Marco Lohse, Michael Repplinger, and Philipp Slusallek
Computer Graphics Lab, Department of Computer Science
Saarland University, Saarbrücken, Germany
{mlohse, repplinger, slusallek}@cs.uni-sb.de

Abstract

Today, there is a growing interest in home entertainment systems consisting of various networked devices, such as set-top boxes, hi-fi systems, TV sets, or multimedia PCs. However, available solutions only provide a very restricted set of features. Dynamic routing of media streams between distributed devices and multi-user scenarios are typically not supported.

In this paper, we present an architecture for a distributed home entertainment system that overcomes these limitations. We especially examine the most important tasks, namely watching and recording TV. The overall system consists of various devices and allows for multiple users to perform different tasks in parallel. Our architecture supports multi-room applications with a single media stream being presented synchronously on different distributed devices. In addition, applications can be handed over to nearby systems. Finally, we present an algorithm that dynamically determines the optimal media routing for such multi-user scenarios.

1 Introduction

The area of multimedia home entertainment has seen a clear trend towards networked devices, such as set-top boxes, hi-fi systems, or TV sets. Often, existing multimedia PCs are also connected to the home network. In future scenarios, more and more systems will be integrated into our environment invisibly. Besides their general networking capabilities, most of these devices are today fully programmable, which – in principle – allows for advanced application scenarios to be realized.

In contrast, commercially available solutions only provide very restricted functionality. For example, the TV program received by a set-top box can only be rendered to a directly connected display or streamed to a single specialized streaming client. If different media sources are available, e.g. different receivers for analog or digital TV, they can not be integrated seamlessly into the system. Recording TV shows from different sources becomes a tedious task, where each device needs to be programmed manually. More advanced services are not supported.

Previous research in the area of multimedia home entertainment has mainly concentrated on providing location-aware services. Different approaches for accessing media streams using nearby stationary devices are presented in [1] for IEEE 1394 networks and as an OSGi compatible solution in [3]. A solution for "follow-you-and-me video" is described in [7]. Multimedia service delivery with dynamic handoffs to mobile or stationary devices is also presented [2]; synchronized and seamless handoffs were demonstrated in [5]. In [9], an application framework is proposed that can map running applications between different environments depending on the users position.

In contrast, the software architecture presented in this paper provides a transparent view on the network and therefore allows for realizing much more complex scenarios. In particular, we examine the dynamic routing of media streams in multi-user environments, especially for the common tasks of watching and recording TV. Using our system, users can transparently access all available TV channels provided by different distributed sources, e.g. receivers for digital or analog TV connected to the home network. When applications compete for these resources – e.g. when different users try to access the same media source, or a video recording task interferes an already running task – our system tries to share resources or to re-route media streams in order to keep the current quality of service. In addition, a running task can be duplicated to run simultaneously at several locations. As a special case, the media output of an active application can also be handed over between different systems, e.g. to the system that is closest to the user. We argue that such services are essential for future multimedia home entertainment systems.

In the following, we first describe an exemplary home network in Section 2. Then, the various applications and the imposed requirements are pointed out in Section 3. In Section 4, the dynamic routing of media streams will be

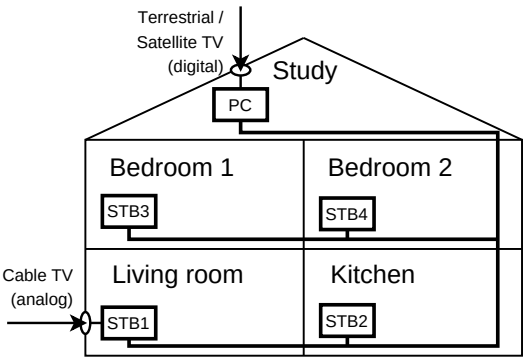

Figure 1. Exemplary setup of a home entertainment network including different TV receivers (analog and digital) and various output devices (STB1-4, and PC).

described for several application scenarios of interest. We start the presentation with relatively simple configurations that only include a single user and one or two running tasks. These configurations will be extended incrementally to include more complex setups. A generic algorithm that optimizes media routing is presented in Section 5. Finally, conclusions are drawn and future work is presented in Section 6.

2 Exemplary Home Network

Figure 1 shows an exemplary setup of a network for multimedia home entertainment. Notice that we use a moderate sized example for simplicity; the presented approach can also be applied for larger numbers of users and devices. Our setup includes different devices connected using commodity fully switched 100 Mbit LAN networking technology that is able to transmit several encoded high quality audio/video streams in parallel.

Two sources for receiving TV are available in our example. The set-top box in the living room (STB1) receives a large number of channels via analog cable TV and is used to watch TV using a connected display. Furthermore, we assume that this device also includes a hardware unit for encoding media streams to MPEG2 in real-time. Such encoded streams can then either be stored on the hard disc integrated into the device itself or transmitted to other devices within the network.

As second source, a PC in the study contains a board for receiving terrestrial or satellite TV, e.g. using one of the standards defined by Digital Video Broadcasting (DVB), Advanced Television Committee (ATSC), or Integrated Services Digital Broadcasting (ISDB). Since the received streams are already available as MPEG2, they can either be stored on the hard disc of the PC or forwarded to other devices.

In particular, three additional set-top boxes, STB2, STB3 and STB4, are available within the kitchen and the two bedrooms, respectively. These devices can be used to decode and render MPEG2 streams received via the network. Notice that this feature is also supported by STB1 and the PC.

We assume that all these devices are fully programmable. In particular, each device runs a software architecture supporting distributed multimedia – the Network-Integrated Multimedia Middleware (NMM) [4]. The features of this architecture will be described in more detail in Section 4.

Notice that TV sources provide different quality. For example, we assume that streams received digitally (PC) offer a better quality then analog TV encoded to MPEG2 (STB1). Furthermore, each TV source provides a specific list of channels. For the following examples, we assume that the PC includes a newly purchased DVB-T board for receiving terrestrial digital TV – a typical scenario for Germany and other European countries. Therefore, only 15-20 TV channels are available using the digital TV receiver, compared to 30 or more for analog TV. While some channels might be available for both sources, some can only be watched by accessing a specific device. To take the differences in quality and number of channels into account, we use a *unified channel list*. Within this list, all available channels (for analog and digital TV) are arranged according to users' preferences. For each channel, available TV sources and the corresponding devices are given and sorted according to their "quality", i.e. digital TV is preferred over analog.

3 Applications and Requirements

For the two users present in our setup, namely Alice and Bob, following applications are provided.

- *Live TV:* A user watches a specific channel using a TV source (analog using STB1 or digital using PC) and an output device (STB1, STB2, STB3, STB4, or PC). Channel-hopping is fully supported, i.e. a user can switch between channels received by different devices seamlessly. Live TV supports multi-room playback, i.e. the same media stream can be rendered synchronously in several rooms using different available devices. A special case thereof is a "follow-me" scenario, where a user can hand over the media output of an already started live TV application to his current location, e.g. when moving between rooms.
- *Video recorder:* Records a TV show encoded as MPEG2 to the hard disc of STB1 or the PC, respectively. A recording is specified by a *timer* that includes start and end time an the intended channel; the TV

source to be used is determined when programming the timer.

- *Electronic Program Guide (EPG):* A single EPG task (e.g. running on the PC) collects program information transmitted within the digital data stream and stores it within a database. All devices can access this database for displaying additional information on the current TV show. Furthermore, users can browse the EPG database for easily programming timers for video recorders.

Notice that the system described in the following is able to handle several simultaneously running instances of the live TV and the recorder application, e.g. because multiple users are active or more than a single TV show is to be recorded at a time.

From the above described applications, following general requirements can be derived.

- Access, control, and integration of media streams received by distributed devices needs to be provided, e.g. for remote TV receivers connected to the network.
- Simultaneous and shared access to a single distributed device has to be supported, e.g. to allow for different users to watch the same TV channel using different output devices in different rooms.
- Synchronous playback of media using several networked systems is required, e.g. to avoid offsets in the audio and video presentation when a single stream is to be rendered using output devices located in adjacent rooms. This feature is needed for supporting multi-room and "follow-me" presentations.
- A last requirement to be fulfilled is the appropriate handling of the dynamic behavior of the overall system including concurrently operating applications (e.g. live TV and channel-hopping, video recorders, or background tasks that acquire EPG information) competing for shared resources (e.g. TV receivers).

4 Dynamic Media Routing

We first describe the architectural approach of the underlying multimedia framework used for our system. Then, the dynamic routing of media streams is presented for different use cases of importance for home entertainment.

4.1 Architectural Model

Within common multimedia architectures, such as DirectShow [8] or the Java Media Framework [10], all multimedia functionality is modeled by *flow graphs*. A flow graph is a directed graph consisting of independent processing elements. Multimedia data streams are encapsulated into buffers that are forwarded in "downstream" direction from sources of data to sinks. Intermediate nodes perform transformations on data streams, e.g. decoding or demultiplexing.

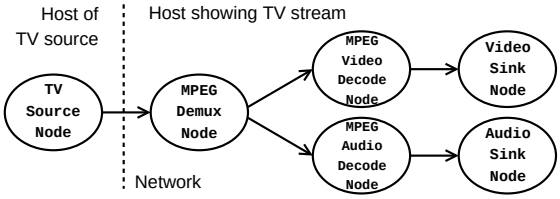

Figure 2. Flow graph for live TV including a possibly remote TV source and nodes for demultiplexing, decoding and rendering the received MPEG2 audio/video stream.

While flow graphs of commonly available architectures are restricted to operate on a single host, the underlying NMM framework used for our work allows for transparently *distributed flow graphs* [4]. Processing elements, called *nodes*, running on remote systems can be connected and controlled the same way as locally running elements. A *registry service* supports the search for specific components within the network and handles the creation of distributed flow graphs.

4.2 Live TV and Channel-Hopping

Since NMM allows for transparently distributed flow graphs, the above described live TV application can be realized by creating the setup shown in Figure 2: a flow graph including a possibly remote TV source and nodes for demultiplexing, decoding and rendering the received MPEG2 audio/video stream. Notice that any device in the home network can host this application and therefore be used as output device. If, for example, user Alice wants to watch TV in the kitchen using STB2, the corresponding application will request a TV source from either STB1 or the PC; the audio/video presentation will performed on STB2.

The TV source to be used is determined by a look-up in the unified channel list (compare Section 2). When Alice changes channels, the system checks if the newly selected channel is still supported by the currently used TV receiver. If not, the flow graph is dynamically adapted, i.e. the source node is replaced. To avoid interrupting the presentation, the new data source is first requested and configured for the selected channel before the data stream from the currently used source node is interrupted. Together, this allows for seamlessly and transparently switching between different distributed TV receivers.

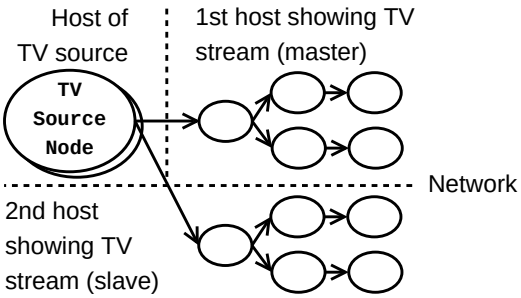

Figure 3. The source node for receiving TV is shared among two independent distributed flow graphs to enable multi-room live TV and a follow-me feature.

4.3 Multi-Room Live TV and Follow-Me TV

Another important use case occurs if Alice wants to continue watching TV while moving within the home. As an example, consider the situation when Alice wants to watch TV while preparing a meal in the kitchen. Then, she changes her location several times, e.g. she enters and leaves the living room when serving dinner. Such scenarios require to route the same media stream to several output devices simultaneously.

Within our architecture, a special service called *session sharing* provides this functionality [6]. This service automatically maps request for flow graphs to be created newly to already running flow graphs. All nodes within running flow graphs that are "identical" are determined and then shared among different applications. Conceptually, this results in two independent applications, each using a specific flow graph that internally shares certain nodes with another flow graph.

For our example, the result of this process is shown in Figure 3. The source node for receiving TV is shared among two applications operating on different hosts. Since Alice started to watch TV in the kitchen using STB2, the corresponding flow graph is system is called *master*; the additionally created flow graph running on STB1 in the living room is called *slave*. Notice that full control is provided for both systems. This means that Alice can change channels operating STB1 or STB2, e.g. using a remote control.

In addition, the underlying NMM architecture provides lip-synchronous playback of audio and video on all participating devices. The offset between different systems is reduced to only a few milliseconds, which is sufficient for multi-room solutions [6]. This is achieved by using following approach. First, the internal clocks of all systems are synchronized using the Network Time Protocol (NTP). Second, the maximum networking and processing delay of all paths within all shared flow graphs is determined. The audio and video presentation of all devices is then delayed artificially until this offset is reached. During runtime, the maximum delay is updated if necessary.

For realizing a "follow-me" service, the session sharing service is first used to create an additional shared flow graph that renders audio and video on a different system, e.g. a nearby system that is manually selected by the user. Once the slave graph is running, the master graph is destroyed.

4.4 Live TV versus EPG

In this use case, Bob starts watching sports using the TV in the living room connected to STB1. Let us assume that the live TV application chooses the digital TV receiver (PC) because it is the preferred data source for the selected channel. However, since the EPG application is continuously running in the background and using the digital receiver to collect EPG information, we define following rule:

> *Live TV can take control of a TV source that is used by an EPG task.*

When the live TV application running on STB1 requests the digital TV source, only a shared instance is returned by the session sharing service as described in Section 4.3. Based on our rule, the live TV application takes control of this source and informs the EPG task that channel switching is no longer allowed. Therefore, the EPG task tries to use another digital TV source. Since there is no such resource available in our setup, the EPG task continues to collect program information from the channel that is currently watched by the live TV application controlled by Bob. As soon as the live TV application controlled by Bob stops using the digital TV receiver, the EPG task takes over control again and continues collecting program information from all channels.

4.5 Video Recorder Timers

In the next use case, Bob wants to schedule the recording of a TV show to be broadcasted next week. Therefore, he selects the corresponding entry using the EPG, which in turn creates a timer to be stored in a global database accessible by all recording tasks. If there are no other conflicting timers, the digital TV receiver is chosen as preferred data source.

However, in general, there will be several timers provided by different users. Therefore, whenever a new timer is created, the EPG checks whether it can schedule all recordings using all available TV sources. For assigning TV sources to timers, we use a depth-first search. The algorithm starts by assigning the most preferred TV source for the specified channel to the timer that was provided first. Then, this process is repeated for all further timers. In each

step, the most preferred TV sources are checked first. As soon as a TV sources is assigned to each timer, the algorithm terminates. If no solution can be found at all, the last timer provided is discarded and a message is displayed to inform the user that the timer could not be set.

In the worst case, the algorithm needs to test n^t combinations, where n is the number of TV sources and t the number of timers. However, since we use a greedy depth-first search that terminates once a solution is found, the runtime of the algorithm is acceptable even for larger numbers of TV sources and timers. For future work, we would like to apply more advanced approaches.

4.6 Multi-User Live TV

Another use case occurs if several users are watching TV concurrently and try to access the same TV source, e.g. during channel-hopping. For handling this case, we define following rule:

> *The live TV application that requests a TV source first obtains full control of it.*

As an example, consider that Bob is still watching sports using the TV in the living room connected to STB1; the corresponding live TV application is using the digital TV receiver as data source (PC). If Alice starts watching TV in the kitchen using STB2, she will only obtain full control of the analog TV source – the digital TV source can only be shared without any control options. When Alice switches to a channel that is preferably received via digital TV (or that is only available using the digital TV receiver), the live TV application controlled by Alice requests the currently set channel of the digital TV source *before* using it as shared node within its flow graph (compare Section 4.2). If the channel requested by Alice is not the channel that Bob is currently watching, another TV source is tried to be chosen, e.g. the analog TV source. If the channel is not available using that source, a message is displayed, telling Alice that the chosen channel is currently not available since the required TV source is controlled by Bob.

4.7 Multi-User Live TV versus Recording

Finally, the most demanding use case occurs if several users are watching live TV and a video recording must be started. According to traditional analog video cassette recorders, our recording application will switch to the specified channel of the assigned TV source even if a user is currently watching a different channel using that source. Together, we apply following rule:

> *A video recorder can take full control of a TV source that is used by another task, such live TV or the EPG.*

Let us extend the example presented in Section 4.6. Again, Alice and Bob are watching TV concurrently. The timer to be recorded next is scheduled to use the digital TV source (compare Section 4.5). Thus, the video recorder requests the digital TV source, but only obtains a slave graph since this device is already used by Bob. Based on our rule, the video recorder takes control of the TV source, switches to the intended channel and starts recording – in fact, it becomes the master. Therefore, Bob is notified about this change.

If the video recorder accesses the same program that Bob is currently watching, the running live TV application is continued without further modification. Otherwise, the live TV application tries to find another TV source that can be used for watching the channel that was originally selected by Bob. Since no such TV source is available in our example, a message is displayed stating that the channel is no longer available. In this case, Bob can either watch the channel that is currently recorded or manually cancel the recorder application. In addition, Bob can switch to the channel that Alice is currently watching. Alice still has full control of the analog TV source and can switch between all its available channels.

5 Generic Media Routing Algorithm

Based on the previous sections we derive a generic algorithm for media routing. The algorithm expects three arguments. The first, `source_list`, is a list that includes all TV sources that provide the channel to be selected, which is given as second argument called `channel`. The third argument `application` specifies the application that invoked the algorithm.

The algorithm is split into three parts. In the first part, we try to find a TV source that can be fully controlled by the application. The main idea is to request each suitable TV source using the session sharing service (`requestSource()`). This service returns a master graph if the TV source is currently unused. Then, we specify the application to be the "master" of the TV source (`setMaster()`), i.e. the TV source will only accept commands, like channel switching, from the application specified.

Otherwise, the algorithm tries to get full control of the TV source based on the rules defined in Section 4. If our application can take control of a TV source, we notify the original master of this source that it can no longer switch the channel (`removeMaster()`). Since the original master of the TV source may need another TV source to continue its work, it can also invoke this algorithm to find another possible solution.

If no possibilty was found to acquire full control of a TV source, a TV source that is already set to the wanted channel

is tried to be used as shared node (`currentChannel()`) within the second part of the algorithm.

Otherwise, the third part of the algorithm is reached: An error message is returned, if the method was invoked by a live TV application, because no suitable TV source can be used. If the algorithm was invoked by an EPG task, we simply return the first TV source of the list of TV sources that can be used to collect at least the program information of the corresponding selected channel. Together, the pseudo code of the algorithm is as follows.

```
tv_source searchTVSource( source_list, channel, application )
{

  /* First part of the algorithm */
  foreach( tv_source in source_list ) do {

    if( requestSource( tv_source ) == Master ) {
      setMaster( tv_source, application );
      return tv_source;
    }

    if( application == LiveTV ) {
      if( isMasterOf( tv_source ) == EPG ) {
        removeMaster( tv_source );
        setMaster( tv_source, application );
        return tv_source;
      }
    }

    if( application == VideoRecorder ) {
      if( isMasterOf( tv_source ) == EPG or
          isMasterOf( tv_source ) == LiveTV ) {
        removeMaster( tv_source );
        setMaster( tv_source );
        return tv_source;
      }
    }

  }

  /* Second part of the algorithm */
  foreach( tv_card in source_list ) do {

    if( currentChannel( tv_card ) == channel) {
      return tv_card;
    }

  }

  /* Third part of the algorithm */
  if( application == LiveTV ) {
    throw Exception( ''Channel is currently not available'' );
  }

  if( application == EPG ) {
    return firstEPGSource( source_list );
  }

}
```

6 Conclusions and Future Work

In this paper, we presented an architecture for dynamically routing media streams in multimedia home networks. While we especially examined the important tasks of watching and recording TV, the described approach can be applied to other sources of media as well. The overall system consists of various networked devices, such as set-top boxes and PCs. Different types of TV sources are seamlessly integrated and can transparently accessed by all applications. Users are allowed to watch TV on any device, media sources are determined automatically. Furthermore, our architecture supports multi-room and location-aware presentations. When different applications, such as live TV, video recording, or an EPG task, are operating concurrently, a generic algorithm determines an optimal routing of media streams and dynamically adapts to changing conditions.

Future research will concentrate on the integration of mobile devices. Especially the automatic adaptation of media streams for such resource-poor systems seems to be demanding. Since users currently have to select their location manually, we would also like to include solutions for tracking users and mobile devices into our architecture.

References

[1] H. Baldus, M. Baumeister, H. Eggenhuissen, A. Montvay, and W. Stut. WWICE – An Architecture for In-Home Digital Networks. In *Proceedings of Multimedia Computing and Networking (MMCN)*, 2000.

[2] Y. Cui, K. Nahrstedt, and D. Xu. Seamless User-level Handoff in Ubiquitous Multimedia Service Delivery. *Multimedia Tools and Applications Journal, Special Issue on Mobile Multimedia and Communications and m-Commerce*, 22, 2004.

[3] H.-J. Eikerling and F. Berger. Design of OSGi Compatible Middleware Components for Mobile Multimedia Applications. In *Protocols and Systems for Interactive Distributed Multimedia Systems, Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems, IDMS/PROMS 2002, Proceedings*, 2002.

[4] M. Lohse, M. Repplinger, and P. Slusallek. An Open Middleware Architecture for Network-Integrated Multimedia. In *Protocols and Systems for Interactive Distributed Multimedia Systems, Joint International Workshops on Interactive Distributed Multimedia Systems and Protocols for Multimedia Systems, IDMS/PROMS 2002, Proceedings*, 2002.

[5] 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)*, 2003.

[6] M. Lohse, M. Repplinger, and P. Slusallek. Session Sharing as Middleware Service for Distributed Multimedia Applications. In *Interactive Multimedia on Next Generation Networks, Proceedings of First International Workshop on Multimedia Interactive Protocols and Systems, MIPS 2003*, 2003.

[7] J. Nakazawa and H. Tokuda. A Pluggable Service-to-Service Communication Mechanism for Home Multimedia Networks. In *Proceedings of ACM International Conference on Multimedia*, 2002.

[8] M. D. Pesce. *Programming Microsoft DirectShow for Digital Video and Television*. Microsoft Press, 2003.

[9] M. Roman, C. K. Hess, R. Cerqueira, A. Ranganathan, R. H. Campbell, and K. Nahrstedt. Gaia: A Middleware Infrastructure to Enable Active Spaces. *IEEE Pervasive Computing*, 1(4), 2002.

[10] Sun Microsystems. *Java Media Framework API Guide*, JMF 2.0 FCS edition, 1999.