MSC’05

Proceedings of the First ACM International Workshop on Multimedia Service Composition

November 11, 2005 • Singapore
(co-located with ACM Multimedia 2005)

Sponsored by the ACM Special Interest Groups SIGMM & SIGGRAPH
Foreword

It is our great pleasure to welcome you to the 1st ACM Workshop on Multimedia Service Composition – MSC’05 in conjunction with ACM Multimedia 2005. Being a Brave New Topic at ACM Multimedia 2004 Conference the topic of multimedia service composition has recently gotten considerable attention. Service-oriented architectures are currently heavily researched and promise to introduce a maximum of flexibility and reusability of components also into multimedia applications. Composition of basic services to implement even complex workflows is a concept strongly discussed and researched in the Web community today. Web services are expected to take over an essential part of everyday’s responsibilities and their composition is necessary to extend their benefits to more and more complex tasks and personalized value chains. Besides the efficient provisioning and improved reusability of components, the move from data-driven to service-driven architectures promises to open up a whole new field of value adding applications dynamically built on top of basic components and flexibly adapted for different users.

Building on the broad interest that last year’s brave new topic session evoked, this workshop aims at assisting the multimedia community on their move from monolithic multimedia applications towards more flexible solutions. Such solutions could be provided either between content providers and clients or even peer-to-peer over the network. However, most Web-based concepts and constructs today suffer from being generally invariant to data types including the new datatypes that are being heavily explored in the multimedia community. Therefore, it is absolutely necessary to start exploring multimedia service composition problems as a specific topic, including components, meta-data descriptions and their mutual dependence within value-adding workflows. Raising awareness for the basic problems this workshop will help to pave the way towards a more service-oriented multimedia applications design. Bringing together researchers from the multimedia and the Web community and offering a platform for discussions in conjunction with the ACM Multimedia conference series will thus hopefully create synergies with mutual value.

Putting together MSC’05 was a team effort. First of all, we would like to thank the authors for providing the content of the program. We would like to express our gratitude to the program committee, who worked very hard and under a tight schedule in reviewing papers and providing suggestions for their improvements. Finally, we would like to thank our sponsor, ACM SIGMM, for their support of this workshop given that it is such a novel and challenging topic. We hope that you will find this program interesting and thought-provoking and that the workshop will provide a valuable platform to share ideas with other researchers and practitioners from institutions around the world.

Wolf-Tilo Balke
MSC’05 Chair
L3S Research Center Hannover, Germany

Klara Nahrstedt
MSC’05 Chair
University of Illinois at Urbana-Champaign, USA
Table of Contents

MSC 2005 Workshop Organization ................................................................. vi

Sponsors ........................................................................................................ vi

Keynote
Chair: W.-T. Balke (L3S Center)

• Building Large-Scale Multimedia Systems: Should We Use More SOAP to Clean Up Our Act? ................................................................. 1
  R. Zimmermann (University of Southern California)

• Towards Building Large Scale Multimedia Systems and Applications: Challenges and Status ................................................................. 3
  K. Nahrstedt (University of Illinois at Urbana-Champaign),
  W.-T. Balke (University of Hannover and L3S Research Center)

Session 1: Composition Frameworks
Chair: K. Nahrstedt (UIUC)

• Seamless Service Composition (SeSCo) in Pervasive Environments ................................................................. 11
  S. Kalasapur, M. Kumar (University of Texas at Arlington),
  B. Shirazi (Washington State University)

• A Distributed Scheme for Autonomous Service Composition ................................................................. 21
  S. Herborn (University of New South Wales),
  Y. Lopez, A. Seneviratne (National ICT Australia)

• Transparent End-Host-Based Service Composition through Network Virtualization ................................................................. 31
  S. Götz, K. Wehrle (University of Tübingen)

Session 2: Applications
Chair: W.-T. Balke (L3S Research Center),

• Resource-Aware Service Composition for Video Multicast to Heterogeneous Mobile Users ................................................................. 37
  S. Yamaoka, T. Sun, M. Tamai, K. Yasumoto (Nara Institute of Science and Technology),
  N. Shibata (Shiga University), M. Ito (Nara Institute of Science and Technology)

• Digital Media and Entertainment Service Delivery Platform ................................................................. 47
  C. J. Pavlovski, Q. Staes-Polet (IBM)

• Supporting Meetings with a Goal-Driven Service-Oriented Multimedia Environment ................................................................. 55
  K. A. Hummel, W. Jochum, S. Leitich, B. Schandl (University of Vienna)

Author Index ........................................................................................................ 66
MSC 2005 Workshop Organization

Workshop Chairs:  Wolf-Tilo Balke *(L3S Research Center Hannover, Germany)*
Klara Nahrstedt *(University of Illinois at Urbana-Champaign, USA)*

Program Committee:  Christian Becker *(University of Stuttgart, Germany)*
Hao-Hua Chu *(National Taiwan University, Taiwan)*
Peter Dolog *(University of Hannover, Germany)*
Xiaohui Gu *(IBM T.J. Watson Research Center, USA)*
Dejan Milojicic *(HP Labs Palo Alto, USA)*
Nalini Venkatasubramanian *(UC Irvine, USA)*
Matthias Wagner *(NTT DoCoMo Euro Labs, Germany)*
Klaus Wehrle *(University of Tübingen, Germany)*
Xing Xie *(Microsoft Research Asia, China)*
Dongyan Xu *(Purdue University, USA)*

Sponsors:  

[ACM SIGGRAPH]
Keynote Talk

Building Large-Scale Multimedia Systems: Should We Use More SOAP to Clean Up Our Act?

Roger Zimmermann
Integrated Media Systems Center
University of Southern California
Los Angeles, CA, USA
rzimmerm@imsc.usc.edu

ABSTRACT
This paper is a position statement given as keynote talk on the First Workshop on Multimedia Service Composition on specific challenges in the area of multimedia service composition.

Categories and Subject Descriptors

General Terms
Algorithms, Languages, Design.

Keywords
multimedia service composition, service-oriented architectures.

1. KEYNOTE ABSTRACT
The multimedia community has had great success in finding solutions to some of the most challenging multimedia problems. We have high-performance and scalable codecs, many protocols exist for the timely delivery of real-time streams, QoS mechanisms have been developed, media archives exist and are in use, just to name a few examples. Hence, the components now exist that allow us to build large-scale, distributed multimedia applications and systems.

In reality it remains a labor-intensive and hard problem to build these complex and sophisticated systems that provide many integrated functions. Often systems are constructed in a monolithic, stove-pipe fashion. Historically there have been good reasons for this. Multimedia processing was often very processing intensive and performance was of paramount importance. Furthermore, standards were evolving and interfaces not well defined. However, we are now entering an era where some of the basic problems have been (almost) solved, and the question emerges: Can we build systems in a more flexible and modular manner?

In this talk I reflect on my experiences with a number of projects and initiatives. At USC’s Integrated Media Systems Center (IMSC) we have worked over the last decade towards the vision of a real-time and multi-site distributed interactive and collaborative environment. Several prototype systems have resulted from the research, each integrating a number of components. Concurrently we have also pursued work with our Civil Engineering department on the design and implementation of a Web platform for the exchange and utilization of geotechnical information. Here Web services are used to access distributed data sources and processing modules across the Internet to enable complex simulations. Each of these projects has resulted in a number of lessons learned and they have put the spotlight on the challenges, advantages and disadvantages of the different approaches used.

2. KEYNOTE SPEAKER
Dr. Roger Zimmermann is currently a Research Assistant Professor with the Computer Science Department and a Research Area Director with the Integrated Media Systems Center (IMSC) at the University of Southern California.

He received his Ph.D. degree in Computer Science from the University of Southern California in 1998. He has co-authored more than sixty-five conference publications, journal articles and book chapters in the areas of multimedia and databases. He was the co-chair of the ACM NRBC 2004 workshop, the Open Source Software Competition of the ACM Multimedia 2004 conference and the short paper program systems track of the ACM Multimedia 2005 conference. He is on the editorial board of SIGMOD Disc, the ACM Computers in Entertainment magazine and the International Journal of Multimedia Tools and Applications. He has served on many conference program committees such as ACM Multimedia, SPIE MMCN and IEEE ICME.

His research activities focus on streaming media architectures, immersive environments, and multimodal databases. His work on streaming media has resulted in a number of distributed systems and prototype implementations. For example, the Yima architecture is the basis of the Remote Media Immersion (RMI) system which is designed for high quality, on-demand media distribution. Recently, he has been investigating scalable high-performance data recording platforms (project HYDRA) for collaborative, large-scale group communications. He has also worked on Web services based spatial data repositories for geotechnical information. Several patents have been filed on the developed techniques.

His industrial experience includes his participation in several large-scale projects while at Zühlke Engineering AG in Switzerland and consulting services for a number of companies.
Towards Building Large Scale Multimedia Systems and Applications: Challenges and Status

Klara Nahrstedt  
University of Illinois at Urbana-Champaign  
Urbana, IL 61801, USA  
klara@cs.uiuc.edu

Wolf-Tilo Balke  
University of Hannover and L3S Research Center  
Hannover, Germany  
balke@l3s.de

ABSTRACT
This paper is a position statement of the co-chairs for the First Workshop on Multimedia Service Composition on specific challenges in the area of multimedia service composition. The goal is to present and discuss problems that occur when considering building large scale multimedia systems via service composition. Today the realization of multimedia systems still heavily relies on building monolithic systems. Hence, building complex large scale multimedia systems is always a difficult, costly, time-consuming and challenging problem. Service-based architectures and the possibility to flexibly compose basic services to implement more complex workflows (or rather execution flows), as proposed in the Web and Grid communities, can provide a possible solution to this problem. However, due to the special characteristics of multimedia applications and the rich semantic structure of multimedia data and workflows, Web or Grid-based research results still cannot be readily applied. In this introduction-paper, we summarize challenges that need to be addressed and present a snapshot of the current state of the art towards building large scale multimedia systems.

Categories and Subject Descriptors

General Terms
Algorithms, Languages, Design.

Keywords
multimedia service composition, service-oriented architectures.

1. INTRODUCTION

Being a Brave New Topic at ACM Multimedia 2004 Conference [1], the topic of multimedia service composition has sparked considerable attention within the multimedia community. Composing basic building blocks in service-oriented architectures promise to introduce a maximum of flexibility and reusability of components into building even advanced multimedia applications. The first workshop on multimedia service composition in conjunction with ACM Multimedia 2005 provides a platform to investigate the necessary concepts in more detail and points to some related research, composition frameworks and prototypical multimedia applications. This paper is intended as a positional statement of the workshop co-chairs on the specific challenges that will have to be addressed by the multimedia community and for that purpose showcases some relevant related work.

Service-oriented computing and service-oriented architectures are concepts strongly discussed and researched in the Web and Grid communities today. With the advent of frameworks and languages to build and manage Web services and protocols to enable conversations between them, a lot of work (mostly driven by industry alliances) has been invested in standardization. Generally speaking Web applications can already now be flexibly modeled using services as basic building blocks and the market – especially in B2B interactions – is constantly growing. Beside the efficient provisioning and improved reusability of components, the move from data-driven to service-driven Web architectures promises to open up a whole new field of value-adding applications. These applications can be built on top of existing components and thus reuse individual services to form new and increasingly complex workflows in a time- and cost-aware manner. Moreover, new innovative business models for content-, service- and network-providers can be employed and used for mutual benefit.

Given the enormous development costs for large scale applications the multimedia community is currently on the move from monolithic multimedia applications to more flexible solutions. Extensive solutions are in the domain of data semantics. The multimedia community already provides sophisticated standards for media coding accompanied with meta-data descriptions (e.g. MPEG-7, MPEG-21). Nevertheless, useful concepts from Web services research on dynamically building complex applications and execution flows using semantically well-defined descriptions did not make a broad impact on multimedia systems development yet. On the other hand, Web-based models, concepts and constructs are invariant to new data types that are being heavily explored in the multimedia community. Therefore, the Web community tries to solve a much more general problem domain leading to a lot of problems that could possibly be avoided, if the problem space is limited down to a concrete domain. Therefore, the benefit of bringing together novel Web-based service-oriented concepts and the sophisticated handling and processing of multimedia data and annotations will be mutual.

In this introduction-paper we will outline some of the challenges for bringing service-oriented concepts into the multimedia domain and the status we see in this integration. In Section 2 we briefly present the multimedia application model and requirements on large scale multimedia systems. In Section 3 we discuss the system challenges and status, and Section 4 presents the semantic data challenges and provides an overview of the current state of the art. We conclude in Section 5 with a discussion of future directions in this area.
2. COMPOSITION REQUIREMENTS

Multimedia service composition is a composition process, where multiple services (e.g., retrieval, transcoding, display services), processing multimedia data,

- are connected via functional and data dependencies to create a new multimedia service (e.g., a video-on-demand service), and
- span over heterogeneous network and distributed system infrastructures.

To pose clear requirements on the composition process, first we need to have a well defined multimedia service model, which then will provide the atomic functional unit in the overall composition process.

2.1 Multimedia Service Model

Multimedia applications are generally flow-based applications, since their data usually are continuous streams (e.g., video and audio streams), i.e. dependent in time and space. This data-time and space dependency puts stringent timing and spatial constraints on the functional services that assist in processing and communication of the multimedia data in distributed environments. Moreover, quality constraints often need to be taken into account, adding another dimension. Hence, due to the rich semantic relations of multimedia data, and their time and space dependencies, functional services end up with rich dependencies, and it makes the building of large scale multimedia applications and systems truly challenging.

In summary, a multimedia service is a functional entity that assists processing and communication of multimedia data in timely and space-aware fashion. Each service includes the concept of time, space and dependency relation to other services that precede or follow the application service. The time, space, data and functional dependency relations among individual multimedia services form a service graph, which yields new multimedia services. To compose independently developed services, each service needs to have a clear description of its timing, spatial, semantic data and functional capabilities. This description is expressed via meta-data and published in order for other services to be discovered and used.

Multimedia services within the service graphs can be divided into input, output and intermediate/transformational services. An example of an input service is a video capturing service that captures video data from a camera and prepares the data in digital form (e.g., at 30 frames per second, with frame size of 640x480 pixels, 8 bits per pixel) for further processing and communication. An example of an output service is a display service that takes the video and displays its bitmap on the hardware display. An example of an intermediate/transformational service is a transcoding service that takes MPEG-4 encoded video and transforms it into H.263 coded video.

Service descriptions, expressed via metadata, can be categorized into media-specific and functional descriptions. Media-specific metadata describe multimedia data characteristics and their related Quality of Service (QoS) specifications such as frame rate, frame size, jitter, end-to-end delay, throughput, loss rate. Functional metadata describe functions embedded in services such as encoding transformation, retransmission, or filtering functions.

2.2 Requirements on System Infrastructure

To build a large scale distributed multimedia application, the underlying system infrastructure must provide a strong support across multiple protocol and service layers for the overall service composition process. The service composition process consists of four phases such as the service synthesis, discovery, selection and execution, and the requirements on system infrastructure manifest themselves as demands of each of the service composition phase onto the various processing and communication protocol and service layers:

- **Demands of Service Synthesis:** Large scale composed applications form abstract dependent service graphs, and the synthesis of these graphs may be created off-line via high-level programming tools. If this is the case, then the underlying system infrastructure needs to accommodate two types of mapping:
  (1) If the placement of physical multimedia services is already given through proxy service providers (e.g., IBM administers its own service proxy network [9]), then the service request needs to trigger mapping between the abstract service graph and the physical service network [9, 18].
  (2) If the placement of physical multimedia services is not given, i.e., services are stored in a central service repository (e.g., Gaia smart room uses a central service repository [44]), then the services in service graphs need to be requested, mapped and uploaded into the physical service network infrastructure.

- **Demands of Service Discovery:** In case a requested service is not available, discovery protocols of substitutable services, and eventually replication and/or customization of services may be needed. Furthermore, service discovery will require scalable content-addressable network [2], scalable lookup services [3], search for service paths [6], mapping of media-specific QoS requirements onto their own (e.g., transport packet-specific) system/network QoS representations, fitted towards system-based processing and communication services and data (e.g., connection setup service, flow control service, scheduling service) [10], and other protocols and services.

- **Demands of Service Selection:** In case multiple services of the same functional description are found, service selection is needed [6]. The selection needs to be then guided by the media-specific metadata and their corresponding system/network QoS metrics since they need to match across composed system services (e.g., rate, data format), and if they don’t match, intermediate multimedia services (e.g., transcoding) need to be requested and invoked to make the end-to-end service composition holistic. Moreover, even between services with exactly the same capabilities a selection has to be performed considering e.g. statistical parameters like the services’ expected availability.

- **Demands of Service Execution:** Timely multimedia service delivery can only be achieved, if the underlying systems and networks support resource management mechanisms, protocols and policies for performance-related Quality of Service (QoS) metrics such as deadlines, throughput, jitter, loss rate, and other time- and space-related metrics. These QoS metrics are part of the media-specific meta-data descriptions (e.g., end-to-end video delay, video jitter).
2.3 Requirements on Semantic Data

Due to the rich semantics of multimedia (e.g., MPEG-7 and MPEG-21 standards introduced a large set of metadata to allow for content-rich query in multimedia databases), large scale multimedia applications will end up with large amount of multimedia streams of different qualities and characteristics, hence with a rich set of metadata. The media-specific metadata must satisfy the following requirements:

- The multimedia metadata needs to be expressed in an easily readable (and machine understandable) form, such that services can address it, program it and manipulate it.
- The multimedia metadata needs to be organized, so that easy management and efficient searches can be executed.
- The multimedia metadata needs to be compatible so that other services such as Web services can utilize it for its inclusion and processing.

3. SYSTEM INFRASTRUCTURE CHALLENGES AND STATUS

Large scale multimedia applications and their service composition process will run on heterogeneous network and system platforms, hence we split the system challenges into network challenges and system challenges.

3.1 Network Challenges

To support service composition, future networks will need to assist in service synthesis, discovery and selection via an appropriate service path/graph establishment process, and in service execution via timely data delivery process. Within this composition framework, we will discuss two major challenges dealing with quality of service:

Challenge 1: QoS Mechanisms for Service Composition

The biggest challenge to support establishment and execution of QoS-aware service graphs is the inclusion of QoS-aware resource management mechanisms into the Internet protocol stack. The QoS-aware service graph establishment needs QoS mechanisms through the whole protocol stack. For example, the MAC layer needs priority mechanisms or time division multiplexing (TDMA) approaches, the network layer would benefit from QoS-aware routing, the transport layer could use rate-based flow control, selective retransmission, service and data differentiation, and the session should have timing and adaptive coding service.

Furthermore, even if some QoS mechanisms exist, often they are not accessible to higher layers such as middleware and application services (cross-layer design). For example, TCP/IP protocol stack hides fully any QoS mechanisms in physical and MAC layers, hence many researchers conduct end-to-end QoS measurements to estimate possible network resource availability and implicitly availability for multimedia service graphs on top of these networks [19,20].

3.2 System Challenges

The current status is that some QoS mechanisms, such as priority MAC scheduling, jitter control via queue management, exist at MAC and routing layers and are being utilized (e.g., CANS [12]), but many mechanisms and policies are not available in the lower network layers, e.g., QoS routing [17], or are not accessible to the higher protocol layers for its usage and control. Moreover, trade-offs within the QoS mechanisms have to be considered while managing multiple instances of multimedia data. There is a direct relation between the content flowing through a network and the available services managing that flow (especially in the presence of user-dependent trade-offs), as e.g. discussed in [31].

Challenge 2: QoS-aware Policies for Service Composition

The establishment and execution of service graphs deals with many dynamic situations since multimedia data changes its content over session time (e.g., during two hour movie viewing) and hence it changes its throughput, loss rate, and delay QoS characteristics. Handling this type of dynamic traffic requires QoS-aware adaptive policy management which is currently not present in the Internet protocol stack. The QoS-aware adaptive policy management must provide assistance in selection of physical service graphs, media types, new intermediate services, tradeoffs in case of resource shortage, and other assistance.

The current adaptive policy management frameworks are still mostly part of individual research projects. Most advances of the adaptive policy frameworks have been done in the area of Wide Area overlays and Peer-to-Peer networks [4, 5], where resources and services are being traded when finding service paths [6] and finding servers [7]. Adaptive and dynamic service composition frameworks have been explored in the CANS framework [12], OverQoS [13], SpiderNet [8, 14], and others. However, a lot of tradeoff policy management work is still missing in the wireless and pervasive environments, although some initial results are coming up [21].

4. SYSTEM INFRASTRUCTURE CHALLENGES AND STATUS

The current status is that some QoS mechanisms, such as priority MAC scheduling, jitter control via queue management, exist at MAC and routing layers and are being utilized (e.g., CANS [12]), but many mechanisms and policies are not available in the lower network layers, e.g., QoS routing [17], or are not accessible to the higher protocol layers for its usage and control. Moreover, trade-offs within the QoS mechanisms have to be considered while managing multiple instances of multimedia data. There is a direct relation between the content flowing through a network and the available services managing that flow (especially in the presence of user-dependent trade-offs), as e.g. discussed in [31].

Challenge 2: QoS-aware Policies for Service Composition

The establishment and execution of service graphs deals with many dynamic situations since multimedia data changes its content over session time (e.g., during two hour movie viewing) and hence it changes its throughput, loss rate, and delay QoS characteristics. Handling this type of dynamic traffic requires QoS-aware adaptive policy management which is currently not present in the Internet protocol stack. The QoS-aware adaptive policy management must provide assistance in selection of physical service graphs, media types, new intermediate services, tradeoffs in case of resource shortage, and other assistance.

The current adaptive policy management frameworks are still mostly part of individual research projects. Most advances of the adaptive policy frameworks have been done in the area of Wide Area overlays and Peer-to-Peer networks [4, 5], where resources and services are being traded when finding service paths [6] and finding servers [7]. Adaptive and dynamic service composition frameworks have been explored in the CANS framework [12], OverQoS [13], SpiderNet [8, 14], and others. However, a lot of tradeoff policy management work is still missing in the wireless and pervasive environments, although some initial results are coming up [21].

3.2 System Challenges

The service composition operations (synthesis, discovery, selection and execution) also rely on system resources such as processors, memory, and disk that need to be appropriately allocated and coordinated in order to assist in timely service composition. We discuss three major challenges.

Challenge 1: Broad Availability of Multimedia OS

For multimedia services to perform according to their media-specific descriptions, each computing node should have multimedia operating systems that would monitor, allocate, schedule and manage local resources in a time and space-aware fashion. This means that to deliver multimedia streams in timely fashion, we need deadline-based scheduling algorithms at the processor and disk level. Furthermore, we need time and space-based monitoring, prediction and management algorithms to deal with the dynamic characteristics of multimedia streams that are processed and communicated at the various computing nodes.

The current status is that various multimedia scheduling algorithms for processors and disks have been explored (e.g., [39, 40, 41]), but none of them are part of current operating systems (e.g., Linux or Windows XP). The benefit analysis shows clearly that it would be of great performance advantage to have any of the researched soft-real-time scheduling algorithms, however, due to significant cost of embedding them into the general purpose OS and due to relatively small video traffic on our computing nodes, inclusion of deadline-based schedulers will have to wait.

1 It is important to stress that multimedia service composition uses Internet protocol stack across wired and wireless networks. The wireless networks, especially 802.11 networks, represent a very difficult infrastructure to support deterministic or statistical QoS guarantees for multimedia service delivery [37, 38].
Challenge 2: Automated Service Graph Establishment
One of the major difficulties with current composed multimedia services is that one has to manually
(a) setup all physical service components in the distributed infrastructure or in a central repository,
(b) provide a static service dependency path among physical services,
(c) ensure that sufficient resources are available for composed service, and
(d) invoke the appropriate physical service path for timely multimedia data delivery.

So the challenge is to automate the overall composed service graph establishment or at least part of this process. This means that there is a strong need to
(1) provide automated high-level programming tools that would assist in creation and synthesis of abstract service graphs,
(2) provide automated service discovery and selection,
(3) provide automated mapping and matching between abstract service graphs and physical distributed service infrastructure, and
(4) provide automated QoS-aware service routing and fault-tolerant invocation of service graphs.

The current state is that pieces of the establishment process have been automated. For example, there are limited programming tools that allow for abstract service synthesis, and creation of service graphs such as the QoSTalk tool [42, 47]. There is also an extensive body of work on automated service discovery and selection including Chord P2P lookup service [3], media proxy finding service [6], QoS-aware discovery service [43, 16], and others. Assistance for service mobility, multicast, anycast and overall service composition can be obtained via the Internet Indirection Infrastructure (I3) framework [50, 51]. The automated mapping and QoS-aware service routing have been explored in SpiderNet [14], in service multicast framework [10, 18], and via the QoS Compiler framework [48, 49]. Fault-tolerant invocation of service has been explored in [15].

Challenge 3: Understanding and Dealing with Heterogeneous Devices
The large scale multimedia applications will run on very diverse devices which differ in processor power, memory capacity, network throughput, network connectivity, energy efficiency, distance accessibility, mobility, security, and other attributes. Many of these devices are connected via 802.11, Bluetooth, or 3G networks that differ in their range, MAC protocols, QoS support, and other characteristics. Many of these devices range from running a single service (e.g., sensors, iPAQs) to multiple services (e.g., laptops, PC servers). The integration of these different devices is not very well understood. Hence, the multimedia service composition challenges are
(a) scalable algorithms to manage a large number of devices (hundreds of sensors or mobile devices),
(b) dynamic addressing of devices and content in case of mobility,
(c) fast hand-shake in case of service discovery and selection,
(d) timely and scalable delivery, and many others.

The current state is that a lot of research has been done in smart rooms and other ubiquitous environments where many small and mobile devices reside. However, little has been done in integrating the multimedia pervasive computing research into large scale wide area distributed infrastructures. Few examples show interesting results in some of the settings: scalable and mobile delivery in smart rooms was explored is the Gaia middleware system [44, 45], scalable content-addressable network is discussed in [2], and seamless hand-shake is presented in [46].

In summary, the research community explored some of these system challenges in simulated or controlled environments on community networks such as smart rooms or Planetlab, however, unless these research results are integrated with Web or Grid system services, which do have much broader usage due to large commercial or defense backups, multimedia service composition will have difficulties when building large scale systems.

4. SEMANTIC DATA CHALLENGES AND STATUS
The ultimate goal of Web services is to provide interoperability for a possibly large number of applications by providing a generic syntax and interface to service components. Standardized languages like SOAP and WSDL [27] for communication between services and the description of service interfaces are based on XML and also rely on XML for representing the data types involved. In this section we will consider multimedia service composition challenges from their multimedia semantic data and service description point of view.

4.1 Modeling Compositions
While for simple interactions or conversations with services the SOAP and WSDL standards already provide a good foundation, the problem of composition is somewhat harder. Compositions deal with the implementation of complex applications that are in turn offered as new composite services. The component services that are invoked in this application are generally different (atomic or composite) services usually offered by multiple providers. The sequence and conditions in which a Web service invokes other services to perform a certain task together is often referred to as orchestration. As we already discussed before, the basic problems in performing such compositions occur in different steps during the composition process:
• Service synthesis
• Service discovery
• Service selection
• Service execution

These service composition operations apply to multimedia service composition as well as discussed in Section 2.1, and Section 3. From the semantic modeling point of view they yield four distinct modeling challenges.

Challenge 1: Modeling of Service Synthesis
The first step in the service composition process is the service synthesis, which builds from basic and independent components the synthesis of a suitable invocation flow; a task very similar to
specified an intended workflow describing the application. Though sometimes results from artificial intelligence research like goal planning (see e.g. [33]) might be applicable, for most applications the synthesis has still to be performed manually (e.g. by specifying sequence or activity diagrams of alternative invocation flows) or at least supervised semi-automatically. An example for such a specification of alternative invocation flows is the composite services description language (CSDL) used in the eFlow system [11]. Here a process schema for a composite process is modeled by a graph, which defines the order of execution among the nodes in the process. Composition graphs in eFlow can include service, decision, and event nodes, where service nodes represent invocations of services; decision nodes specify alternatives and rules controlling the execution flow, while event nodes are used to send and receive notifications with respect to other services. If different steps can be composed to a single service satisfying a subgoal, this subgoal can of course also be used by different composition schemes not necessarily having the same overall goal.

**Challenge 2: Modeling of Service Discovery**

After one or more correct invocation flows for the application have been determined, suitable services for composition have to be found during the service discovery. While it is a general problem to figure out what functionality a service generally provides (also state of the art standards like UDDI [28] only amount to simple keyword matching during discovery), there are more questions to address. For a running application it is essential to ensure that services involved will be able to interact properly, in other words that they are compatible with each other (see e.g. the discussion in [32]). Services can be incompatible for a variety of reasons. First, there is the general semantic incompatibility of the functionality (e.g. an encoder service obviously cannot perform scheduling). Here it is important to notice that services could also perform more specialized tasks only (and thus would need additional services in the composition), or might be able to even perform more complex tasks, whose functionality may not be fully needed, but does not hurt either. Since in restricted domains there might at least exist a common understanding or even a standardized classification of what will be expected from a specific service, this problem can sometimes be put aside.

Second, incompatibility might arise from mismatches in interfaces (as e.g. defined by WSDL) or the type of messages they can exchange. Usually also this problem can be quite easily checked and obvious mismatches could be avoided. More challenging is a mismatch in the dynamic behavior of services, e.g. possible deadlocks during execution given certain message exchanges. Formal representations of a services behavior as given by e.g. Petri nets or state machines, thus have to be reasoned about to guarantee a correct application (see e.g. [25]). Compatibility is also closely related to another problem in flexibly composing applications, substitutability. Substituting a previously used service by a new one is often necessary, for instance when a specific service is unavailable due to network or server problems.

**Challenge 3: Modeling of Service Selection**

Service selection mainly poses the problem of choosing adequate services that have been discovered in the previous step. There is a difference between trade-offs induced by limitations in the capabilities of the set of services to choose from (these trade-offs usually do already occur at discovery time and sometimes are handled cooperatively with respect to the user respect, e.g. [23]), or differences in the functionality of individual services even if all services support the basic task. There may be differences in many characteristics like service costs, quality of service guarantees, or expected service availability. Although this decision for the individual service can often be made based on an individual user's preferences, see e.g. [24], or a group profile for a certain application incorporating rules like e.g. always choosing those services offering the best quality of service guarantees, the impact on the composition is hard to assess. Choosing specific services e.g. optimizing costs at some stage in the composition process can lead to problematic situations later in the invocation flow. The problem thus becomes a multi-objective problem that has to be solved before the instantiation of a specific composition can be offered. Solving this problem is, however, usually possible in acceptable time, because there will only be a limited number of discovered services and their possible characteristics. On the other hand, dynamically putting together compositions (e.g. if unavailable services have to be substituted) is often difficult to facilitate given restrictive quality of service constraints.

**Challenge 4: Modeling of Service Execution**

The main problem in the execution is usually in the controlling and monitoring of the application and its characteristics. Applications may be on a simple best effort basis where the failure of individual components might not amount to serious problems, but can also range to commercial services that will have stronger demands, for example due to putting penalties on quality of service violations. Thus also the execution models range from simple frameworks trying to substitute failed services as quickly as possible, to full-fledged transaction models, e.g. the XML-based model discussed in [29]. Defining an adequate set of control parameters, monitoring throughout the composite application (even if certain components should be replaced dynamically) and proactively managing undesired situations during service execution will thus demand a lot of attention. Current implementations like the business process execution language for Web services (BPEL4WS), see e.g. [35], do already allow for a limited amount of exception handling [34], but are still far from what is needed to control complex multimedia applications.

### 4.2 Meta-data for Compositions

As we have pointed out in the previous section, the production of viable compositions basically can be managed as a planning problem. Though some basic composition patterns could be designed manually, given the variety of different technical devices and thus different implementations of services will definitely need some automation. If Web service compositions have to be performed automatically, a general understanding of the terms involved (e.g. descriptions of service capabilities or the compatibility of certain data types) has to be shared. This sharing of common vocabularies and the benefit brought by machine-understandable meta-data has evolved into a large research area within the Web community, called the Semantic Web.

Semantic Web technologies focus on managing structured collections of information, often together with sets of inference rules that can be used for automated reasoning. The challenge is to provide a language that expresses both data and rules for reasoning about the data and that allows rules from any existing knowledge-representation system to be exported. Any composition engine that has to compare or combine information across two or more services to be combined, needs to know the exact meaning of terms used e.g. for description and if they refer to the same or at least similar concepts. A powerful solution to this problem is
given by **ontologies** providing structured collections of information and formal definitions of the relations among terms. Advanced types of ontologies provide a **taxonomy** and a set of **inference rules**. The taxonomy defines classes of objects and relations among them. For multimedia service composition we identify two major challenges that will need to be solved to make service descriptions easily expressible, organized and compatible.

**Challenge 1: Multimedia Service Taxonomies**

Suitable taxonomies are a basic problem that has to be solved independently for each application domain, although some basic concepts might be transferrable. In multimedia applications there are already some first approaches towards building taxonomies, see e.g. [1]. However, it remains to be seen, if the sophisticated media-specific descriptions of multimedia data types as e.g. given by the MPEG-7 standard can provide a suitable taxonomies or how they have to be extended. Furthermore, also important concepts in multimedia applications like QoS parameters and their interdependence needs to be modelled. Ontologies thus can enhance the functioning of composition engines by improving the accuracy of service capabilities looking for precise concepts instead of ambiguous keywords. More advanced applications will use ontologies to relate the information on services or data types to the associated knowledge structures and inference rules.

**Challenge 2: Semantic Ontology Multimedia Language**

The development of an ontology for Web services using the **Semantic Web ontology language OWL** based on the DAML+OIL standards [26], has led to the creation of the Ontology Web Language for Services (OWL-S, formerly DAML-S). OWL-S [36] is a Web service ontology developed in OWL, a description logic-based language for describing content. OWL-S has well-defined semantics and can be used to describe the process model of a Web service. The challenge is if this type of description logic-based language could lead to a Semantic Ontology Multimedia Language to allow for multimedia content description and how it will mesh with the current MPEG-7 and MPEG-21 standards that do define media-specific metadata. Unlike descriptions in WSDL that provides no means to represent the semantics of defined operations and associates messages, OWL-S provides a language for specifying functional descriptions in the form of preconditions and effects of operations together with semantic types for both input and output values of the service. The definitions of all semantic concepts used (what for instance an effect is meant to be) then can be made available using a **uniform resource identifier** (URI) and can be shared (and more or less understood) by different services. Another important aspect is that in this way also the output (types) of a service can be correctly interpreted. OWL-S is an OWL ontology featuring three parts:

- a profile,
- a process model,
- and a grounding.

The profile refers to the service capabilities, whose description is needed prominently for discovering services that are capable of performing a requested task in compositions. How to match different descriptions of essentially similar or even identical capabilities, however, remains a largely unsolved problem. The process model provides an insight into how the service works and thus enables the invocation (and to a certain degree also monitoring and recovery) in actual compositions. The grounding finally maps constructs of the process model to detailed specifications of message formats, protocols, etc. For a more detailed description of the profile, process model and grounding sub-ontologies see e.g. [36]. However, even considering such promising frameworks like OWL-S, the Semantic Web community is still a long way from the goal of automated Web services composition, and the same applies to automated multimedia service composition. Beside the fundamental planning problem (that has already been researched in AI for quite some time, too), the representation is still not rich enough to suffice for complex compositions in composite processes. It is also questionable, if the concepts of preconditions and effects are sufficient for deriving service guarantees like needed in most multimedia applications. On the other hand, especially in terms of the planning building multimedia applications seems generally not as hard a problem as composing all purposed business processes out of arbitrary Web services. Having strongly typed data, structured descriptions and quite often to some degree predefined workflows, it remains to be seen, if current multimedia standards can be used to augment service ontologies strong enough to tackle the composition problem for large scale real world applications.

5. **SUMMARY**

In this paper we have outlined service composition challenges that need to be solved for large scale multimedia applications to become reality. We have addressed system infrastructure challenges as well as semantic data challenges. From the current state of the art it follows that the multimedia community has already progressed far in terms of understanding the underlying system infrastructure. This includes topics like multimedia streams setup and delivery, time and space-aware (QoS) data specification, timely delivery services and protocols, as well as monitoring and management services that assist in handling of independent resources. That means the community can handle single services quite effectively and has sufficient means to control their execution.

However, we are still missing a lot of service-based models, frameworks and implementations that would provide timely dependency management during the four main steps of dynamic interaction with services for composition tasks: synthesis, discovery, selection, and execution of composed services, and many other capabilities especially in terms of controlling a composed execution flow (especially with respect to quality of service) and dynamically adapting to failures, e.g. the problem of timely substitution of failed services. Given that the challenges can be overcome service composition could become a broadly used concept and software engineering pattern in building even large scale multimedia applications.

The presented challenges clearly outline many future directions that service composition research needs to explore. We conclude with two further samples of future questions that may be of interest to the community:

(a) Can services decouple and express QoS-aware adaptive policies for external management? If yes, how do we match different adaptive policies to enforce stable multimedia delivery? How do we coordinate different adaptive techniques (e.g., linear control, fuzzy control) in end-to-end composed services over heterogeneous devices and networks?

(b) Can MPEG-7 be used to define an upper ontology for content retrieval purposes? What other upper ontologies are needed e.g. for QoS guarantees, service capabilities, capabilities of technical devices, etc.?
6. ACKNOWLEDGEMENTS

As co-chairs of the workshop on multimedia service composition (MSC’05) the authors would like to thank ACM SIGMM for sponsoring the workshop in conjunction with ACM Multimedia 2005. We are also grateful to all members of the program committee, Christian Becker, Hao-Hua Chu, Peter Dolog, Xiaohui Gu, Dejan Milojicic, Nalini Venkasubramanian, Matthias Wagner, Klaus Wehrle, Xing Xie, Dongyan Xu, and the keynote speaker Roger Zimmermann who helped to make the workshop possible.

7. REFERENCES


Seamless Service Composition (SeSCo) in Pervasive Environments

Swaroop Kalasapur and Mohan Kumar
Dept of Computer Science and Engineering
University of Texas at Arlington
Arlington, TX - 76013, USA
{kalasa, kumar}@cse.uta.edu

Behrooz Shirazi
School of EECS
Washington State University
Pullman, WA - 99164, USA
shirazi@eecs.wsu.edu

ABSTRACT

It is a challenging task to develop applications and systems that cater to the needs of ever increasing multimedia applications. Additionally, in pervasive computing environments, multimedia data needs to be delivered to heterogeneous devices with varying capabilities over a variety of communication channels. The objective of this research is to dynamically compose services by effectively utilizing the collective capabilities of resources available to deliver multimedia. Existing schemes provide composite solutions to multimedia applications, work either on a centralized system or assume that the environment is ad-hoc in nature, resulting in additional overheads during composition. Further, some of the existing composition schemes are an extension to discovery, resulting in a discover + match + coordinate scheme. Such schemes would not be effective in dynamically changing environments, due to the uncertainties involved. In this paper, we present a novel composition scheme, called Seamless Service Composition (SeSCo), that operates on automatically configurable resource hierarchies for discovery and composition. SeSCo attempts to weave necessary services by utilizing available individual services seamlessly. Experimental results demonstrate the superiority of our scheme over existing broadcast based schemes.

Categories and Subject Descriptors: C.2.4 [Distributed Systems]: Network Operations

General Terms: Algorithms, Design, Theory

Keywords: Service oriented Computing, Service Composition, Pervasive computing, Multimedia Delivery

1. INTRODUCTION

With the availability of a variety of personal devices capable of supporting a number of features, seamless multimedia delivery to mobile users is a challenging problem. Although the devices users carry have considerably increased capabilities, due to other limitations such as available network bandwidth, power and memory limitations it is difficult to employ classical client-server kind of architecture for multimedia applications. We make a case that it is important to employ the capabilities of resources available en route the delivery path to accomplish effective multimedia delivery. The capabilities of available resources need to be utilized not only to lift the burden in terms of processing, memory and communication from resource poor user devices, but also to ensure that the multimedia data is being delivered in a format acceptable to the user device.

To accomplish the task of utilizing the collective capabilities of the available resources, a Service Oriented Architecture (SOA) [12, 13] is most suitable. SOA offers a layer of transparency by abstracting the available resources as services to be used by users and other applications. The technique of combining two or more services together to achieve a complex goal is known as service composition. Research in service based paradigm has been in the lime light in recent years due to the popularity of the web services [10] paradigm. Service composition mechanisms [2, 4, 15] built to work on the web services [3] are not directly applicable to the dynamic target environments where multimedia is envisioned to be delivered. Web services operate under the luxury of being relatively stable, with static information such as its location, resource availability, etc. However, in pervasive computing, multimedia needs to be made available to users under a variety of scenarios, in the face of dynamisms in location, resource availability and such important factors.

Essentially, there are two types of service composition mechanisms [3] in practice. In static composition, the elements that would go into the composed service are known a priori, whereas in dynamic composition the required individual services are located on demand [3]. The static mechanism of composing services is popular in the web services arena [2], where the required resources are well defined, and where usually there needs to be some business agreement among the service providers. It has been argued [3] that due to the dynamic nature of pervasive computing environments, dynamic schemes are best suited to meet the needs of applications in pervasive computing. Irrespective of the mechanism followed, when it comes to composition itself, traditionally the mechanism used is an extension to service discovery. The requirements are specified in the form of a template, detailing the needs in terms of service elements. The composition process is responsible for locating candidates that can fill in the place holders within the request template, and
2. SERVICE MODEL

The pervasive computing paradigm is leading us to intelligent environments. The devices around us are capable of not only performing a predefined task, but also are equipped with computing and communicating capabilities. New architectures will have to support follow me type of multimedia rendering, to support user and resource mobility. While the delivery of multimedia data can still be considered as a communication between two or more remote sites, the configuration of each of the remote sites itself is going to be dynamic. There will be a number of variations in the type of devices used to render multimedia. Due to user and resource mobility, the type of devices on which media will be delivered to the user can change during the course of a single multimedia session. Figure 1 gives an example of the pervasive multimedia delivery architecture. Based on all the changes observed at the user sites, the components involved in the session need to be recomputed and used on the fly. By modeling all the involved resources as services, and by employing a dynamic service composition mechanism, such dynamism can be handled effectively.

In building such service oriented environments, we employ the event oriented middleware called PICO [9], designed to accommodate a variety of resources and to utilize their features as services. The basic constructs in our architecture are (i) *camileuns* - abstract representation of devices, (ii) *delegents* - software entities representing the device features as services, and (iii) *communities* - logical organization of a number of delegents working toward a common goal.

2.1 PICO Architecture

Each device/resource within the pervasive computing environment has certain features that are represented as services to the external world. Similarly, there are a number of characteristics associated with each resource. *Camileuns* capture these features and characteristics in their abstraction. Camileuns are represented by the tuple $C = \{H, F\}$, where $H$ is the set of characteristics such as the memory available, processing power, type of communication supported, etc. The set $F$ represents the functional features of the resource, for example, the feature set of a printer can include B/W printing, color printing, etc. The features identified during the camileum abstraction are modeled as services by designing *delegents* to represent the features in the network. Delegents are essentially software agents, de-

- Locality of services
- Quality of composition
- Semantics and
- Mobility of users and resources.

The rest of the paper is organized as follows. In Section 2, we present the service architecture of our scheme, detailing the service model, the request model and the aggregation model. In Section 3, we present the hierarchical service overlay that is generated based on the capabilities of the device. Section 4 details the composition scheme used in our approach. The properties of our scheme to support a number of issues critical to service composition in pervasive multimedia environments are discussed in Section 5. Results of our experiments illustrating the improvement of our schemes over traditional "discover + match + coordinate" approaches are presented in Section 6. We also present the details of our prototype multimedia system built to utilize the different multimedia services that are available around the user to accomplish the task at hand. In section 7, we present the conclusions and future research directions for our work.
scribed by the tuple $D = \{M, R, S\}$, where $M$ is a set of modules that are used to build the delegent, $R$ is a set of rules, describing the transition of control from one module to the other, based on an observed event, and $S$ is a set of services that the delegent provides.

To accomplish a complex task, two or more delegents are grouped together into a community. The community is described as the triple $P = \{U, G, E\}$, where $U$ is the set of members or the set of delegents that build the community, $G$ is the set of goals that the community achieves, and $E$ is the set of operational characteristics observed within the community, such as response rate, current load, cost, etc.

2.2 The PICO middleware stack

The PICO middleware is a layered architecture, with different operations being achieved at different layers in the middleware stack. The reference middleware stack is shown in Figure 2. Based on the capabilities of the device, there can be three different versions of the stack, which essentially differ in the operations they can perform. For highly resource constrained devices, a minimal version of the middleware is installed, with a reduced stack. The minimal version of the middleware is capable of advertising itself and participating in service discovery, but is not equipped to conduct more complex tasks such as service aggregation and composition. The second version, although is complete in its operation, differs in the fact that it is intended to be installed on devices that are possibly mobile, such as PDAs, laptops etc. The complete version of the middleware is installed on resource-rich, infrastructure based devices such as PCs, servers and such devices to perform complex tasks including discovery and composition. Resource-rich devices are also employed to host delegents (that act as a proxy) on behalf of their resource poor counterparts. Based on this distinctive versioning system, we present in more detail, the auto-generation of the device hierarchy and its applicability in Section 3.

Since the focus of this paper is related to the service layer, we briefly present the components that build the service layer. Figure 3 shows the details of the service layer. The service layer in the middleware stack is responsible for basic service related operations such as service advertisement, discovery and composition. The advertisement manager periodically sends its own advertisements and also collects and responds to external advertisements and requests. The service aggregator is responsible for storing the received service advertise-
egin{figure}[h]

![Figure 2: PICO middleware stack.]

The PICO middleware stack.

Figure 3: Service layer components.

2.3 Service representation

Each of the delegents offer one or more services to be used in the pervasive environment. In our scheme, we employ a directed graph based approach to represent different services, and the service requests. Essentially, each service can be described as a network element performing transformation of one form (or a set) of input into another form (or set). We represent each of the services with a service graph $G_S$, which is a directed, attributed graph described as $G_S = \langle V_s, E_s, \mu_s, \xi_s \rangle$, where the vertex set is represented as $V_s$, the edges are represented with the set $E_s$, and $\mu_s$ is the vertex attribute function, responsible for attaching the different attributes such as the name and location of the service, the cost of utilizing the service, the quality of the represented service, etc. The edge attribute function $\xi_s$ takes care of the edge attributes such as the type of parameter, acceptable and expected size of the parameters, etc. An example service graph is shown in Figure 4.

Each user application, that requires a number of services from the external world, is represented as a request graph $G_R$, which essentially details the plan to accomplish the goal of the application. The request graph $G_R = \langle V_r, E_r, \mu_r, \xi_r \rangle$, is a directed, attributed graph, and has the vertex set $V_r$, representing the individual service elements that are needed to accomplish the goal. The edge set $E_r$ contains details
about data exchange among the services. The vertex attribute function \( \mu_r \) takes care of associating the attributes on vertices, such as the name of the service, the expected quality of the required service, the acceptable cost for the service, etc. The edge attribute function \( \xi_r \) is responsible for populating the attributes along the edges, which include the type and size of data being exchanged, the messaging format, etc. The request graph is used as a template during composition to identify the suitable services to accomplish the requirements. An example request graph is shown in Figure 4. In Section 5, we present the service composition mechanism, based on the service graphs \( G_S \) and the request graph \( G_R \).

![Figure 5: Request for file-reader operation.](image)

### 3. HIERARCHICAL SERVICE OVERLAY

In a typical pervasive computing environment, there are a number of devices with varying capabilities. Traditional systems have treated the environments either with a centralized architecture, where an individual or a group of entities are in control of all the resources involved, or in an ad-hoc architecture, where each element is autonomous, and needs to discover the required resources on its own. Typical pervasive computing environments are envisioned to be comprised of a variety of resources. This variety in the involved devices means that there are going to be a number of devices that are constrained in one or more of the much needed resources such as memory, processing power, communication bandwidth, etc. Also, there are going to be a number of other devices such as PCs, Laptops, etc., which have higher resource availability. In a cooperative world, it is ideal that devices with abundant resources assist their resource poor counterparts in accomplishing the desired task [6, 7]. Due to the dynamism involved, especially user and resource mobility, it would be difficult to have a tightly coupled relationship among the proxies. To overcome this problem, in the proposed hierarchical structure, relatively resource-rich devices accommodate their resource poor counterparts, by assisting them in performing their operation. The assistance is sought in terms of process offloading, communication, and service discovery and composition. We describe the hierarchical organization with an emphasis on service related operations such as advertisement, discovery and composition. Based on the profile of the device, we classify each of them into one of the four device levels (L0 - L3). Devices which have no additional facilities for software installation are categorized into Level-0. Such devices need a more resourceful device to be associated with them to be able to contribute in cooperative operations. An example of such a device would be a legacy printer, which is connected to the computer. The printer itself will be categorized as a Level-0 device, and its features are exported as services by implementing a delegent which is hosted on the computer. Level-1 devices are capable of hosting custom software, essentially one or more delegents which export the functionality of the device as services in the network. Due to resource limitations Level-1 devices can not accept to act as proxy entities for other devices. Examples of Level-1 devices are lower end PDAs, sensors such as Motes, cellular phones, etc. Level-2 devices have higher resource availability as compared to Level-0 and Level-1 devices. They have sufficient capabilities in terms of memory, processing power, etc., to host a number of services through a number of delegents. Level-2 devices can also act as proxies for devices lower in the hierarchy. But, Level-2 devices are associated with a degree of dynamism such as mobility, associated with them. This dynamism separates Level-2 devices from Level-3 devices. Examples of Level-2 devices include laptops, high-end PDAs and cell phones, etc. Level-3 devices also have sufficient capabilities to host a number of services and to act as proxies for other devices. Also, Level-3 devices bring in a degree of guarantee in terms of resource availability, and duration of availability. Examples of Level-3 devices include PCs, servers, grid systems, etc.

Table 1. summarizes the device classification into one of the four levels. Using the generated classification, we arrange the involved devices into a service overlay hierarchy, with higher devices located higher up in the hierarchy, forming a hierarchical relationship with other devices lower in the hierarchy. In this set up, lower level nodes are allowed to exploit the resources of higher level nodes. We utilize this disparity in resources to achieve service composition at higher level devices, when the devices lower in the hierarchy can not accomplish the desired goals on their own.

### 3.1 LATCH protocol

We now describe the process of creating the hierarchy based on the device level, assigned according to the procedure presented in Table 1. Each device, upon becoming active sends out advertisements, that includes the level assigned to the device. The process of sending out advertisements and inspecting the incoming advertisement messages is done by the service layer of the PICO middleware stack. All the devices within the range can receive and inspect these advertisements.

When device \( A \) receives an advertisement message from device \( B \), \( A \) becomes aware of the level of \( B \) (\( \alpha(B) \)). If \( \alpha(B) \) is higher than \( \alpha(A) \), then \( B \) sends a \( \text{LATCH}_\text{REQ} \) message to \( A \). If \( \alpha(A) \) is less than \( \alpha(B) \), then \( B \) sends a \( \text{LATCH}_\text{INVITE} \) message to \( A \), inviting it to attach itself as a child. If \( \alpha(A) = \alpha(B) \), then they add each other as siblings by sending \( \text{LATCH}_\text{SIBLING} \) messages. The resulting structure from this procedure is an overlay, with Level-3 devices forming the highest layer of the hierarchy and Level-0 devices being in the lowest level. The \( \text{LATCH} \) protocol used to form the device hierarchy is presented in Figure 5. An example hierarchy formed through the \( \text{LATCH} \) protocol is shown in Figure 6.

Once the hierarchy is formed, the liveness of involved devices are periodically checked with a \( \text{LATCH}_\text{HELLO} \) message. When new services are added, the updates are sent up the hierarchy by using the \( \text{LATCH}_\text{ADD} \) message, and the sensors which are no longer available are updated using the \( \text{LATCH}_\text{REM} \) message. In building the hierarchy based on the device profiles, we
<table>
<thead>
<tr>
<th>LEVEL (α)</th>
<th>Middleware Version</th>
<th>Features</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NONE</td>
<td>Features exported through delegents on Level-2 or Level-3 devices, No native personalization support</td>
<td>sensors, legacy printers</td>
</tr>
<tr>
<td>1</td>
<td>Minimum</td>
<td>Community member, can not be a proxy, possibly mobile</td>
<td>Cell phone, Mote sensors, smart printer</td>
</tr>
<tr>
<td>2</td>
<td>Complete</td>
<td>Community member, leader, Can act as a proxy, possibly mobile, Resource rich</td>
<td>Laptop, PDA</td>
</tr>
<tr>
<td>3</td>
<td>Complete</td>
<td>Community leader, can act as a proxy, not mobile, Resource rich</td>
<td>Servers, PCs, clusters, Grids</td>
</tr>
</tbody>
</table>

Table 1: Device classification chart.

make an assumption that the network under consideration forms a connected graph. In other words, there exists at least one path of communication between any two resources in the environment.

The hierarchical overlay based on the device profile ensures that there is at least one device with higher resources available for all the resource constrained devices. As a result of forming the hierarchy, any resource within the environment is at a distance of at most 2 hops away from a Level-3 device within the service overlay. In practice, there will be a number of devices that will be strategically placed in different locations within the pervasive environment. Such devices will be networked by an experienced administrator, or will be governed by advanced control devices such as gateways and access points. Such devices will also be a part of our architecture, and will assist the devices with a higher degree of dynamism. For the part of the network which is pre-configured, there is a flexibility to attach additional context information such as location information, with varying levels of granularity. Such information will be useful in supporting transparent service access by the users. In this paper, we assume that all the devices which are a part of the infrastructure are mapped as Level-3 devices, and have knowledge of each other, either directly or through a distributed lookup algorithm. Since the Level-3 devices usually are a part of the infrastructure, we assume that the time it takes for all the Level-3 devices to communicate among each other is a constant $T_k$. For other devices, such as the ones being carried by the users, the formation of the hierarchy is based on the communication range of the device. We assume that the parent-child relationship is governed by the communication range that the device possesses.

4. SERVICE COMPOSITION MECHANISM

A novel feature of the service composition mechanism in [8] is the ability to weave the required services by utilizing those available. In addition, SeSCo incorporates extensions to overcome challenges due to heterogeneity and mobility for seamless multimedia delivery in pervasive computing environments.

In general, a request is expressed by $R = \{s_1, s_2, \ldots, s_n\}$, a set of required services. In traditional composition systems, the result of a successful composition $C = \{c_1, c_2, \ldots, c_m\}$, with $|R| = |C|$, where $|R|$ and $|C|$ are the size of the request set and the composed service set respectively. Essentially, in traditional composition systems, there needs to be a one-to-one correspondence between the services specified within the request and those selected by composition. If the discovery mechanism fails to locate a suitable match for a service $s \in R$, then the composition would either fail or enter an extended discovery mode.

When both users and resources are mobile, and the involved
devices are resource constrained, it is easy to see a situation when a suitable match for a service specified within the request might not be available. SeSCo dynamically constructs a service \( \mathcal{C} = \{\mathcal{C}_i\}_{i=0}^m \) from the available services, so that \( \mathcal{C}_i \approx \mathcal{S}_i \in \mathcal{R} \).

### 4.1 Service aggregation

In SeSCo, each individual service is expressed as a directed attributed graph \( G_S = \{V_S, E_S, \mu_S, \xi_S\} \). Likewise, requests are also expressed as a directed attributed graph \( G_R = \{V_R, E_R, \mu_R, \xi_R\} \). As described in Section 3.1, at the time of registration, each device sends the service graphs for all the services that are available locally to its parent. Each service graph, that describes the corresponding service transformation from one parameter type to the other is aggregated at the parent based on the I/O parameters for the service. Essentially, at each parent, a parameter graph \( G_P = \{V_P, E_P, \mu_P, \xi_P\} \) is maintained. The vertex set \( V_P \) represents the set of unique parameters that can be handled in the local service zone either as an input parameter or as an output parameter. The set \( E_P \) is a set of directed edges, representing the services, which can achieve a transition from the source parameter to the destination parameter they connect. The vertex labeling function \( \mu_P \) is responsible for labeling each element in \( V_P \) with relevant information such as the parameter type, acceptable size, etc. The edge labeling function \( \xi_P \) takes care of attributing the edges with service information such as the name of the service, the location and device on which the service is available, the cost of utilizing the service, the quality parameters supported by the service element and the values of those quality parameters. Figure 8a, shows some example service graphs, and the resulting parameter graph is shown in Figure 8b. 

Based on the Parameter graph, we make the following definitions.

1. **Service Zone**: The service zone of a device \( A \) includes all the services available through \( A \) and all its children. The parameter graph \( G_P^A \) contains all the services available in the service zone of \( A \). Therefore, \( G_P^A = \bigcup_{G_S \subseteq \mathcal{S}_A} \{\text{children}(A)\} + G_S^P \).

2. **Search Zone**: The search zone for a particular device \( A \) is essentially the service zone, where \( A \)’s services are aggregated. If \( A \) is a Level-2 or a Level-3 device, the search zone and the service zone are the same. If \( A \) is a Level-1 or a Level-0 device, the search zone of \( A \) is the service zone of \( \text{parent}(A) \). When \( A \) has a service that has to be composed, the process of composition starts from its search zone, expanding to higher zones, if needed.

From the aggregated parameter graph \( G_P \), we can make an interesting observation. All the nodes within \( G_P \), with only incoming edges represent parameters that are usually presentation parameters, such as display, or output. Nodes in \( G_P \), with only outgoing edges correspond to parameters which are usually user inputs, such as audio in, video capture, etc. These parameters can be termed as interactive parameters. Therefore, \( \forall v_P \in G_P \), if \( \text{in\textunderscore degree}(v_P) = 0 \), or \( \text{out\textunderscore degree}(v_P) = 0 \), then the edges corresponding to such nodes represent services which interact directly with the users. We call such services as interactive services. All other services, which are used in between the interactive services, are termed as processing services. By maintaining a list of such interactive services, we can improve the ability of our scheme in handling mobility and other dynamic situations.

### 4.2 Updating Aggregations at \( G_P \)

Due to the dynamism associated with a pervasive computing environment, the approach to maintaining changes within the environment plays an important role in the efficiency of the system. The changes within the environment can be due to a resource such as a handheld camera moving to different locations, or due to the changes in availability of resources such as battery power on a handheld computer.

A parent detects unavailability of a child node, when the periodic LATCH.HELO messages are missing. Once the missing device \( A \) is identified, all the services that were being offered by \( A \) need to be removed from aggregated graph \( G_P \), and the changes need to be reflected at each level of the hierarchy. The missing service \( S_i \) from a parent’s service zone is removed in the local service zone, by removing the edge corresponding the service \( S_i \), the edge from \( p_{n_i} \) to \( p_{i} \), and the nodes \( p_{n_i} \) and \( p_{m_i} \) from \( G_P \), if there are no other edges into and out of those nodes. This change can be termed as \( \Delta G_P \). Similarly, the missing service is reported higher up in the hierarchy with the parent sending LATCH.REM(\( \Delta G_P \)) message to the parent.

When a new device is added to the service zone, either as a result of resource mobility or because of a new device being powered up, the new services are made available to the immediate parent of the device. The inclusion of a new service involves a process similar to that of the aggregation procedure. To keep the information current at all levels of the hierarchy, the addition needs to be reflected into the service zone of the parent, at the higher level. This change in service \( \Delta G_P \) is updated to the parent with a LATCH.ATTRIB(\( \Delta G_P \)) message. Therefore, when the state of a device is altered within the service zone, the changes at each layer of the hierarchy is \( G_P' = G_P \pm \Delta G_P \).

### 4.3 Request resolution

User tasks and application requirements which need services to be composed are modeled as request graphs (\( G_R \)). Request graph \( G_R \) is essentially a plan to accomplish the task at hand. The nodes of the graph, elements of set \( V_r \), are individual services needed to accomplish the task, and the edges of \( G_R \) represent the flow of data among the involved services. With such a request, the desired output from the process of composition is to come up with a valid match for all the elements in \( V_r \). If a single service \( S_j \) matching \( V_i \in V_r \) is not available, then a composite service \( S_j = \{S_i\}_{i=1}^n \) needs to be assembled to meet the needs. In our scheme, we generate such a composition based on the aggregated parameter graph \( G_P \). The aggregated parameter graph \( G_P \) has different parameters as its node elements. The directed edges indicate services that can achieve a transformation from the parameter in the source node to that in the destination node. Therefore, by generating the shortest path from the node matching the input parameter \( p_{in} \) of the requested service in \( v_i \in V_r \) to the output parameter \( p_{out} \) of \( v_i \in V_r \), we have a matching service composed to meet the needs specified by \( v_i \). Essentially, \( v_i = (p_{in} \sim p_{out}) \). For the request graph in Figure 8c, the request resolution generates a composition shown in Figure 8d.
Through the hierarchical service overlay, we have the inheritance between the requesting entity and the services used in its capability to support the locality of service property. Data to users need to be within close proximities to the user. Multimedia in particular, the devices used to render multimedia composition, its ability to handle user and resource mobility, and its support for semantics. Specifically, we present discussions on ScSCo’s capabilities to ensure (i) locality of service, (ii) quality of composition, its ability to handle (iii) user and resource mobility, and its (iv) support for semantics.

5.1 Locality of service

Locality of service refers to the requirement that the distance between the requesting entity and the services used in composition be as small as possible. With respect to multimedia in particular, the devices used to render multimedia data to users need to be within close proximities to the user. Through the hierarchical service overlay, we have the inherent capability to support the locality of service property.

Recall that the hierarchy is created based on the communication range of a service. Consider a hierarchy similar to the one shown in Figure 8. If a request for composition arises at a device A in the hierarchy. As shown in the figure, an attempt for composing all the required services is first done at the device B, the parent of A, over the service zone of B. Since the service zone of B includes all the services around B, any successful composition within B’s service zone is guaranteed to be the closest possible composition for the request. The locality of service is an important aspect for multimedia applications that require interaction. The interactive services need to be present as close to the user as possible. These interactive services can be identified, as mentioned previously, by locating nodes (in \(G_r\)), with either in\(_{\text{degree}} = 0\), or the out\(_{\text{degree}} = 0\). If there exists an interactive service within the vicinity of the user that satisfies the requirements specified in the request, it can be guaranteed that such a service will be used in the resulting composition.

5.2 Quality of composition

Typically, multimedia data are associated with some quality requirements that need to be considered during the composition process. In [11], an intuitive mechanism for attaching quality parameters to resources has been presented. We utilize this scheme to embed quality related information to services. Essentially, the quality parameters associated with each service is embedded as attributes of the service. The node attribute function \(\mu_s\) of each service graph is responsible for attaching and maintaining the quality attributes \(q^s = \{Q_{in}, Q_{out}\}\) for each service. The quality requirements of a request are also specified as node attributes within the request graph \(G_R = \{V_r, E_r, \mu_r, \xi_r\}\).

Therefore, \(\forall v_i \in V_r, \exists \{\xi_{in}, \xi_{out}\}\).

The result of a composition for \(v_i \in V_r\) is \(S_C = \{s_{in}\}_{i=0}^k\), which are all the services that form the path \(p_{in} \sim p_{out}\) in \(G_r\).

Figure 8: Process of Service Composition.

Figure 9: Expansion of search zone to maintain locality of service.
While computing the path $S_C$, we need to make sure that, $q^s(S_C) \geq q^v(w_i)$. And, $q^s(S_C) = \sum_{i=1}^{n} q_i$.

The composed service can be ensured to meet the quality requirements of the request by using the attributes on the edges of $G_P$, to compute the shortest path $(p_{in} \sim p_{out})$ in $G_P$.

5.3 User and resource mobility

One of the major challenges in personalizing multimedia solutions is to address the issues of mobility. In any pervasive computing environment, once the initial composition is identified and a multimedia session is established, the mobility of the user can change the composed solution. The user might walk out of a room where the video is being displayed, or can move to a more capable terminal. Also, there is a possibility that one of the services currently being used in the composition becomes unavailable. For example, a handheld device being used to present the video stream might no longer be available due to power limitations.

In such situations, the challenge is to reconfigure the session under progress as quickly as possible, by considering the current resource availability around the user. With the hierarchical service overlay, it is possible to ensure that the request can be recomputed with minimal interruption of the session under progress.

5.3.1 User Mobility

The effect of user mobility while a multimedia session is in progress can lead to a complete recomposition of the service. But, typically, it is sufficient to identify a new service that can present the multimedia data to the user or capture the data from the user. Since SeSCo maintains a list of services that directly interact with the users, the time taken to reconfigure the session in progress is reduced. When a session is in progress, there are a set of services $\{s_i\}_{i=1}^{n}$, that are initially a part of the composition. When the user moves to a new location, the first task is to identify a suitable service $s_k$ that can present/capture the multimedia data to/from the user.

Based on the service selected to interact with the user, the composition needs to be changed accordingly. Through simulations, we have observed that, in a majority of the cases, it is sufficient to only recompose a part of the original composition.

Since the maximum depth of the hierarchy is 3, it is sufficient to expand the search zone twice to reach the Level-3 device, and if required, another lookup can be made among the Level-3 devices.

5.3.2 Resource mobility

When a resource being used in a composition is mobile, or is no longer available due to resource restrictions, we need to identify an alternative service to fulfill the part that was being played by the missing service. Therefore, when a reconfiguration is necessary due to resource unavailability. Recall that the result of a successful composition is a set of services $\{s_i\}_{i=1}^{n}$, where a subset of the composed services are used in matching the needs of each node in the request graph $G_R$. So, $\forall v_i \in V_R$, $\exists \{s_i\}_{i=1}^{n}$. If a service $s_j \in \{s_i\}_{i=1}^{n}$ present on the device which moved out of a particular service zone, was a part of an ongoing composed session, $(a$ node in the request $G_R)$, only that part of the request needs to be recomposed. Based on the hierarchy, a new composition can be generated to meet the requirements of the affected node $v_k \in V_R$ in at most two search zone expansions and another lookup on $G_P$ among the level-3 devices.

5.4 Room for semantics

The major advantage of semantics is the ability to incorporate reason based service selection during composition. The reasoning ability about the selected services enable selection of a wider variety of services, and also will improve the user experience. Through semantic reasoning, ambiguities about the attributes of a service such as its name, location, etc., can be resolved. Also, by attaching semantics to the I/O parameters of the services, differences among different naming conventions or naming standards can be effectively resolved. Although a semantic match enables the selection of a particular service, at the time of operation, it is important to make sure that the selected service, can be seamlessly utilized in the composition. In other words, although the semantic match can improve the selection of a particular service, it is important to ensure syntactic match among the I/O parameters also exist. Therefore, syntactic match can be considered as a foundation, on top of which, semantics can improve the selection process.

Since the service model utilized within our scheme is a directed, attributed graph, semantic information about the service and the I/O parameters of the service can be embedded as attributes of nodes and edges respectively. During the process of aggregation, the creation of aggregated parameter graph $G_P$ can be modified based on the semantic match among the different parameters, thus improving the probability of a service being used in a particular composition.

Therefore, by using the syntax based matching scheme, supported by semantic matching techniques, the overall quality of composition can be improved. By providing a foundation through syntax matching, and creating room for semantic based reasoning, we believe that our scheme can be effectively applied in personalization of multimedia applications.

6. RESULTS

In this paper, we have presented a novel scheme to compose services to meet user needs in delivering multimedia data to users. Since the technique of composition is unique in its ability to dynamically weave the required services, we first compare the composition success ratio of our scheme with a simple discover + match approach. We have used the JIST [1] simulation platform to carry out our experiments.

6.1 Composition success Rate

The success ratio $s$ is measured as the ratio of the number of successful compositions $c$ to the number of composition requests $r$, $s = c/r$. To illustrate the power of our approach, we have considered a set of services that perform transformation between two alphabets. There are essentially 625 different alphabet transformations. For example $\{a \rightarrow b, a \rightarrow c, \ldots, a \rightarrow z, b \rightarrow a, b \rightarrow c, \ldots, b \rightarrow z, \ldots, z \rightarrow a, z \rightarrow b, z \rightarrow y\}$. The alphabet transformation services are considered only for the purpose of simulations, to illustrate the power of SeSCo. In reality, the services are going to be more complex in nature, and will perform more meaningful
operations. At different time intervals, we vary the number of available services, chosen at random and measure the success ratio of our scheme as compared to a simple discover + match scheme. At each simulation run, 25 random requests were taken for composition. The reported values are an average over 10 different runs of the experiment. The results of the experiment are shown in Figure 10.

![Figure 10: Composition success rate comparison](image)

### 6.2 Mobility induced recomposition time

To measure the recomposition time to support user mobility, we have considered a simulation setup with 9 Level-3 devices connected in the form of a mesh, forming the nodes in the infrastructure. Each of the Level-3 nodes is assigned 6 nodes as its children. The children assigned to each of the nodes are chosen randomly as a mix of Level-2, Level-1 and Level-0 devices. All the nodes are assumed to be equipped with wireless interfaces. The services are distributed so that, for the request set considered, a successful composition can be guaranteed within the simulation environment. The user is associated with a Level-1 device for identification and for making the request. The user is initially a child of node $A$, and moves within the mesh, following a Manhattan grid mobility pattern, stopping at each of the nodes for a predefined time interval. At simulation time $t = 0$, the user makes a request to the immediate parent, where the composition process begins, as explained in Section 3.2. To measure the recomposition efficiency, we have compared the hierarchical service overlay approach with a simple broadcast based approach. All the nodes in the broadcast approach also are equipped with wireless interfaces. In the broadcast approach, a client broadcasts the request for composition to all its neighbors, with a predefined hops to live count. The request is re-broadcast within the network, until the hops to live becomes zero. The composition scheme for broadcast approach is a simple match, with a one-to-one correspondence required among the requested services and those used within the composition.

#### 6.2.1 User mobility

The advantage of the hierarchical service overlay can be clearly seen from the results for recomposing a service. When a user is mobile, the first task is to identify an alternate service that can present the multimedia data to the user. The recomputation is done in a hierarchical manner and the time taken to compose a new matching presentation service within the new service zone. Based on the nature of the identified presentation service, the rest of the composition may or may not change. The effect of a mobile user on recomposition time, with different service densities in the environment is shown in Figure 11.

![Figure 11: Service Recomposition time for user mobility](image)

### 6.2.2 Resource mobility

A service used in a composition may become unavailable due to either the mobility of the node or due to the changing resource levels. In such situations, the node within the request graph, which was composed by using the missing service needs to be recomposed. Therefore, only a part of the composed service needs to be recomposed as a result of resource mobility. The chart in Figure 12 shows the recomputation time against service density. Since SeSCo uses the hierarchical approach to recompose only a part of the request, the saving is significant in terms of the time it takes to compute the composition. In contrast, in a broadcast based scheme, the complexity of locating the required services increases as the number of available services increase.

![Figure 12: Service recomposition time for resource mobility](image)
7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented SeSCo, a dynamic service composition scheme for multimedia applications within pervasive computing environments. The composition approach itself is capable of dynamically weaving the required complex services from the available, basic services. The scheme employs the middleware based hierarchical organization, automatically generated during operation to effectively compose services within the environment. The hierarchical service overlay ensures that the resource poor devices collaborate with a device with higher availability for service related operations such as discovery and composition, through the Latch process. We have shown that dynamisms inherent to pervasive computing arena, such as user and resource mobility, resource restrictions such as limited battery power, etc., can be handled effectively with the hierarchical service overlay.

7.1 Future work

- SeSCo, currently is efficient in performing linear compositions, by attaching the output of a particular service to the input of its successor. We are working on extending this work to accommodate more complex service patterns such as split and merge, loops, etc.
- By employing a fundamental data structure, a directed attributed graph for service representation, we have laid the foundation for embedding semantic information, to empower the composition process.
- SeSCo works on the premise that the request graphs are available for composition. We are working on generating the request graphs, by considering user and application profiles.
- We are currently working on developing a set of service related metrics to enable quantitative analysis of service oriented environments, based on our current scheme.

8. ACKNOWLEDGMENTS

The work presented in this paper was completed under funding from the National Science Foundation Grant NSF-STI 0129682.

9. REFERENCES


A Distributed Scheme for Autonomous Service Composition

Stephen Herborn  
University of New South Wales  
stephen.herborn@nicta.com.au

Yoann Lopez  
National ICT Australia  
yoann.lopez@nicta.com.au

Aruna Seneviratne  
National ICT Australia  
aruna@nicta.com.au

ABSTRACT
Some multimedia content may be divisible into independently routable components, e.g. audio and video flows. As a result, media content adaptation services may be linked in serial, parallel and hybrid configurations to form a directed, acyclic graph of composed services. We specify a distributed service path selection scheme for the construction of composed directed service graphs, which integrates a peer-to-peer routing algorithm, a service discovery mechanism, and abstract scheme for content description. Our approach enables the autonomous selection of converging and non-converging service graphs, which enable media content to be separated into sub-components and delivered to separate devices, applications or network interfaces. Our content, client and service description scheme focuses on addressing mobility, multi-device, and multi-homing requirements. We include results of simulation designed to study the performance of several service discovery options, and present initial conclusions based on our findings.

Categories and Subject Descriptors
C.2.0 [Computer-Communication Networks]: Distributed Systems – Distributed Applications

General Terms

Keywords
Service Composition, Context Awareness, Media Routing

1. INTRODUCTION
Much recent networking research has been based on the assumption that a large number of heterogeneous and likely mobile network-enabled devices may be used to access both static and streaming media content over the Internet. Such devices may be connected to the network via one or more instances of an ever-growing range of last-hop connection technologies (e.g. GPRS, Wi-Fi, ADSL), and may host any number of media display applications (e.g. realplayer [27], mediaplayer [26]). Users are also a source of heterogeneity, for example in terms of languages, monetary budget, trust-levels, and other preferences.

Given the broad heterogeneity of users, applications, devices and their network interfaces, certain items of multimedia content may be required to be adapted, filtered, or transformed in some way before they can be delivered according to cost and/or QoS constraints, or properly displayed to the user. Adaptation may also be motivated by the service provider [17], network provider [12], or by user preferences. In most cases it would be unreasonable to place the burden of adaptation on the client device, as mobile devices are limited by performance constraints such as battery power, processing capability, and available media codecs, as well as by physical factors such as interface bandwidth. It is also unrealistic to expect that service providers can or should be responsible for performing all required adaptation operations. Thus, there is a need for media processing and adaptation services (which we term MediaPorts, or MP) somewhere in the network, between the sink (MediaClient, or MC) and the source (MediaServer, or MS), that are able to transform the media stream from the MS into a form that is acceptable for the MC.

![Figure 1: Variations of composed media service paths](image1.png)

It remains to be seen whether such services will actually become widely deployed, whether they will exist only at the network edges (as in Figure 1) or also in the network core, and more importantly whether they will be interoperable, in such a way that they can be ‘chained’ together to perform multiple media processing operations over the end-to-end path, forming a directed media service path (as in Figure 1a). This may be useful in circumstances where a single adaptation service is not able to...
perform all of the required media processing operations. However, such an approach introduces a plethora of research issues, as described by Nahrstedt and Balke in [14] including service path selection techniques, service discovery, and the design of common ontology for media description. Additional important issues include service path reconfiguration due to mobility events, QoS assurance, and management of service dependencies. In [14] and [16] the possibility of so called ‘hybrid’ or ‘parallel’ service paths is also explored, in which media streams may be split into sub-components and routed through an independent set of MediaPorts before converging and being delivered to the client. The potential for hybrid service paths, which are essentially directed acyclic graphs, introduces some novel routing problems. In this paper we provide details of an integrated distributed approach to the construction of hybrid composite service paths on a media overlay network. Our proposal consists of a distributed, stateful path search algorithm, and several options for media adaptation service discovery including a scope-limited path directed search technique. We develop an abstract ontology for media description for the purpose of describing a set of logical functions that we use as part of our service path selection scheme. Our description scheme is novel in the sense that it is developed specifically with mobility and multi-homing in mind. Additionally, we address the potential for non-converging service graphs, as in the example scenario depicted in Figure 1c. A basic level of QoS optimisation in our scheme is performed by favouring paths with the low end-to-end latency if there are a number of potential paths available. We do not emphasise low level QoS assurance or monitoring within the scope of this paper, though we discuss our plans for a more comprehensive treatment of QoS assurance in the context of future work particularly in regard to providing synchronisation over non-converging service graphs.

The rest of the paper is structured as follows: In the remainder of this section we discuss the motivation for media overlay networks and discuss the service graph composition problem. In section 2 we examine related work in the areas of service composition, service discovery, and media endpoint description. Section 3 contains details of an abstract scheme for media description with an emphasis on accommodating mobility and multi-homing. In section 3 we also specify a number of comparison functions on media descriptions that are used as building blocks in later sections. Then, in section 4, we discuss several approaches to media service discovery, and in section 5 we provide details of a service graph selection algorithm that is able to compose serial service paths, as well as parallel and hybrid service graphs. In section 6 we provide details of experimental analysis and we conclude and discuss future work in section 7.

1.1 Media Overlay Networks

Some level of infrastructural support is needed for service composition, in order to discover services and establish service paths. In [14], a distinction is made between two classes of infrastructural support for service composition: unstructured peer-to-peer networks, and managed service overlay networks. In order for services to be interoperable in either case, they must share a common means of interacting with their peers, above and beyond the functionality provided by the underlying network. As such, composable media services implicitly form a Media Overlay Network (MONet), the administration of which may be centralised, totally distributed, or somewhere in between. In this paper, we focus primarily on a distributed MONet, based on a peer-to-peer network of physical Overlay Nodes (ONodes), due to the ability of peer-to-peer to handle dynamic data in a more scalable fashion [25]. The ability to handle dynamic data is important since changes in user context (e.g. swapping to a new device/location) may result in the need for timely reconfiguration of a service path. The mechanics of the peer-to-peer communication employed are out of the scope of this paper, but may involve a structured or unstructured system. With this in mind we describe how composite service paths may be composed autonomously i.e. with minimal explicit configuration from external entities. We explore the effects of both centralised and distributed service discovery techniques.

A MONet can be viewed from two different perspectives, logical and physical, since MediaPorts are logical entities and as such a single physical ONode may play host to more than one MediaPort. Accordingly, it is possible for several services on a composed service path to be provided by MediaPorts that are hosted by a common ONode. We do not account for this phenomenon explicitly, as in [3], but do observe that it is the task of the distributed service discovery mechanism to discover relevant services on behalf of MediaPorts. Thus, if a discovered service is provided by a MediaPort residing on the same ONode from which the service discovery function was called then it is highly likely that this particular service will be looked on favourably due to its negligible distance from the MediaPort that initiated the query.

1.2 Service Graph Composition

An interesting development in service composition stems from the fact that since media streams may consist of several separable and independently routable components, which we term media flows, e.g. audio and video, and since some MPs may be able to split or join certain media streams, it is possible to construct an end-to-end service path that is composed of two or more converging sub-paths (see Figure 1b). The benefits provided by splitting and joining media include the potential for selection of service graphs that are more efficient than any available serial service path, as well as the potential to utilise media services that only accept media data of a given sub-flow type. For example, audio/video content may need to be split before the audio stream can be passed though a translation service then rejoined and synchronised before final delivery. We adopt existing terminology from [14] and call such paths ‘hybrid service graphs’, i.e. a hybrid of purely serial and purely parallel directed service graphs. It is foreseeable that in some cases it may actually be desirable if service graphs do not completely converge, for example in order to deliver the audio component of a media stream to a network-enabled headset and the video component to a wall mounted LCD (see Figure 1c), to deliver different media flows to different network interfaces. Such composed service configurations, as well as even more complex configurations, as in the scenario presented in Figure 1d, are discoverable by the scheme detailed in this paper.

Possible examples of independently routable media content components are audio, video, images and text. Each type of component is associated to different QoS requirements which can be subject to objective (e.g. ordering in a text flow) and subjective (e.g. jitter in an audio flow) measures of QoS. Routing independent media flow components over different service paths makes it possible to perform differentiated routing and application.
adaptation [29]. Some media components that have been routed over different service paths may need to be synchronized before final delivery, a functionality which may be built into MediaPorts that are able to multiplex/join media content.

The concepts presented above entails some novel media routing problems regarding the construction of ‘valuable’ service graphs that may consist of multiple sub-paths linked in serial and/or parallel. For a given MS/MC pair and a media content item, it is difficult to nominate the canonical set of services that are required in an a priori fashion since the MediaPorts that are available to perform a given required service may introduce dependencies that also need to be accounted for. Thus, the ability to avoid explicit management of MediaPort dependencies serves as another motivation for a peer-to-peer approach to service composition whereby each MediaPort on a given service path is directly responsible for choosing its successors on the path.

2. RELATED WORK

Our analysis of related work may be broadly classified into those that deal with media description schemes, those that deal with service discovery, and those that deal with the construction of composed service paths, though there is obviously some overlap due to the interdependent nature of these topics.

2.1 Service discovery and composition

Amir et al. introduce the concept of composable services in [2]. In [13] Nahrstedt and Balke make the case for a more detailed study of techniques for service composition, and in [14] they subsequently develop a taxonomy for multimedia service composition. A number of different service composition scenarios are analyzed, and the concept of hybrid service paths i.e. service paths that are composed of both serial and parallel sub-paths is described. UDDI is mentioned as a service location mechanism in the context of web services. Our proposal, in some sense, extends on concepts expressed in this taxonomy, although it does not address non-converging service paths.

The Ninja Paths project [20] allowed services to be automatically discovered and composed into a path, which provided a stream-like interface to route data between composed services that may convert or process the data. In Ninja Paths, candidate services are discovered using the Ninja Service Discovery Service, consisting of hierarchically organized indexing nodes. MeGaDiP [19] is another approach to service discovery that is explicitly targeted towards media stream processing services, rather than the so-called ‘sink-like’ services in Ninja Paths. The path directed search technique adopted by MeGaDiP is comparable to one of the media service discovery models used in this paper, although it does not account for services with splitting or joining capability. Additionally, our model is a purely peer-to-peer one in which the ONodes involved in service discovery may also be responsible for providing services, rather than using a separate media service indexing system such as MeGaDiP.

In [3], the problem of locating services and routing service paths in an overlay media service proxy network was addressed with the design of a service discovery and path computation system. The emphasis of the system was on constructing serial service paths while satisfying bandwidth and processing capacity constraints of media service nodes. The means by which desired intermediate services are determined is not mentioned in great detail in this proposal, nor are the issues regarding parallel and hybrid service paths such as those illustrated in Figures 1c-d.

Concepts relating to service composition with an emphasis on QoS assurances are developed by Gu et al. in [5], who propose an architecture to support QoS provisioning for composed services, based on the SLA contracts of individual service components. In [16] Gu goes on to propose SpiderNet, a fully decentralized service composition framework which provides statistical multi-constrained QoS assurances and load balancing for service composition. Service composition in SpiderNet revolves around a bounded composition protocol. The intent and high-level approach of SpiderNet is somewhat similar to that of this paper, however it does not specifically consider issues relating to non-converging service graphs paths or multiple end-devices.

Performance issues of ‘media gateways’ (analogous to our MediaPorts) are analysed by Ooi et al. in [23] and [8], where it is found that passing a media stream through a media gateway can introduce up to 30 ms of latency due to the encoding and re-encoding process. Ooi also ventures in the foray of media service location with AGLP, an Adaptive Gateway Location Protocol [24], which uses propagation time as a parameter to decide if a gateway is suitable to service a client. Reference is made to a previous study [28], which found that there is little correlation between geographical locations, topology or number of hops in determining network level proximity. In [8], Ooi experiments with composable services by distributing a given media processing operation over a number of media gateway nodes, though the number of gateways in serial is limited to two.

2.2 Service and multimedia description

In [1], Gu et al. introduces an XML-based hierarchical QoS markup language, HQML, to enhance distributed multimedia applications on the web with QoS capability. As in our proposal, HQML classifies value tags into several levels, i.e. user level, application level, and system resource level, though the intention of this classification differs somewhat from our scheme.

Exposito et al. also specify an XML QoS specification language, with the intent of providing a global QoS description language in order to map application QoS requirements to the required transport, network and system resources.

SDP next-generation, or SDPng [4] defines a language for the description of media sessions with respect to configuration parameters and capabilities of end systems. SDPng, as the name suggests, is attempt to extend the Session Description Protocol described in RFC2327 with an extensible XML-based notation scheme and provide integrated support for specification of codec parameters and media gateways (as defined in RFC3015). Other standards based approaches to media description are MPEG-7, a standard for describing multimedia content data, and MPEG-21, a related ongoing framework specification aiming at defining a normative open framework for multimedia delivery and consumption. It is yet to be seen if either standards will become widely used enough to reach the same level of ubiquity as previous MPEG standards.

The SIP user agents profile delivery [10] internet draft, though not a description framework in itself, makes a conscious separation between device, user, application and local network profiles. This separation is motivated by the desire to support different kinds of
physical and logical mobility (i.e. of devices, users, application and network interfaces) and classifies information into profiles according to the above listed groups. Such an approach to information organization provides the potential for simpler re-configuration of service paths when the properties of endpoints change due to mobility. The description scheme we adopt in this paper is also targeted at organising information in a way that enables intuitive handling of mobility and multi-homing.

In [9] Rafaelsen and Eliassen construct a description language for media gateways, called Gateway Definition Language. The focus of GDL is to simplify the process of determining whether or not a certain media gateway is able to provide a meaningful service on a given media stream. WSDL [21], the description language for web service may also be seen as a candidate for the description of services offered by media gateways.

3. SERVICE DESCRIPTION SCHEME

Service interoperability is a crucial factor in the viability of the service composition model. More specifically if service composition is to be performed autonomously, as we propose, then the common scheme by which media content, services, providers, and consumers are described must facilitate intuitive comparison. In this section we develop an abstract description scheme for media routing with the goal of ensuring that it is easy to discover and compare the semantic difference between descriptions. Easy comparison is needed in order to determine if an MP or MC is able to receive the media content in its current state, and to determine whether or not a given MP can perform a meaningful adaptation on the stream. Here we introduce a new term media endpoint, in order to describe what we consider to be the ultimate endpoints of a media stream, i.e. the content or the MC. Note that we do not consider the MS to be a media endpoint, since the end-user rarely has an interest in the actual physical host on which the content resides, nor should the content host itself interact with the content it provides, other than to serve as a medium between the content and the user.

3.1 Description scheme

We adopt an abstract description scheme to express a clear, hierarchical relationship between the elements that make up a media endpoint. Our description scheme classifies media endpoint elements into four object classes, listed below in descending order of hierarchy:

User - user level, e.g. preferred language etc.

App - application level, e.g. available codecs

Dev - device level, e.g. supported video resolution

Ifc - network interface level, e.g. supported bitrates

Our motivation for this classification comes from the perspective of mobility, multi-homing and context management. Users, applications, devices and network interfaces constitute a comprehensive grouping of objects that may be mobile (logically or physically) and thus may change characteristics independently from one another. Once the configuration of media adaptation services is largely driven by available context information, it is important to be able to draw a clear demarcation between the specific sub-components of a media endpoint to which an article of context information relates. In order to reflect multi-homing, a given media endpoint description may consist of many instances of each class, for example a multi-homed device would correspond to a description where several Ifc objects belonging to a single Dev object (as exemplified by Figure 2).

To carry one media flow from one endpoint to another, only one instance of each object is used, for example only one combination of [application, device, network interface] out of those available. A media description with only one object of each class is termed irresolvable. Thus, media descriptions need to be resolved down to irresolvable forms in the service path discovery phase of media delivery. Figure 6 depicts a resolved description.

The classification of elements into the above four classes is largely subjective and some elements may exist in more than one class, but in general characteristics should be classified according to the lowest class in which they may feature as a constraint. For example, ‘supported bitrate’ may exist at both the application and network interface classes, however it is obvious that the possible bitrate of data delivered to applications is ultimately bounded by the capability of the network interfaces, and thus the authoritative ‘bitrate’ element would belong at the network interface class.

To develop the simple XML representation used in section 3.1 and a set of comparison functions in section 3.4, we first specify a formal notation for media endpoints. Unresolved media client endpoint descriptions can be expressed as a set of elements \( \Gamma \), which can be defined according to the following set of rules:

\[
\Gamma = \{ \xi | \xi \in I \}, \quad \forall \xi \exists \xi = \{ I, i = 1, ..., n \}, \forall \xi \exists \xi = \{ | I, i = 1, ..., n \}, \forall \xi \exists \xi = \{ | I, i = 1, ..., n \}, \forall \xi \exists \xi = \{ | I, i = 1, ..., n \}
\]

For all \( l \in \Gamma \), where \( M = \{1 ... m\} \) and \( m \) is the number of classes in the description scheme, in our case four. The notation \( x = y \) indicates that \( x \) is a child of \( y \), \( x = y \) that \( x \) is a parent of \( y \). Each class \( l \) contains zero or more elements \( \xi \), i.e. \( l = \{ \xi, ..., \xi \} \) for \( n \geq 0 \), where \( n \) is the number of elements in to \( l \) and \( \xi = \{ N, t, v \} \) where \( N \) is a set of namespace tags indicating the element’s modal scope, \( t \) is some nametag, and \( v \) is a value. An irresolvable media endpoint is any subset of a resolvable media endpoint \( \forall \xi \exists \xi = \{ | I, i = 1 \} \). Both media content endpoints and MediaPort inputs/outputs can also be expressed in the same way. Table 1 provides a summary of this notation.

Element nametags serve to identify the role of the element, and namespace tags are used to denote the sub-flow to which the element relates. For example, the ‘language’ element may only be relevant to audio, so would belong to an ‘audio’ namespace. Namespaces are important since they may facilitate intuitive conversion of media flows to new modalities. An example would be converting an audio flow to a text stream in the case that only a text capable device is available to the user.

As identified in [4] and [9], description element values may be optional or mandatory, and may either be single (e.g. language="french"), ranges (e.g. bitrate="(100-200)"), or enumerated (e.g. codec="[divx, mpg4]"), which contain one or more single values or ranges. For the purpose of comparison, if a value is denoted as mandatory, then single values should match outright, whereas range and enumerated values only need to share some region of intersection.

We intentionally do not attempt to further develop this into a complete media description framework, as many previous
solutions exist for the same purpose, e.g. [1],[4],[7],[21]. Rather, we observe that just about any media endpoint description can be viewed in terms of the four sub-groups of characteristics outlined above. In any case, the scheme can be easily extended to include more classes if needed. In our simulation we use an SDPng [4] style description as a base, and apply translation process to convert the description into a format that conforms to our abstract scheme.

Given the means to describe media endpoints, a significant related challenge is how to perform comparison and logical transformations on media endpoint descriptions. Simply said, it should be straightforward to look at a content description, then look at a client description, and from this to tell whether or not the content is in a form that is able to be received by the client. Similarly, it should be possible to determine the effect that the application of a given media service will have on a content description. Finally, it is vital to be able to ascertain from this information whether or not the service is able to perform a function that could be deemed ‘useful’. We define ‘usefulness’ to mean that some desirable end-to-end adaptation operation is performed. More formally, if \( \lambda \) denotes a given irresolvable MediaClient description and \( \gamma \) denotes an irresolvable multimedia content description we define \( \text{Diff}(\lambda, \gamma) = \Psi_{\lambda \gamma} \) to represent the end-to-end mismatches between the client and the content,. Thus, \( \Psi_{\lambda \gamma} \) also represents the set of adaptations required on the path between the content and the client, e.g. no adaptations are needed if \( \Psi_{\lambda \gamma} = \emptyset \). If for a given MediaPort \( P \), \( \Psi_{\lambda \gamma} \) is the set of adaptations required after passing the media through \( P \), then the condition \( |\Psi_{\lambda \gamma}| < |\Psi_{\lambda}| \) must be satisfied for \( P \) to be considered ‘useful’. In Table 1 we introduce the notation that we use throughout the remainder of the document.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A )</td>
<td>An unresolved media client description</td>
</tr>
<tr>
<td>( \Gamma )</td>
<td>An unresolved media content description</td>
</tr>
<tr>
<td>( \lambda )</td>
<td>An irresolvable media client description</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>An irresolvable content description</td>
</tr>
<tr>
<td>( \Psi_{\lambda \gamma} )</td>
<td>List of changes needed for ( x ) to become ( y )</td>
</tr>
<tr>
<td>( P^{ui} )</td>
<td>A MediaPort input/output description</td>
</tr>
<tr>
<td>( \text{Diff}(\Gamma, A) )</td>
<td>Difference function</td>
</tr>
<tr>
<td>( \xi ) ( &lt;N,t,v&gt; )</td>
<td>An atomic media description element</td>
</tr>
<tr>
<td>( N )</td>
<td>A set of namespaces (e.g. ‘audio’, ‘video’)</td>
</tr>
<tr>
<td>( T )</td>
<td>An identifier (e.g. ‘language’, ‘bitrate’)</td>
</tr>
<tr>
<td>( V )</td>
<td>A value (e.g. ‘french’, (10-100), [divx,rm])</td>
</tr>
<tr>
<td>( \Omega )</td>
<td>An end-to-end service graph</td>
</tr>
<tr>
<td>( \omega )</td>
<td>A serial sub-path belonging to ( \Omega )</td>
</tr>
</tbody>
</table>

\( \text{1 We further define this function in section 3.4} \)

### 3.2 Describing Media Endpoints

Below in Figure 2 we provide a sample XML formatted description of a media endpoint, simplified for the purpose of clarity. In can be understood from this figure that the depicted client endpoint consists of a user, ‘bob’, who has access to some media display application ‘mediaplayer’ which is installed on two available host devices, ‘pda’ and ‘laptop’. The ‘pda’ device has one currently connected network interface, whereas the ‘laptop’ device has two. The figure includes examples of all the element value types mentioned in section 3.1. Additionally, some element tags are repeated at different levels in the hierarchy, for example \(<\text{codec}>\) and \(<\text{bitrate}>\). In this way, entities belonging to different levels of the hierarchy are able to express their preferences regarding characteristics that are constrained by lower layers. In the example given in Figure 2, the user has a preferred video codec choice of “DIVX”, however the associated application supports only “MPG4” or “RM” codecs. Similarly, the application can process data at bitrates of 200 kbps up to 20000 kbps, however the maximum bitrate supported by any of the available network interfaces is 10000 kbps. In the case of such conflicts, lower levels of the hierarchy will always take precedence, however if there is a region of intersection that exists between the higher level preferred value and the lower level value then this region of intersection represents a negotiated ‘best-choice’ for this particular characteristic. In the example discussed, the negotiated best-choice would be the intersection of 200-20000 and 0-10000, i.e. 200-10000 kbps.

---

**Figure 2: Sample XML description of client endpoint**

For completeness, a similarly formatted description of another media endpoint, this time a multimedia content item, is presented below in Figure 3. From comparison of this figure and figure Y, it should be apparent that there are mismatches at several levels of the hierarchy between what is mandatory or preferred on the part of the MC, and what is available at the MS. The set of end-to-end adaptation operations can be inferred from this mismatch set shown in Figure 5. For example, an language audio mismatch between ‘french’ and ‘english’ indicates that an intermediary...
3.3 Describing MediaPorts

MediaPorts, the overlay entities that provide intermediate media processing services, can be modelled by a set of descriptions that describe one more input ports, and one or more output ports. An input port represents a media flow that is required by the MediaPort before it can perform any processing operation, and an output port represents one independent media flow that is produced by the MediaPort. Thus, a MediaPort with one input and one output is one that simply performs a service on an incoming media stream, and then outputs a single processed stream. MediaPorts with one input and several output ports, on the other hand, belong to the class of ‘ Joiners ’ or ‘ Multiplexers ’, and MediaPorts with one output port and several input ports are ‘ Joiners ’ or ‘ Multiplexers ’, as in Figure 4. In this paper we do not consider the possibility of MediaPorts with both more than one input port and more than one output port, though we observe that such an entity may be modelled as a composition of several of the MediaPorts as illustrated in the figure below.

From Figure 4 it can be seen that we model MediaPorts as simple functional blocks. A MediaPort has a set of irresolvable endpoint descriptions, I, representing its input ports. And a set of irresolvable descriptions O representing its output ports.

From the comparison output it can be seen that there are several mismatches between the two endpoint descriptions, which implies the need for adaptation, i.e. Diff (‘bob’, ‘starwars’) > 0 . When we proceed to search for service paths to eliminate the end-to-end mismatches, we do so for each possible path from a leaf node to the root node (i.e. for each irresolvable form of each endpoint) until there is a complete service path for all sub-flows of the content, or no suitable service path exists. In this way, non-converging service graphs may be discovered if more than one irresolvable endpoint is used by either the MS or the MC to achieve full delivery of the media content. An example of such a scenario is depicted by Figure 1c.

Figure 6 depicts an XML representation of the set of end-to-end adaptations that are required before ‘bob’ can receive ‘starwars’. This represents one possible combination of media endpoints. It can be seen from the figure that some adaptation is required: This represents one possible combination of media endpoints. It can be seen from the figure that some adaptation is required: From the MS for all outputs of a de-multiplexer, or all inputs of a multiplexer. A node is required to be complete before it can be included on a composed service path. We make use of this definition in the routing algorithm detailed in section 5.

3.4 Comparing Media Endpoints

Media adaptation and transformation services are required in order to eliminate mismatches between the media content and the media client. However in order to determine the adaptation services that are needed on the end-to-end path between the content and the client, there first needs to be some way to compare their respective descriptions for mismatches or other indications that the media cannot be delivered ‘as-is’. In our simulation implementation we use a modified X-Diff [30] algorithm to analyse the similarity between two media endpoint descriptions coupled with a set of logical functions used to infer whether the application of a certain service would be useful. Figure 5 depicts sample output of a comparison between the two media endpoint descriptions illustrated in the above figures.

![Figure 4: MediaPorts](image)

![Figure 5: Sample output of comparison](image)
Finally, the Useful heuristic may also be expressed as a logical function. It can be deduced from Expression 3 that this heuristic will return a positive result if the number of differences between the adapted media description and the goal state description is less than the number of differences between the original media description and the goal state description. In order not to limit MediaPorts from introducing dependencies and thus not exclude MediaPorts that may be able to offer a valuable service, the heuristic only considers discrepancies concerning elements that exist in the original media description. \( |\text{Diff}(\gamma, \gamma^\text{goal})| = 0 \) for the purpose of this expression.

\[
\text{Useful}(\gamma, \gamma', \gamma^\text{goal}) \Rightarrow \left| \text{Diff}(\gamma', \gamma^\text{goal}) \right| > \left| \text{Diff}(\gamma, \gamma^\text{goal}) \right|
\]

Expression 3: The Useful function

Where \( \text{Diff()} \) is defined by the following expression:

\[
\text{Diff}(\gamma, \gamma') = \gamma'' \text{, where } \gamma'' = \{ \xi \{N, t, v\} | \xi \{N, t, v\} \in \gamma, \xi \{N, t, v\} \in \gamma', v \land v' = \emptyset \}
\]

Expression 4: The Diff function

Using the above functions, our proposed service path construction algorithm is able to intuitively compare media and media port descriptions on the fly, and determine whether or not a given MP should be included on a candidate service path. We further detail how these functions are applied in the following sections on service discovery and service path routing.

4. SERVICE DISCOVERY

As identified in the introduction, our scheme for autonomous composition of directed service graphs requires some means to discover services that are able to perform a desired media processing action. The means by which services can be discovered depends greatly on the infrastructural model being used, and in our case we opt to examine peer-to-peer service discovery methods that can be easily integrated in the distributed search algorithm detailed in section 5. In order to discover candidate media services, we define the function \( \text{find_services()} \), which is used by each successive hop in our distributed routing algorithm. We experimented with several permutations of the MP discovery function, namely: global directory service, limited scope broadcast, and directed path search. The potential signalling overhead caused by each call to the MP discovery function is influenced by the amount of information that is provided to it, i.e. a call to \( \text{find_services()} \) with an overly brief set of search terms may simply result in a list of all MPs in range of the current hop, whereas if the discovery function is provided with additional information e.g. current state of the media flow, pending adaptation operations etc. then it is clear that though the aggregate cost of sending and forwarding query messages would be higher, it will result in a far lower number of irrelevant results that need to be propagated back to the querying MP. Such an approach distributes the processing burden of media description comparison throughout the overlay and provides more relevant results, since MediaPorts are able to first check whether or not they can make a valuable contribution to the service graph. In our simulation, we
compare three different modes of MP discovery: global directory, limited flooding, and path-directed search

![Figure 7: Scope comparison of service discovery mechanisms](image)

### 4.1 Global directory
In the global directory media discovery approach it is assumed that there exists, somewhere in the network, a globally accessible database of MP descriptions. A structured overlay such as Chord [25] may be used to provide a scalable global service directory, as may systems based on UDDI. Any ONode may submit a query for a specific service, or for a service that is able to perform media processing on content with a given description. Obviously, in the case that the MONet contains huge numbers of MediaPorts, a poorly qualified search query may return an unacceptably large number of results. One solution to this would be to iteratively return results in batches, as is the present practise by most web search engines. Another possible approach would be to filter results that are sufficiently distant from the querying ONode. We adopt the latter approach in our simulation.

### 4.2 Scope-limited flooding
Scope limited flooding is a basic peer-to-peer search technique by which a peer floods a scope limited search query to all of its neighbours on the MONet, which they subsequently propagate further to their neighbours. The search query is embedded with a TTL which indicates how far away from the originating ONode the search request should be propagated. Thus, the search pattern resembles a circle centred at the originating ONode and of a radius determined by the size of the TTL used.

### 4.3 Path directed search
Media routing is heavily influenced by the need to satisfy end-to-end delay constraints, particularly for real-time applications, thus it makes sense that we only consider paths through the overlay network that bring the media stream successively ‘closer’ to the MC with each hop. Similar media service search techniques are explored by Xu et al. in [19], and Asmare et al. in [32]. More formally the conditions that must be met by MPs on a given service path can be expressed as:

**Condition 1:**
\[
\forall \text{MediaPort } MP \in \text{service graph } \Omega, \ MP' \supset MP
\]
\[
D(MP, MS) / D(MP, MC) \geq D(MP', MS) / D(MP', MC) - \epsilon
\]

**Condition 2:**
\[
\exists d = D(MP, O): d \leq \epsilon, \text{ where O is the set of ONodes that lie along the direct network path from the MS to the MC.}
\]

Where \(D(A,B)\) is a distance function, and \(\epsilon\) is a scope constant which may be used to relax the strictness of this heuristic. The distance function may be implemented using ICMP round trip time measurements (assuming clock synchronisation), or by some other means, the details of which are out of the scope of this paper. Thereby, we perform a path-directed search where each ONode only discovers ONodes (and MPs) that are close to underlying network path between the two endpoints.

### 5. SERVICE GRAPH ROUTING
The intent of media routing is to transform media content from its original state into a ‘goal state’ that is acceptable for the client, as well as to physically deliver it to the client. In this section we propose a distributed algorithm that ensures the media content progresses closer to its goal state at each successive intermediate service hop, and to discover and build media service paths such as those shown in Figure 1. We assume that there exists some MP discovery mechanism at each overlay node (or ONode), as discussed in section 4 above. The algorithm, as listed below, is a ‘stateful’ depth first search where nodes are selected according to heuristic (see lines 4, 13, 18 of Fig. 8) and search state information (e.g. Fig 8, line 5). Search state is used to determine if a ‘joiner’ MP is waiting for more inputs before continuing the search.

**Input:** current path \(\omega\), pending end to end adaptations \(A\), current state of media \(\gamma\), goal media state \(\gamma^{goal}\)

**Output:** complete service path \(\Omega\)

```
1. do while Diff(\(\gamma\), \(\gamma^{goal}\))\neq\emptyset
2. \(X := \text{find_services}(\omega, A, \gamma, \gamma^{goal})\)
3. for each candidate \(x \in X\)
4. if \(x\) can join this flow,
5. if \(x\) is complete,
6. define \(\omega' \supset \omega\) plus \(x\)
7. add \(x's candidate sub-paths to \omega'
8. find_path(\omega', A, \gamma, \gamma^{goal})
9. else
10. save \(\omega\) as a candidate sub-path in \(x\)
11. put \(x\) into waiting state
12. backtrack
13. else if \(\text{Useful}(A, \text{Adapt}(\gamma, F^\omega), F^\omega)\)
14. define \(\omega' \supset \omega\) plus \(x\)
15. define \(A' \supset A\) less operations done by \(x\)
16. define \(\gamma' \supset \text{Adapt}(\gamma, F^\omega)\)
17. find_path(\(\omega', A', \gamma', \gamma^{goal})\)
18. else if \(x\) can split this flow
19. for each \(F^\omega\)
20. define \(P \supset P\) plus \(x\)
21. define \(\gamma' \supset \text{Adapt}(\gamma, F^\omega)\)
22. find_path(\(P', A', \gamma', \gamma^{goal})\)
23. return \(P'\) as \(\Omega\)
```

![Figure 8: Service path discovery algorithm](image)
The service discovery function mentioned in section 4 is used on line 2 of the algorithm, and line 13, line 17 show the context in which description comparison functions detailed in section 3.4 are used.

The term candidate sub-path as mentioned on lines 7 and 10 of the path search algorithm refers to a path leading into a non-complete MediaPort. A non-complete MediaPort is a ‘joiner’ that is still waiting for additional input components before it can output a stream e.g. an audio/video stream multiplexer and synchroniser. When an incomplete MediaPort is encountered, the discovered path up to that point is saved (locally in that MediaPort) as a candidate sub-path for potential inclusion in the completed end-to-end path. In the case that there are many potential service graphs that achieve the same result, we select the one with the lowest end-to-end latency i.e. the path that is returned first.

6. EXPERIMENTAL EVALUATION

An initial experimental evaluation was performed in order to compare the effect of different service discovery models on the service graph routing logic. We implemented the media description comparison functions as described in section 3.4 by extending the X-Diff algorithm presented by Wang in [30]. The X-Diff algorithm is designed to detect changes and determine a minimum cost edit path between two XML documents i.e. determine precisely where the two documents differ. This functionality is highly relevant to the service composition problem space, since many media description schemes use XML as a basis.

Our simulation was implemented in Java, using Inet-3.0 [40] to generate a IP-level network topology. It is composed of 3500 nodes in a 10,000 * 10,000 node 2-dimensional overlay space.

In the first experiment, we compare the limited flooding and path-directed service discovery techniques. We do not include the global directory approach in our evaluation since it differs in a primarily qualitative rather than quantitative manner, however we note that the success rate of the global directory approach will necessarily be 100% if a path exists. We run each algorithm on the same topology with variable search scope. Search scope is a relative value that indicates how far the search query may be propagated from its originator in terms of network distance, as discussed above in section 4. For each search, we estimate the success rate of finding a complete service graph given that such a service graph does exist. The success rate in our case is the ratio between the number of times a complete end-to-end service graph is found, to the number of times such a graph is not found. From Figure 9 it can be seen that the path directed search is only more effective after a certain search scope value, corresponding to a search scope of about 3000 in the above simulation case. However once this value is reached its success increases dramatically, quickly overtaking the limited flooding method and eventually attaining a 100% success rate. The limited flooding search, on the other hand, provides a better success rate initially, but grows slowly and is not able to guarantee discovery of a complete path even if one exists (given a reasonable search scope).

Figure 9: Success of different discovery methods

Our second and third experimental evaluations, shown in Figures 10 and 11, were performed to evaluate the search overhead of the limited flooding search and path directed search against the search scope. From Figure 10 it can be seen that the limited flooding method produces a far greater number of service queries overall. The sharp declination observable in both plots coincides with the search scope value for which a path is most likely to be found, with some variation evident due to the differing path lengths for successive experiments. From Figure 11 it can be seen that the flooding approach results in less overhead from query responses for smaller values of search scope, however this value.

---

2 See definition of complete MediaPort in section 3.3.
rapidly increases. The path directed search on the other hand results in a bounded level of overhead due to query responses.

From the results of the initial simulation study discussed above it can be concluded that the path directed search approach provides a higher average success rate, and entails a lower search overhead in the case that the required services are not located close together in terms of network distance. However, if the required services for a given service graph are located in close succession to each other, then the limited flooding technique will perform better.

7. CONCLUSION AND FUTURE WORK

In this paper, we have presented and analysed an integrated distributed approach to the composition of directed composed service graphs. Our approach is focused on enabling a high degree of autonomy in the selection of services, and on accommodating potential device multiplicity and multi-homing. Initial simulation studies of different ad-hoc service discovery models led us to the conclusion that the directed path search approach is a good candidate when distance between successive service graph components is expected to be large, whereas a limited flooding approach performs better where these distances are small. We did not experimentally establish which approach results in better quality or cheaper service configurations, but plan to investigate this issue in the context of future work.

For additional future work we intend to use the open source media streaming platform VideoLAN [6] and Planet-Lab [33] to further explore the mechanism by which non-converging service graphs are selected, and to investigate implementation issues related media service composition. In particular, we aim to address media flow synchronisation issues introduced by composed service paths, especially hybrid and non-converging service graphs such as those illustrated in Figures 1b-d. We aim to incorporate parts of our design into a larger context management and mobility handling architecture [11] with the goal of providing mobility that is seamless not-only in the sense of network connectivity, but also in the sense of end-user perception.

8. ACKNOWLEDGEMENTS

The authors wish to thank the anonymous reviewers for their valuable comments and suggestions, as well as José Rey and all the active participants of Ambient Networks WP5 for providing the inspiration and motivation for this work.

The views and conclusions contained herein are those of the authors and do not necessarily reflect the official policies, expressed or implied, of the Ambient Networks project.

9. REFERENCES


[27] RealNetworks RealPlayer, http://www.real.com


[31] Inet Topology Generator, http://topology.eecs.berkeley.edu/inet/


[33] Planet-Lab, http://www.planet-lab.org
Transparent End-Host-Based Service Composition through Network Virtualization

Stefan Götz  Klaus Wehrle

Junior Research Group
Protocol Engineering & Distributed Systems
University of Tübingen
Auf der Morgenstelle 10c
72076 Tübingen, Germany
{stefan.goetz|klaus.wehrle}@uni-tuebingen.de

ABSTRACT
Mobile devices have become a popular medium for delivering multimedia services to end users. A large variety of solutions have been proposed to flexibly compose such services and to provide quality-of-service guarantees for the resulting contents. However, low-level mobility artifacts resulting from network transitions (disconnected operation, reconfiguration, etc.) still prevent a seamless user experience of these technologies. This paper presents an architecture for supporting legacy applications with such solutions in mobile scenarios. Through network virtualization, it hides mobility artifacts and ensures connectivity at the network and transport level. Its adoption for multimedia applications poses unique challenges and advantages, which are discussed herein.

Categories and Subject Descriptors
C.2.2 [Computer Systems Organization]: Computer-Communication Networks—Network Protocols

General Terms
Design, Experimentation

Keywords
Legacy Support, Mobility, Multimedia, Service Composition

1. INTRODUCTION
Mobile devices form an increasingly attractive platform for multimedia applications. Corporate environments in particular obviate such mobile applications. Users ubiquitously access multimedia data through a variety of devices in their offices, meeting rooms, cars, at their customers' site, or at home. In these settings, video, audio, and textual data need to be continuously adapted. For example, users may expect a text-to-speech conversion of their e-mails while driving or tools for rich-media collaboration in real time over wireless links. Many of the challenges of providing multimedia services to users in such scenarios have been addressed by recent research in the areas of adaptability to changing networking environments, quality of service, and service composition [10, 6, 4].

However, to take advantage of these services, changes to the applications themselves and the underlying operating systems become necessary. Consequently, it is difficult and expensive to evaluate novel protocols, frameworks, and middleware, and to deploy them on end systems, resulting in low rates of adoption. Furthermore in today's systems, mobile users experience artifacts of mobility which impair service availability and quality. Thus, moving between such diverse environments as wired and cellular networks burdens users with administrative tasks. Multimedia and streaming media applications in particular exhibit sub-optimal quality or experience complete loss of services under varying network conditions or network transitions.

To address these issues, we propose a network virtualization layer with three main responsibilities in mobile scenarios: (1) to relieve the user of system re-configuration due to a changing networking environment (e.g., the transition from a wireless LAN to a GSM-based connection); (2) to hide changes in the networking environment from legacy applications; (3) to perform a mapping between legacy traffic and richer communication paradigms ranging from solutions for enhanced multimedia services and service composition to overlay-based routing or IPv4/IPv6 tunneling.

Our architecture achieves these tasks by allowing flexible protocol transformations on the end-system and leveraging overlay routing systems, service composition frameworks, and higher-level protocols in general at the network level. On the end system, network traffic needs to be intercepted, analyzed, transformed if necessary, and forwarded without requiring changes to applications or the operating system. At the network level, we use overlay routing services such as i3 [12] which provide mobility support and generic support for service composition. By combining both aspects, multi-
media and other services can be delivered seamlessly to mobile legacy applications. This forms a flexible research platform for evaluating new multimedia protocols and frameworks with legacy applications in mobile environments.

While our architecture is generic and applies to a large variety of applications, this paper discusses opportunities and challenges of using it for mobile multimedia and service composition. It is organized as follows: section 2 introduces an application scenario which illustrates the necessity for seamless mobility support for multimedia applications; our architecture addressing this challenge is described in section 3; section 4 discusses architectural issues arising specifically in the context of mobile multimedia applications; section 5 analyzes related work and section 6 concludes.

2. MOBILE MULTIMEDIA SCENARIOS

Many mobile applications serve as prime scenarios for motivating multimedia service composition. Here, the user experience of multimedia services can particularly benefit from customization and adaptation of contents: data needs to be available in formats matching the capabilities of mobile devices. Furthermore, formats and their properties should adapt to the changing social, networking, and administrative environments in which the users move. In the remainder of this paper, we will use the following three scenarios to illustrate the challenges of mobile multimedia applications (cf. Figure 1).

In scenario A, a company employee uses a hand-held device in a docking station for a video conference with colleagues. When the device is undocked in scenario B, it should switch to the company wireless LAN, perform the necessary authentication through a VPN client, and adjust the quality of the video and audio streams to match the new connection properties. On their way home, the user may be engaged in a VoIP call with a friend and enter a cafe after leaving the company site (scenario C). Here, the device can choose between GPRS and a commercial Wi-Fi hotspot to maintain connectivity. Next to the VoIP or video conference application, the user then starts to download a file from a company server for which additional encryption is desired.

These scenarios call for modular solutions to implement multimedia services. The costs and effort of re-implementing components and frameworks for each service and system in a monolithic manner are prohibitive. Ideally, these service components can be orchestrated into service compounds, which are delivered to the end user. Our architecture facilitates and augments such solutions to make them available to legacy applications. It also allows to enrich legacy services to increase the functionality or enhance their quality.

The changes of network links and their properties due to user mobility manifest in two aspects. The first aspect is that connectivity needs to be maintained, not only by configuring network devices appropriately, but also by adapting to protocol requirements. From the scenario above, the transition from the wired to wireless link including the necessary VPN authentication illustrates this point. Therefore, the first goal of our architecture is to automate this tasks and require as little administrative interaction with the user as desired. The second aspect of mobility artifacts applies to applications. Most applications use TCP or UDP over IPv4 and can not handle mobility seamlessly. Thus, our second goal is to provide mobility support for legacy applications.

The third goal is to leverage new overlay networks and network architectures. In a multimedia context, this would allow novel protocols and frameworks (such as [6, 4]) to provide service composition and QoS guarantees to, e.g., a legacy video player. Furthermore, the reuse of legacy applications can significantly reduce the effort required to create realistic test and evaluation environments. Thus, our architecture can serve as a research platform to ease the development and evaluation of new protocols, platforms, and distributed systems.

3. ARCHITECTURE OVERVIEW

The overall goal of our end-system architecture is to introduce new protocols and additional functionality into the standard network stack without requiring changes to legacy applications or operating systems. We call the software implementing these features (and instances of it) a proxy as it is responsible for intercepting and relaying network traffic. In contrast to remotely deployed application-level proxies (e.g., remote caching HTTP proxies), our proxy resides on end systems and intercepts network traffic locally.

The network-related part of our architecture augments legacy protocols and applications with additional functionality. It exploits network-based services such as routing, transcoding, or encryption. In particular, the i3 overlay routing infrastructure provides support for host mobility [19] and service composition [13], and further protocols can be layered on top of it.

Figure 1: Example scenario of mobile multimedia applications.
3.1 End-System Architecture

As illustrated in Figure 2, the two main components of the proxy are a packet filter and a set of protocol transformers. While standard protocols like TCP/IP are implemented as part of the operating system kernel, the proxy is a regular user-level application. This fact substantially reduces the effort of developing and debugging protocol transformers and new protocols within the proxy. Depending on the underlying operating system, the packet filter cannot be integrated directly with the proxy but must instead be implemented as an in-kernel component. However, the packet filter and the protocol transformers connect through a generic interface which abstracts from these platform-dependent details.

3.1.1 Packet Interception

The packet filter is responsible for intercepting packets going from applications to the network and vice versa. Here, interception means that packets leave their normal flow of processing in the operating system and are delivered to the proxy. Furthermore, it must be possible to inject arbitrary packets into this flow. In scenario B, the packets of the video-conferencing application need to be transmitted through the VPN tunnel. For outgoing packets, the packet filter intercepts them before they leave the system. It passes them to the proxy which encapsulates them with the VPN protocol and returns them to the packet filter. The filter then injects them into the network stack of the operating system from where they are sent to the wireless network. Incoming packets are handled symmetrically to decapsulate them and pass them to the conferencing application.

On many Unix systems, tun/tap devices [16] allow us to implement this form of packet interception as part of the proxy implementation and no in-kernel code is necessary. On Microsoft Windows systems, the proxy uses a Windows driver providing similar functionality to tun/tap devices. The driver also allows for conditional packet interception in order to deliver only relevant packets to the proxy application. The implementation as a driver requires no changes in the operating system.

3.1.2 Application Transparency

The support for unmodified legacy applications is a central aspect of our end-system architecture. It is achieved by leaving the application programming interface and application binary interface between application and operating system intact. Instead, the proxy only interacts with applications by intercepting and relaying their network traffic. Thus, application transparency needs to be ensured at the protocol level.

Due to the almost exclusive use of the IPv4 protocol in legacy applications and the benefits of i3 for mobility and service composition, the mechanisms for protocol transparency will be briefly illustrated on the example of this combination of protocols. While tunneling IP traffic over i3 itself is straightforward, service discovery is not because i3 communication endpoints are not identified by pairs of IP addresses and port numbers. Thus, the proxy associates i3 endpoints with unused virtual IP addresses (e.g., from a private address range such as 10.0.0.0/8). All traffic from an i3 endpoint is modified to appear to originate from a host with the associated virtual IP address. Conversely, packets with virtual destination IP addresses are encapsulated and tunneled to the associated i3 endpoint.

Virtual IP addresses are provided to applications by intercepting name resolution attempts such as DNS queries. These legacy mechanisms can thus be augmented with other schemes for name resolution and service discovery. They can range from hashing the DNS name locally into an i3 endpoint identifier to, e.g., complex QoS-aware negotiation protocols for locating services and composing communication paths.

3.1.3 Protocol Transformation

Our end-system architecture is structured such that protocol transformations can be stacked on each other. After a packet is intercepted by the packet filter, it is fed to the transformation stack. The transformation modules interact through a generic interface for passing the modified packet on to the next module. Eventually, packets are either dropped or injected back into the regular network stack of the host operating system. This lends itself to the fact that different transformations apply to different protocol layers, as shown in Table 1. Service-composition and multimedia frameworks and protocols would typically be implemented at the application and transport layers.

As illustrated in Figure 3, protocol transformations can also be applied selectively. For example in scenario B, the internal company file server can be accessed directly over the wireless LAN. The video conferencing traffic going to the Internet is handled by the VPN module for connectivity, authentication, and encryption. The conferencing application in turn can be enhanced with higher-level transformations, e.g., QoS management or multicast and mobility support.

Table 1: Protocol transformations can be grouped by network layer.

<table>
<thead>
<tr>
<th>Layer</th>
<th>Transformation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Adaptation and transformation of data and protocols, integration of new protocols, service discovery &amp; composition</td>
</tr>
<tr>
<td>Transport</td>
<td>Adaptation to link properties, link maintenance, QoS management</td>
</tr>
<tr>
<td>Network</td>
<td>Mobility support, (overlay) routing, encryption, authentication</td>
</tr>
<tr>
<td>Adapter</td>
<td>Link detection &amp; selection</td>
</tr>
</tbody>
</table>

Figure 2: Proxy architecture with packet filter and protocol transformers.
3.1.4 Adaptability

Based on external events or user intervention, modules for protocol transformation can be inserted to and removed from the transformation stack at run time. In concert with automatic configuration of network devices, a host becomes significantly more adaptable to a changing network environment. This applies equally to host mobility, such as vertical switch-overs (e.g., WLAN to GPRS), infrastructural requirements (e.g., VPNs, pay-per-use access), and fluctuations in link quality. Thus, the need for administrative action from the user in mobile environments can be reduced or eliminated.

Since automatic adaptation can exhibit side-effects unwanted by the user, we introduce a policy-based approach. A policy, as defined by the user, controls and restricts the actions the proxy may take. As outlined in scenario C in the cafe, the system may have a choice between an expensive but high-bandwidth Wi-Fi connection and the cheaper GPRS link with lower throughput. In such a situation, the proxy itself can detect a bandwidth-intensive application such as the download and thus provide the user with the faster connection. However, if the download is not of importance to the user, he can activate a low-cost policy forcing the proxy to choose the less expensive connection.

3.2 Network Architecture

The network-related part of our architecture utilizes the overlay routing services of the Internet Indirection Infrastructure i3 [12]. Its core idea is to communicate across one or more points of indirection which stands in contrast to end-to-end communication. This scheme decouples the act of sending from the act of receiving and can thus provide additional features like multicast, anycast, mobility support, or service composition.

Every point of indirection is identified by a unique ID in the form of a large integer or fixed-length bit string, respectively. Data packets carry an ID instead of a real IP address as the destination address. Thus with i3, data is addressed to an abstract notion of a service instead of a particular end host. In order to receive data via i3, hosts register so-called triggers with the i3 system. A trigger is an association of a destination ID with an IP address/port pair or another ID. i3 forwards all packets going to an ID to the trigger addresses registered with this ID. In a simple example, a receiver inserts a trigger associating an ID with the IP address and a port it listens. Accordingly, i3 delivers all data sent to the ID to the receiver.

Mobility support in i3 is based on the addressing scheme of using IDs instead of IP addresses. When a mobile host moves between networks and receives different IP addresses, it updates its i3 triggers accordingly. Consequently, the host remains accessible at the i3 level. i3 allows receivers to insert more than one trigger per ID, so the ID itself remains unique but is associated with multiple forwarding addresses. The packets which are sent to such an ID are forwarded to every associated trigger address, which effectively implements multicast communication. For service composition, i3 generalizes the concept of IDs to ID stacks. A packet with a destination ID stack must traverse all the triggers referenced in the stack, which can be regarded as source routing. Similarly, forwarding entries in triggers can also be ID stacks so a forwarded packet must go through all the IDs in the stack. Thus, both senders and receivers can control the route the packet takes including services and transformations the packet needs to traverse.

NAT gateways and firewalls do not limit the reachability of i3 clients, as long as outbound connections are permitted. In scenario C, IP connections from the Internet to the hand-held device can be blocked by the Wi-Fi firewall and NAT configuration. However, the device can still establish a connection to the Internet-based i3 service and the device’s triggers are associated with this connection instead of its (unreachable private) IP address. Thus, i3 packets can reach the device despite NAT and a firewall.

In the proxy, i3 is implemented as a transformation module and is thus an optional component. However, its flexibility and functionality at the routing layer makes it an ideal addition to our architecture. Furthermore, higher-level protocols can exploit its features and its generic support for service composition.

4. DISCUSSION

This section analyzes the challenges of using our proxy architecture in a multimedia context.

4.1 Inferring Application Requirements

QoS and service-composition frameworks often rely on applications to explicitly indicate their requirements and capabilities. In many cases, feedback cycles between layers allow to determine the best compromise between user demands and application and network properties. For example, the video conferencing application can request a maximum acceptable latency and a minimum video frame rate from the service layer. This layer may then choose an appropriate encoding and decide whether additional services, e.g., subtitles for a video can meet these requirements.

By design, our proxy focuses on legacy applications and avoids direct interaction with applications. Thus, the application layer does not explicitly provide service specifications or requirements. Instead, this information must be inferred by the proxy itself. In many cases, it is sufficient to derive this data implicitly from application and system behavior. For example, the necessity to transcode between media formats can be deduced from the service being requested (a specific video), the requested format (e.g., AVI), and the actual format of the service (e.g., MPEG). The need for service composition can also arise from a changing operating environment. For example, the transition from a company
network to a public network, as in scenario C, may trigger the activation of an encryption service.

The user’s requirements for individual services can also be indicated explicitly to the proxy. First, the proxy may provide configuration dialogs for specific services or service classes. For example, the proxy may export a setting which controls whether text sub-titles for video streams are displayed or not, even if the legacy video player application is unaware of such a choice. While such external configuration may hamper usability to a certain degree, it may be acceptable for evaluation purposes or where there is no alternative to using a certain legacy application. Second, legacy name resolution can be exploited for service specification. Instead of passing regular URLs to legacy applications, URLs formatted to contain service composition paths and requirements can be used. While the application remains agnostic to this format, a service composition framework in the proxy can utilize the encoded information. However, such a possibly complex URL format is cumbersome to handle.

Multimedia applications depend on several properties of the whole system, such as available resources and network link characteristics. The proxy can centrally aggregate such properties and supply them to protocols implemented within the proxy. Where resource contention is an issue, resource allocation schemes can also be implemented centrally.

Quality-of-Service constraints can be inferred to a certain extent through observation of system behavior. Based on these observations, QoS parameters can be adjusted to provide higher quality to the user or to better utilize available resources. For example, the transition from the wired company network to the wireless LAN could result in low CPU utilization and high network utilization. This information indicates that the streaming video attempts to consume more network bandwidth than available. Switching to a computationally more complex compression scheme can result in a higher effective frame rate, i.e., better quality provided to the user.

4.2 Flow Identification

Since individual applications and their network connections have different requirements, the proxy must be able to differentiate between them. For example, the company wireless LAN may allow unrestricted access to internal services while the Internet is only accessible after authenticating with a VPN. The proxy can support such an environment with selective protocol transformation by running only remote traffic through the VPN. Similarly, the user may place different demands on different multimedia streams based on customized policies (e.g., giving the video conference application a higher priority than another background video stream). Consequently, the proxy must identify these streams and handle them individually in order to meet user demands.

The more accurate flow identification needs to be, the more knowledge about protocols and analysis of traffic is necessary. In simple cases, such as the VPN example, traffic flows can be distinguished based on transport-level information, i.e., IP address and ports. Closer analysis is required for multi-flow protocols like SCTP. It is to be noted that such a detailed packet inspection need not be implemented in the proxy in general but only in the respective transformation modules.

4.3 Performance

The structure of our proxy imposes a processing overhead for network traffic on end hosts. This overhead is due to intercepting, parsing, and processing packets in the proxy. At the current stage of implementation, no experimental results are available for a performance evaluation. Thus, a quantitative analysis follows.

Each intercepted packet is transferred from the operating system’s network stack to the proxy for further processing. The proxy analyzes the packet to determine whether it is to be forwarded unmodified or passed to the transformation stack. Eventually, the proxy injects the packet back into the regular network stack. Thus, packet interception causes two context switches and two additional copies of each packet. Since data rates are low for mobile devices with wireless links, this overhead is assumed to be negligible.

Analyzing packets and forwarding them between transformation modules can be assumed to cause only a very modest performance impact. These operations are comparable in cost to those performed in the operating system’s network stack. The encapsulation of a packet increases its size on the wire. For large packets, this can lead to additional packet fragmentation. Packet processing in transformation modules is potentially expensive but may not be regarded as overhead introduced by the proxy architecture itself.

5. RELATED WORK

Implementing and evaluating network protocols at user level has been an issue in operating system and network research for a long time [15, 8, 11, 9]. Where these solutions strive to replace kernel-level protocol stacks, they trade API compatibility for performance or security. In contrast, the support for legacy applications is a primary concern of our approach. Other than evaluation approaches like Alpine [1], our end-system architecture does not attempt to replicate real execution environments for protocol implementations. Thus, it can be used on several platforms and remains more lightweight. Application-transparent architectures like CANS [3] or Conductor [18] share goals with our approach in hiding the mobility artifacts and supporting legacy applications. However, they are tied to their network architectures and intercept network traffic at the interface between application and operating system. While this results in fewer context switches and better performance, these solutions are heavily system dependent and require significantly more engineering effort.

Commercial applications like the ipUnplugged Roaming Client [5] achieve seamless connectivity with similar techniques for packet interception and redirection as ours. However, they solely focus on VPN and IPsec solutions and cannot serve as a generic research platform.

Delay-tolerant networking (DTN) [2] addresses the effects of mobility stemming from network fragmentation or disconnected operation. We assume our application scenarios to be typically faced with widely varying degrees of link qualities and properties rather than with longer periods of no connectivity. Thus, we view the work on DTN as being orthogonal to ours which could be very well integrated with the proxy.

Our architecture borrows substantially from the i3 proxy [7], including IP address virtualization [14] and DNS rewit-
ing [17]. While the i3 proxy aims at redirecting legacy traffic via i3, our solution provides a framework for arbitrary network modifications, essentially a user-level network stack.

6. SUMMARY

While multimedia services and the composition of such services have been a long standing research topic, it remains difficult to evaluate and deploy new protocols, frameworks, and middleware systems in this area. We propose a research platform with an end-host-based architecture for network virtualization. It allows network traffic to be transformed at the user level while maintaining transparency towards legacy applications. This system significantly simplifies protocol deployment and evaluation, the adaptation to changing network environments, and extensions to legacy services. In mobile settings, it can hide mobility artifacts from users as well as legacy applications and support QoS and service composition.

7. REFERENCES

Resource-Aware Service Composition for Video Multicast to Heterogeneous Mobile Users

Shuichi Yamaoka, Tao Sun, Morihiko Tamai, Keiichi Yasumoto, Naoki Shibata, and Minoru Ito

Graduate School of Information Science, Nara Institute of Science and Technology
Ikoma, Nara 630-0192, Japan

{shuich-y,song-t,morihi-t,yasumoto,ito}@is.naist.jp

† Department of Information Processing and Management, Shiga University
Hikone, Shiga 522-8522, Japan

shibata@biwako.shiga-u.ac.jp

1. INTRODUCTION

Recent innovation and widespread of wireless network technologies have realized various types of portable computing devices (which we call mobile terminals/nodes, hereafter), such as laptop PCs, PDAs, and cell phones capable of connecting to the Internet. A lot of networked applications such as Web browsing, e-mail, downloads of high-quality music files, on-line games are available for these mobile terminals. Among them, video streaming is one of the most promising applications. The computation power of micro processors used in mobile terminals is increasing year by year, and it is now sufficient to play back streaming video in real-time. There are wide variety of screen size, computation power, power consumption, battery amount, maximum network bandwidth in these devices. Moreover, data transmission rate and the access point connecting to the Internet changes as they move. From these reasons, in order to broadcast video to multiple mobile terminals simultaneously, the following criteria should be considered and realized: (1) depending on constraints such as screen size, processing power, remaining battery amount, and possible transmission rate, each mobile terminal should be able to decide an appropriate quality of video (called required quality) to be received; (2) each mobile terminal can change its required quality any time during playback of the video; (3) contents provider should deliver video data of the requested quality; and (4) each mobile terminal can play back video smoothly and continuously even when the access point changes as a consequence of its movement. Letting a server deliver video to multiple mobile terminals by simultaneous unicast streams consumes a lot of computation and network resources. So, (5) it would also be very important to save the resource amount consumed in a content delivery network (CDN) so that the amounts of them are minimized.

There are many research efforts aiming at efficient broadcast of a video to multiple user nodes with different quality requirements. In the most promising technique (e.g., [1]), video data is encoded as a base layer and several extended layers using a hierarchical encoding technique such as MPEG-4 FGS [2] and those layers are broadcasted as separate multicast streams so that each mobile terminal can receive the base layer and a part of extended layers within its available bandwidth to playback video with the quality corresponding to the bandwidth. In this technique, however, extra memory is required for buffering all of receiving layers. More computation power than decoding a single layered
video is also needed. Also, this technique has some drawbacks: the difference between the required quality and the received quality is large when there are only a small number of layers. Also, it can only convert bitrate of video. Picture size and frame rate are fixed to the original value.

In this paper, we propose a new service composition method for efficient delivery of a video to multiple mobile nodes satisfying the above criteria (1) to (5). In the proposed method, we assume the following environments: (i) an overlay network connecting multiple proxies and a video server as shown in Fig. 1 is given as CDN. A fixed amount of bandwidth is assigned to each overlay link between proxies (or connecting to a server node) using existing network level QoS techniques such as Diffserv [3]; (ii) each proxy can execute multiple transcoding services and forwarding services within its available resources; (iii) To each proxy, at most one wireless access point (AP) can be attached; and (iv) each mobile node communicates with a proxy via the corresponding AP which is automatically and uniquely determined based on the current position of the mobile node.

To achieve the criteria (1) and (3), the proposed method utilizes transcoding service running on proxies for transcoding video to lower quality video. It also utilizes forwarding services on proxies to forward video to mobile nodes or to other proxies for transcoding the video to further low quality. To achieve the criterion (5), we propose an algorithm to calculate service delivery paths among a server, proxies and mobile nodes (i.e., a set of delivery paths) on the overlay network as well as input/output video parameters (picture size, frame rate and bitrate) of each proxy so that the total resources consumed (both computation and network resources) will be as small as possible.

In the proposed method, the above criterion (1) is achieved using our energy-aware video streaming technique proposed in [4]. Here, appropriate quality (i.e., vector of picture size, frame rate, and bitrate) of each segment of the video is automatically determined from (a) the user’s requirement consisting of playback duration (e.g., time length of the video), relative importance among video segments and preferable ratio between picture size and frame rate for each segment, and (b) constraints of the mobile terminal consisting of remaining battery amount, available network bandwidth, computation power and screen size. For the criterion (2), we also propose a protocol for periodical reconstruction of new service delivery paths with the latest quality requirements from all mobile nodes. The service delivery paths are reconstructed seamlessly whenever each new video segment is played back. To achieve the above criterion (4), when the current AP (thus the corresponding proxy, too) of a mobile node changes, it immediately starts to receive the video already delivered to the new proxy where the video with the quality closest to (and less than) its required quality is selected. By the next reconstruction of the service delivery paths, the mobile node will be able to receive video with the quality closer to its requirement.

We have implemented the proposed algorithm and measured the consumed resource amount in the overlay network with simulations. As a result, we have confirmed that the proposed method can achieve efficient video delivery to heterogeneous mobile users at low cost, satisfying user’s quality requirements.
[13] proposes an algorithm to calculate efficient service delivery paths by concatenating a multicast tree connecting proxies and local multicast trees consisting of user nodes so that resource consumptions of those trees are minimized. In this method, each local multicast tree is connected to the proxy so that the service delivery path via the proxy has the smallest physical hop count and the larger available bandwidth. However, it does not consider the optimization of resource consumption among multiple service delivery paths on overlay links between proxies nor mobility of user nodes. Our proposed method is different from the above existing studies, since it achieves more flexible service composition where multimedia data can be delivered through the efficient service delivery paths to multiple heterogeneous mobile users whose quality requirements and locations dynamically change.

2. PROBLEM DEFINITION

In this section, we describe the target environments and assumptions, and then formally define the problem to deliver video to multiple heterogeneous mobile nodes using the service composition technique.

2.1 Target Environments and Assumptions

In our propose method, we assume the existence of a content server, an overlay network, mobile nodes, service components and proxies. We assume the followings:

1. Content server: A server transmits a recorded or live video (original video) to other nodes. Mobile user's quality requests are lower than the quality of original video. Starting time of the broadcast is predetermined similarly to TV broadcast.

2. Overlay network: An overlay network consisting of a video server, multiple proxies, multiple wireless access points (called AP, hereafter) and multiple mobile nodes is given in advance (Fig. 1). Here, a certain amount of bandwidth is reserved for video delivery on each overlay link using network level QoS techniques such as DiffServ [3]. At most one AP is attached to each proxy (multiple APs attached to a proxy can be regarded as one AP whose transmission bandwidth is the sum of their bandwidths). Available bandwidth between each proxy and the corresponding AP is larger than the upstream bandwidth of the proxy. Available bandwidth between an AP and the corresponding mobile nodes is larger than the sum of the transmission bandwidths of mobile nodes. Therefore, these links will not be bottlenecks during video delivery.

3. Mobile node: There are multiple mobile nodes (e.g., laptops, PDAs, cell phones, etc) which have different screen sizes, computation powers, available transmission speeds, and so on. They can communicate with a proxy via corresponding AP only when its radio range covers location of the mobile node. The corresponding AP can be uniquely determined from the location of each mobile node, and each mobile node immediately notices that it moves into a radio range of another AP. Mobile nodes do not exchange messages directly.

4. Service: There are two kinds of service components: (i) transcoding service and (ii) forwarding service. The computation powers required to execute these services can be calculated depending on input/output quality of the video and the input/output bitrates of the video, respectively.

5. Proxy: Each proxy has the maximum computation resources (CPU power, memory amount, and so on). Within these capacities, each proxy can instantiate arbitrary number of service components. In this paper, for the sake of simplicity, we treat only the computation power (i.e., CPU usage) required for execution of transcoding services.

2.2 Formal Definition of Problem

In this section, first, we present the notation of parameters used in the rest of this paper. Then, we formally define the problem.

2.2.1 Notation and definition

Overlay network
Let \( s, P = \{p_1, p_2, \ldots, p_n\} \), and \( U = \{u_1, \ldots, u_m\} \) denote a server, the set of proxies, and the set of mobile nodes, respectively. If a mobile node \( u \in U \) is in the radio range of an AP and can communicate with a proxy \( p \in P \) through the AP, we regard that there is an overlay link between \( u \) and \( p \) and it is denoted by \((u, p)\). Let \( W \) denote the set of overlay links connecting to mobile nodes. Note that \( W \) changes as mobile nodes move. Let \( F \) denote the set of overlay links between nodes of \( \{s\} \cup P \). Let \( V = P \cup U \cup \{s\} \) and \( E = W \cup F \) denote the set of all nodes and the set of all overlay links, respectively. Then, an overlay network is represented as a graph \( G = (V, E) \). We denote the maximum available computation resource of each proxy \( p \in P \) by \( c_{\text{avail}}(p) \), and the maximum available bandwidth of each overlay link \((p_i, p_j)\) by \( b_{\text{avail}}(p_i, p_j) \) where \( p_i \in P \) and \( p_j \in \{s\} \cup P \). We denote the physical hop count of \((p_i, p_j)\) by \( h_{\text{hop}}(p_i, p_j) \).

As we stated before, we assume that the maximum available bandwidth of each link \( w \in W \) is not limited. An example overlay network is shown in Fig. 2 (a).

Required resource to execute transcoding service
We assume that the quality of video depends only on picture size (number of pixels), frame rate and bitrate. We denote these parameters by \( q, s, f \) and \( b \), respectively, and hereafter we call the quality of video the quality vector denoted by \( q = (q, s, f, b) \). We assume that the required computation power to transcode the video with quality vector \( q \) to the video with \( q' \) can be represented as the sum of the powers decoding the video (including the power for processing the decoded pictures) with quality vector \( q \) and encoding the video with \( q' \). We also assume that the required powers for decoding and encoding video are proportional to the number of pixels processed per unit of time based on the result in [4]. According to the above discussion, with some device specific constants \( \tau_d \) and \( \tau_e \), the computation powers required for decoding and encoding video are represented by the following expressions.

\[
\begin{align*}
    r_{\text{decode}}(q) &= \tau_d \times (q, s, q, f) \\
    r_{\text{encode}}(q') &= \tau_e \times (q', s, q', f)
\end{align*}
\]

Constraints on service paths
We want to calculate sequences of proxies with input/output quality vectors of video at each proxy to form so-called service paths from server \( s \) to all of mobile nodes \( U \). On each
service path, constraints on maximum available computation power at each proxy and the maximum available bandwidth on each overlay link must be satisfied. In Fig. 2 (b), we show an example of service paths for the overlay network of Fig. 2 (a), where mobile nodes $u_1, u_2, u_3, u_4, u_5, u_6$ and $u_7$ require video delivery with quality vectors $q_1, q_2, q_3, q_4, q_5$ and $q_6$, respectively. Here, $c_{avail}(p_1,p_2)$, $b_{avail}(p_1,p_2)$, $hop(p_1,p_2)$ and $c_{avail}(p_2,p_3)$, $b_{avail}(p_2,p_3)$, $hop(p_2,p_3)$, $c_{avail}(p_1,p_6)$, $b_{avail}(p_1,p_6)$, $hop(p_1,p_6)$, $c_{avail}(p_3,p_4)$, $b_{avail}(p_3,p_4)$, $hop(p_3,p_4)$, $c_{avail}(p_4,p_5)$, $b_{avail}(p_4,p_5)$, $hop(p_4,p_5)$, $c_{avail}(p_5,p_6)$, $b_{avail}(p_5,p_6)$, $hop(p_5,p_6)$, $c_{avail}(p_6,p_7)$, $b_{avail}(p_6,p_7)$, $hop(p_6,p_7)$, $c_{avail}(p_7)$, $b_{avail}(p_7)$, $hop(p_7)$, $c_{avail}(p_5)$, $b_{avail}(p_5)$, $hop(p_5)$, $c_{avail}(p_4)$, $b_{avail}(p_4)$, $hop(p_4)$, $c_{avail}(p_3)$, $b_{avail}(p_3)$, $hop(p_3)$, $c_{avail}(p_2)$, $b_{avail}(p_2)$, $hop(p_2)$, $c_{avail}(p_1)$, $b_{avail}(p_1)$, $hop(p_1)$, $c_{avail}(p_6)$, $b_{avail}(p_6)$, $hop(p_6)$, $c_{avail}(p_5)$, $b_{avail}(p_5)$, $hop(p_5)$, $c_{avail}(p_4)$, $b_{avail}(p_4)$, $hop(p_4)$, $c_{avail}(p_3)$, $b_{avail}(p_3)$, $hop(p_3)$, $c_{avail}(p_2)$, $b_{avail}(p_2)$, $hop(p_2)$, $c_{avail}(p_1)$, $b_{avail}(p_1)$, $hop(p_1)$.

Figure 2: Example of overlay network topology

2.2.2 Problem Definition

The problem is to calculate the set of quality vectors $R(p_i)$ of the videos which each proxy $p_i$ receives, and the set of all forwarding relationships $(v',q') \rightarrow (v,q)$ $(v' \in \{s\} \cup P, v \in P \cup U)$, satisfying the following constraints (3)–(6), when an overlay network $G = (V,E)$, the quality vector of the original video $q_{orig}$, and quality requirement $q_u$ of each mobile node $u \in U$ are given.

\[
\text{for each } (v',q') \rightarrow (v,q), \quad q_s \leq q_s \wedge q.f \leq q', f \wedge q.b \leq q' \cdot b
\]  

\[
\text{for each } u_i \in U, \quad \exists (s, p_{j1}) \exists (p_{j2}, p_{j2}) \ldots \exists (p_{jk}, u_i) \text{ s.t.}
\]

\[
(s, q_{orig}) \rightarrow (p_{j1}, q_1) \wedge (p_{j2}, q_1) \rightarrow (p_{j2}, q_2) \wedge \ldots \wedge (p_{jk}, q_k) \rightarrow (u_i, q_u)
\]

\[
\text{for each } p_i \in P, c_{cons}(p_i) \leq c_{avail}(p_i)
\]

\[
\text{for each } (p_i, p_j) \in F, b_{cons}(p_i, p_j) \leq b_{avail}(p_i, p_j)
\]

Constraint (3) represents that the parent node $v'$ must receive the video with the higher quality vector in all quality parameters than $v$ for transcoding and forwarding. Constraint (4) represents that there must be the sequence of overlay links connecting the server $s$ and each mobile node $u \in U$ via a set of proxies. Constraints (5) and (6) represent that the consumed computation power at each proxy $p_i$ and the consumed bandwidth on each overlay link $(p_i, p_j)$ or $(s, p_i)$ must not exceed the predetermined capacities.

In general, there may be multiple solutions which satisfy the above constraints. So, we use the following objective function to minimize the amount of consumed resources.

\[
\text{Min} \left( \alpha \sum_{p \in P} c_{cons}(p) + (1 - \alpha) \sum_{(p_i, p_j) \in F} b_{cons}(p_i, p_j) \times \text{hops}(p_i, p_j) \right)
\]

The first term of the objective function (7) represents the total sum of computation power consumed at proxies,
and the second term does that of bandwidth consumed on overlay links between proxies considering their physical hop counts. Here, \(\alpha\) is used to make which kinds of resources more expensive.

3. System Behavior

In this section, we describe how the whole proposed system works in detail. In Sect. 3.1, we will explain how each mobile node makes a request of its video quality taking into account the amount of remaining battery. In Sect. 3.2, we explain a grouping method of quality requirements to reduce the number of different quality videos for pre-processing. And in Sect. 3.3, we will explain the communication protocol between nodes for video transfer.

3.1 Determining Required Quality Based on Battery Amount

Let \(bw_u\), \(cp_u\), \(ds_u\) and \(en_u\) denote available receiving bandwidth, available processing power, screen size and the amount of remaining battery for a mobile node \(u \in U\), respectively. The request from a node \(u\) has to satisfy following restrictions.

\[
q_u.b \leq bw_u
\]
\[
q_u.s \leq ds_u
\]
\[
\tau_d \times (q_u.s \times q_u.f) \leq cp_u
\]

If the user of node \(u\) specifies time of duration \(T_u\) for watching a video, the amount of remaining battery has to be considered to decide video quality. We have already proposed a method to find a suitable video quality (a combination of screen size, frame rate and bitrate) for mobile terminal from time of duration \(T_u\), the amount of remaining battery \(en_u\) and constants inherent to the model of mobile terminal (e.g. power consumption of running OS, and so on) in [4]. If the video is not a live video and recorded beforehand, video segments in the video is known, and we assume that the contents provider assigns keywords to each segment by automatic labeling tools such as [14]. In this case, quality of each video segment can be changed according to relative importance of each video segment and preferred playback characteristics (faster framerate or higher resolution) specified by the user.

Hereafter, we describe how a user node decides video quality.

\[
C = \{c_1, ..., c_m\}
\]

is a set of categories (e.g., keywords) assigned to video segments. Each user specifies a relative importance \(p_i\) for each category in \(C\). Here, \(p_i\) is an integer value larger than 0. The amount \(en_u\) of remaining battery is distributed among categories proportional to the product of total length \(T_i\) and specified importance \(p_i\) of category \(c_i\).

That is, \[
\frac{en_u \cdot p_i \cdot T_i}{\sum_{i=1}^{m} p_i \cdot T_i}
\]
is the amount of battery used for playing back video segments which belong to a category \(c_i\). As we described before, playback quality can be differentiated by specifying different playback characteristics, even if the amount of battery used for playback is same. In order to achieve this, the user specifies a ratio between screen size and framerate \(q_u.s / q_u.f = x : y\) for each category. Here, \(x\) and \(y\) are integer numbers larger than 0.

The quality of each video segment can be calculated using the method in [4]. We explain this method by an example soccer video consists of three categories \{shoot, play, other\}. Suppose a user specifies that he wants to see shoot scenes in higher quality, play scenes in medium quality, and other scenes in lower quality. Both screen size and framerate are important for shoot scenes, framerate is more important in play scenes, and screen resolution is more important in other scenes. In this case, he specifies as follows:

<table>
<thead>
<tr>
<th>category</th>
<th>importance</th>
<th>(\frac{q_u.s}{q_u.f})</th>
<th>(\frac{4}{1})</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>shoot</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>10min</td>
</tr>
<tr>
<td>play</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>35min</td>
</tr>
<tr>
<td>other</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>15min</td>
</tr>
</tbody>
</table>

Scenes in the category shoot are played back using \(\frac{en_u \cdot 4}{1} = \frac{4}{en_u}\) of the remaining battery amount, and thus these scenes are played back in higher quality than others.

In the proposed method, video quality can be decided by the method explained above, for recorded video. On the other hand, when a live video is broadcasted, the method above cannot be used since categories of the video segments and their total lengths cannot be known beforehand. In this case, each user specifies a quality from a few levels (e.g. the user selects from High, Medium and Low). If medium quality is specified, the system decides video quality so that the video can be played back using all of the amount of remaining battery for remaining time of the video. If high or low quality is specified, the quality is decided by increasing or decreasing the standard playback power calculated for medium quality by predefined range (e.g. 20%). When the user changes quality specification, or predefined time passes since last change, the system updates the standard playback power.

3.2 Grouping quality requests

Video quality in which each user node receives can be calculated by the above method, but transcoding video too many different quality is not desirable in terms of processing power. Thus, in the proposed method, similar video qualities are grouped into a single video quality. This can be achieved by following steps. (1) Permissible difference range \(r\) of quality is specified to requested quality \(q_u.s, q_u.f, q_u.b\) of each mobile node \(u\), where \(r\) is calculated from restrictions to user’s satisfaction rate. For example, if satisfaction rate of a user is 0.95, permissible difference range \(r\) is \(1 – 0.95 = 0.05\). (2) Let \(S\) be a set of all quality requests. (3) For each mobile node \(u\), a set of quality requests \(S_u\) is calculated so that a quality request \(q_{u'} = (q_{u'.s}, q_{u'.f}, q_{u'.b}) \in S\) is an element of \(S_u\) if and only if \((1 - r).q_{u'.s} \leq q_{u'.s} \leq q_{u'.s} \land (1 - r).q_{u'.f} \leq q_{u'.f} \leq q_{u'.f} \land (1 - r).q_{u'.b} \leq q_{u'.b} \leq q_{u'.b}\). (4) Find the set with maximum number of elements, and exclude elements from \(S\). (5) The steps (3) and (4) are repeated until \(S\) becomes empty.

3.3 Video delivery protocol

The protocol consists of the part before starting video transfer, the part to reconstruct service path, the part used when a node joins or leaving the group, and the part used in handoff of a node between APs. First of all, we describe the protocol used before starting video delivery.

3.3.1 Starting video delivery

1. Let \(t\) be the starting time of video delivery. Before \(t\), each mobile node \(u\) whose user wants to watch the
video sends quality request \( q_s \) calculated by the method described in Sect. 3.1 to the connected proxy \( p \).

2. each proxy \( p \) sends received requests to the content server \( s \).

3. \( s \) does a grouping of all received requests by the method described in Sect. 3.2, and it decides the set of qualities \( E(p) \) to which \( p \) performs transcodings. Let \( q_p, s \) and \( q_p, f \) be the largest screen size and the largest frame rate in \( E(p) \), respectively. \( p \) receives a video stream with quality equal to or better than \( q_p, s, q_p, f, q_p, b \) from the upstream proxy. Bitrate of \( q_p \) can be calculated from screen size and frame rate by the method in [4]. \( p \) can now transcode this video stream to ones with any element in \( E(p) \).

4. \( s \) finds a set of service paths from received video qualities by the algorithm described in Sect. 4. \( s \) sends a message with the set of service paths to all proxies along the service paths. Each proxy starts all transcoding services and forwarding services after receiving the message.

5. At the time \( t \), server \( s \) starts transferring video stream along the service paths. Transcoding services transcode received video to the specified quality, and forwarding services relays video stream to their downstream proxies.

### 3.3.3 Joining and leaving of a node

A joining node \( u_{\text{new}} \) decides video quality \( q_{u_{\text{new}}} \) by the method described in Sect. 3.1. \( u_{\text{new}} \) sends join message including \( q_{u_{\text{new}}} \) to the connected proxy \( p \). \( p \) chooses a video quality close to \( q_{u_{\text{new}}} \) from qualities to which \( p \) is transcoding, and transfers the video to \( u_{\text{new}} \). The video quality is optimized at the next time of service path reconstruction.

If a mobile node \( u_{\text{new}} \) wants to stop receiving video, it can leave anytime. If the corresponding proxy has no other mobile nodes receiving video of the quality at which \( u_{\text{new}} \) were receiving, its transcoding service is stopped. Accordingly, the quality of video at which \( p \) receives from upper proxy can be changed. We will describe how to cope with this situation in Sect. 4.

### 3.3.4 Handoff of mobile node between APs

Each mobile node can move from the range of an AP to the range of another AP. In this case, proxy \( p \) compares requested quality \( q_{u_{\text{new}}} \) of the new node \( u_{\text{new}} \), and the quality \( q_p \) at which \( p \) is receiving from its upstream proxy. If \( q_{u_{\text{new}}}, s \geq q_p, s \land q_{u_{\text{new}}}, f \geq q_p, f \land q_{u_{\text{new}}}, b \geq q_p, b \), it is impossible to instantaneously starting sending video streams at the quality \( q_{u_{\text{new}}} \), and thus \( p \) temporarily sends video of \( q_p \) to \( u_{\text{new}} \). Video quality will be optimized at the next time of service path reconstruction.

If \( q_{u_{\text{new}}}, s \leq q_p, s \land q_{u_{\text{new}}}, f \leq q_p, f \land q_{u_{\text{new}}}, b \leq q_p, b \), either of the followings are performed.

- If \( q_{u_{\text{new}}} \) is in \( E(p) \), \( p \) simply sends an existing stream to \( u_{\text{new}} \), where \( E(p) \) is the set of qualities to which \( p \) performs transcodings. Otherwise, if there is remaining processing power, a new transcoding service is started, and a video stream of \( q_{u_{\text{new}}} \) is sent to \( u_{\text{new}} \). If there is no remaining processing power, the quality nearest to \( u_{\text{new}} \) is chosen from \( E(p) \), and sent to \( u_{\text{new}} \).

- The proxy chooses the element closest to \( q_{u_{\text{new}}} \), and transfers it.

Let \( D_p \) be the delay (or latency) of the service path from server to \( u \), where \( u \) is receiving video from proxy \( p \). Video can be played back seamlessly if \( D_p \leq D_{p'} \), where \( p' \) is the proxy for \( u \) after handoff. However, if \( D_p > D_{p'} \), there can be skip of video playback due to transcoding delay of \( D_p - D_{p'} \). This can be avoided by buffering video data at each proxy similarly to the process of service path reconstruction.

As a mobile node \( u \) moves, its AP and the corresponding proxy changes. If any mobile nodes are not connected to the new proxy, \( u \) does not receive any video from the proxy.

In order to cope with this problem, we slightly extend the algorithm as follows.

Let \( NB(p) \) denote the set of proxies whose APs are neighboring to \( p \)'s corresponding AP. If \( R(p) \neq \emptyset \), for each \( p' \in NB(p) \) such that \( R(p') = \emptyset \), we set \( R(p') = \{max(R(p))\} \). For proxies in \( NB(p) \), we do not apply this modification recursively. By this extension, whenever \( u \)'s AP changes, it can receive the required quality video. In this case, video data stream has to be sent faster than playback speed in order to absorb the difference.

### 4. SERVICE PATH CONSTRUCTION ALGORITHMS

In this section, we describe algorithms to calculate efficient service paths whose objective function defined in Sect. 2 is as small as possible. The inputs of algorithms are topology information of a given overlay network and the quality of video \( q_p = (q_p, s, q_p, f, q_p, b) \) which each proxy \( p \) must receive from its upstream proxy (see Sect. 3.3). Note that \( q_p \) is decided as the maximum quality requirement of user nodes connecting to \( p \). These algorithms are executed on
the server $s$, and its output is distributed to proxies in a way similar to that described in Sect. 3.3.1. The objective function is the weighted sum of the consumed computation power and the consumed network bandwidth. However, this minimization has a tradeoff. In order to minimize the total computational power, the number of transcoding services has to be minimized. In this case, however, if many users requesting the same quality video are distributed among different proxies, it may consume a lot of network bandwidth to deliver the video to those users. On the other hand, if we try to minimize the sum of the consumed network bandwidth, many transcoding services may have to be executed to provide various quality videos to user nodes. Finding the optimal solution to this problem is a combinatory optimization problem (i.e., NP-hard). So, we have to design a heuristic algorithm to solve this problem. Consequently, we adopt a policy to extend the existing heuristic algorithm to construct minimal spanning tree (Steiner tree) proposed in [16]. In Sect. 4.1, we will describe a basic algorithm which generates a set of service paths from a Steiner tree calculated by the method in [16]. Then, we describe two algorithms which minimize the sum of consumed network bandwidth and the sum of computation power respectively. Finally, we describe a hybrid algorithm based on these algorithms in Sect. 4.2.

4.1 Calculating service paths from Steiner tree

We call a proxy $p$ a parent proxy of $p'$, if $p'$ is directly receiving video streams from $p$. $p'$ is called a child proxy of $p$, if $p$ is a parent proxy of $p'$. We call a proxy directly receiving streams from the server $s$ a root proxy. We call a proxy which does not have a child proxy a leaf proxy.

The algorithm described in this section calculates a Steiner tree which minimizes the sum of hop counts of overlay links based on the algorithm in [16]. Since all overlay links on the calculated tree have to satisfy constraints (3) and (4) in Sect. 2.2.2, qualities of the received video streams by each proxy are adjusted. This process consists of following four steps.

**Step1.** Leaf proxy $p$ sends message $r_q$ which includes quality request $q_p$ (i.e., maximum quality requirement of user nodes connecting to $p$) to its parent proxy $p'$.

**Step2.** When $p'$ receives the messages from all of its child proxies, it compares each received quality $q_p$ with its own quality request $q_{p'}$. If $q_p, s \geq q_{p'}, s \vee q_p, f \geq q_{p'}, f \vee q_p, b \geq q_{p'}, b$, it adjusts $q_p$ so that $q_p = (\max(q_p, s, q_{p'}), \max(q_p, f, q_{p'}), \max(q_p, b, q_{p'}))$.

**Step3.** Step 2 is repeated until the message reaches a root proxy.

**Step4.** The root proxy sends $r_q$ to the server $s$.

4.1.1 Computation Power Minimization Algorithm

The case that the total sum of consumed computation power at proxies is minimized (i.e., $\alpha = 1$ in objective function (7)), is that only one transcoder is executed for each quality vector $q$ in a proxy among all proxies. If some mobile nodes have the required quality $q$ and they connect to the proxy $p$ which does not execute any transcoder for $q$, then, as shown in Fig. 3(a), another proxy $p'$ executing a transcoder for $q$ must forward the video to $p$ so that the mobile nodes can receive the video with $q$.

Also, if we let transcoders running on each proxy to use the same decoded video and encode it to multiple videos with different quality, the totally consumed computation power at the proxy will be less than they use decoded videos with different quality. So, this algorithm uses as small number of proxies as possible to output videos with quality vectors requested by all mobile nodes. Since this problem is combinatory optimization problem, the algorithm uses the following heuristics to simplify the calculation.

1. sort the set of proxies $P$ in decreasing order of their available computation powers. Let $SP = (sp_1, ..., sp_{NP})$ denote the sorted list.

2. sort the set of quality requirements from mobile nodes in increasing order of their required computation power (the required computation power for $q$ is given by $r_{\text{encode}}(q)$). Let $QR = (qr_1, ..., qr_{nu})$ denote the sorted list.

3. for $sp_1$, assign as many items in $QR$ as possible, satisfying $c_{\text{avail}}(sp_1) > r_{\text{decode}}(qr_i) + \sum_{q_j \in QR} r_{\text{encode}}(q_j)$.

4. similarly, assign as many items as possible to $sp_j$ ($j \geq 2$) from the left items in $QR$ until all items in $QR$ are assigned to proxies.

5. calculate the spanning tree and adjust the maximum quality $p.q$ of each proxy $p$ using the algorithm in Sect. 4.1.

4.1.2 Network Resource Minimization Algorithm

The case that the total sum of consumed bandwidths on overlay links is minimized (i.e., $\alpha = 0$ in objective function (7)), is that the same number of transcoders as the number of quality vectors requested by mobile nodes connecting to a proxy $p$ are executed at $p$. In this case, as shown in Fig. 3(b), each proxy transcodes a video to videos with the quality vectors required by mobile nodes which connect to it. So, redundant video streams are not transmitted between proxies to deliver video with a quality vector $q$ to mobile users in different proxies. As explained in Sect. 3.3.1, each proxy receives the video steam with the highest quality (i.e., maximum picture size and framerate) in the set of quality requirements of mobile nodes. So, it can transcode the video stream to any quality in the set.

4.2 Hybrid Method

Let $NP_q$ denote the number of proxies which transcode videos to those with quality $q$. In the objective function (7), if $\alpha = 1$, then $NP_q = 1$ for all $q$, and if $\alpha = 0$, then $NP_q = |\{p | q \in E(p) \land p \in P\}|$. Let $NP_{\text{max}}$ be the maximum value of $NP_q$ in the set of all quality requirements from all mobile nodes.

The problem to minimize the objective function (7) is combinatory optimization problem. So, we use the heuristics that calculate the values of the objective function for all possible values of $NP_q$ between 1 and $NP_{\text{max}}$ and select the minimum value among them. Here, we use the same value $NP_q$ for all quality requirements from all mobile nodes.

The algorithm in Sect. 4.1 is used to construct the service delivery paths among proxies. The proposed algorithm is as follows. The Step 2 to Step 4 are repeated for each $i$ from 1 to $NP_{\text{max}}$, and the minimum
value of the objective function is selected among them as a solution.

**Step1.** For each quality vector \( q \), calculate \( NP_q = |\{ p | q \in E(p) \land p \in P \}| \). First, all quality requirements from mobile nodes are divided into multiple groups based on the technique in Sect. 3.2. Let \( N_{q,x} \) denote the number of mobile nodes requiring quality \( q \) at a proxy \( x \). Let \( PS_q \) denote the set of proxies which execute transcoders for \( q \). As items of \( PS_q \), \( i \) proxies are selected from \( P \) in decreasing order of \( N_{q,x} \) where \( x \in P \).

**Step2.** Calculate \( q_{max} \), which denotes the maximum required quality of mobile nodes at proxy \( x \). \( q_{max} \) is calculated by \( q_{max} = \max(E(x)) \).

**Step3.** Construct a steiner tree among proxies. Based on the algorithm in Sect. 4.1, a tree is spanned among proxies with overlay network \( G \) and \( q_{max} \).

**Step4.** Construct a steiner tree for each \( q \). If \( i \) is larger than 1, \( i \) proxies simultaneously transcode and deliver the same quality video to multiple mobile nodes connected to them. So, a steiner tree is constructed to span \( i \) proxies for each \( q \). Here, physical hop count is used as cost metrics.

### 4.2.1 Example

We will give an intuition in the above three algorithms with an example in Fig. 3. In the figure, \( q_{max} \) and \( q_{str} \) represent the quality which the proxy should receive from its parent proxy and the quality vector of the stream transmitted through the link, respectively.

Fig. 3 (a) is an example to which the computation power minimization algorithm has been applied. There are six mobile nodes \( u_1, \ldots, u_6 \) and they have either 150, 200, 300, or 400 as their quality requirements (here, we represent quality vectors just as integers for simplicity). In this algorithm, only one transcoding service is executed at a proxy for each quality. So, four transcoding services \( T1, T2, T3 \) and \( T4 \) are executed at proxies \( p_1, p_2, p_3 \), and \( p_4 \) respectively. For example, \( u_4 \) requires quality 300 and it is directly connected to \( p_1 \), so it can receive the video stream with quality 300. On the other hand, \( u_4 \) requests quality 200 and the transcoder for quality 200 is executed at \( p_2 \). The video stream with quality 200 is transmitted to \( u_4 \) via proxies \( p_3 \) and \( p_1 \). With this algorithm, multiple video streams may be transmitted through each overlay link.

Fig. 3 (b) is an example to which the network resource minimization algorithm has been applied. In this algorithm, each proxy executes transcoding services for mobile nodes which directly connect to the proxy. For example, since \( u_1 \) and \( u_2 \) directly connect to \( p_1 \), \( p_2 \) executes two transcoding services for their quality requirements: quality 200 and quality 400. With this algorithm, only one video stream is transmitted through each overlay link.

Our hybrid algorithm minimizes the weighted sum of consumed computation power and consumed network bandwidth represented as the objective function (7) by allowing the both situations simultaneously.

### 5. PERFORMANCE EVALUATION

In order to evaluate effectiveness of our method, we compared three algorithms in the previous section in terms of the achieved cost. The environment of the experiments is as follows: We generated network topologies with 50 proxies using locality model of GT-ITM, and used it as the overlay network. We assumed that there are sufficient computational power for proxies and sufficient available bandwidth for links between proxies, in order to compare the costs of outputs from three algorithms. In the experiment, We set the number of user nodes to 2000. We determined physical hop count of each overlay link with a uniform random number between 1 and 10. We set \( \tau_d = 0.00057 \), \( \tau_c = 5 \times \tau_d \). Quality requirements of the user nodes are generated by uniform random numbers between 80 \( \times \) 60 pixel, 5 fps and 640 \( \times \) 480 pixel, 30 fps. These are grouped with 20% of permissible difference range. We have measured total costs when \( \alpha \) is changed from 0.0 to 1.0. The results are shown in Fig. 4.

Fig. 4 shows that the hybrid algorithm achieve better cost than other two algorithms when \( \alpha \) is close to 0.4. The computation power minimization algorithm and the network resource minimization algorithm achieves the minimum total costs when \( \alpha = 1.0 \) and \( \alpha = 0.0 \), respectively.

We also measured the performance of the algorithms when
the number of user nodes increases. In this experiment, we measured the computation time to generate service delivery paths with 100 to 3000 user nodes. We executed the algorithms on a PC with Athlon 64 3400+ and 1GB RAM. The results are shown in Table 1.

Table 1: Time to complete path generation (in seconds)

<table>
<thead>
<tr>
<th>number of user nodes</th>
<th>100</th>
<th>500</th>
<th>1000</th>
<th>2000</th>
<th>3000</th>
</tr>
</thead>
<tbody>
<tr>
<td>computation power minimization algorithm</td>
<td>0.016</td>
<td>0.12</td>
<td>0.43</td>
<td>1.68</td>
<td>3.91</td>
</tr>
<tr>
<td>network resource minimization algorithm</td>
<td>0.023</td>
<td>0.13</td>
<td>0.43</td>
<td>1.67</td>
<td>3.91</td>
</tr>
<tr>
<td>hybrid algorithm</td>
<td>0.076</td>
<td>1.34</td>
<td>4.18</td>
<td>9.99</td>
<td>18.7</td>
</tr>
</tbody>
</table>

Table 1 shows that the computation power minimization algorithm and the network resource minimization algorithm take almost the same time to complete path generation. The hybrid algorithm takes longer execution time, but the time is practical enough while the number of user nodes is less than 3000.

6. CONCLUSION

In this paper, we proposed a service composition based method and algorithms to calculate resource efficient service delivery paths for video multicast to multiple wireless mobile users with different quality requirements. The main contributions of our proposed method are as follows: (1) User’s benefit: our method allows heterogeneous mobile users to seamlessly receive and play back video with the required quality which can be dynamically determined based on resource constraints of their mobile terminals such as battery amount, computation power and available network bandwidth, even while they are moving; and (2) Service provider’s benefit: service providers can minimize the required resources for the video delivery and limit the resources by giving a dedicated overlay network consisting of a video server, proxies and wireless access points and overlay links among them where only the given bandwidth of each overlay link and the given computation power at proxies are consumed. Through experiments with simulations, we confirmed that our hybrid algorithm can calculate a good approximation of a tradeoff between the consumed network bandwidth and computation power with reasonable computation time.

As we showed in the previous section, our hybrid algorithm generated slightly better solutions than simpler algorithm. This is because the algorithm searches only a part of the whole solution space (the whole cost computation is done only $NP_{max}$ times). So, by extending the search space, the solution should be improved. In that case, the computation time of the algorithm will be much larger since it is the centralized algorithm. As future work, we plan to develop a decentralized algorithm to make our method more scalable.

7. REFERENCES

[9] Bhaskaran Raman, and Randy H. Katz, ”Load balancing and stability issues in algorithms for service


Digital Media and Entertainment Service Delivery Platform
Christopher J Pavlovski and Quentin Staes-Polet
IBM
Global Services
Sydney, NSW, Australia
+61 132 426
chripavl@au.ibm.com, quentins@sg.ibm.com

ABSTRACT
The emergence of broadband networks, for mobile and fixed environments, has stimulated the multimedia market for the delivery of enriched digital media and entertainment services. A key problem for institutions attempting to capitalize on these new channels for service delivery is a capability to deploy many multimedia services rapidly and cost effectively. The naïve technique is to deploy such solutions independently as so-called point solutions. The strategic approach is the development of an environment that acts as a service delivery platform for a range of digital media and entertainment services. We present our reference architecture that enables the delivery of multiple digital media and entertainment services in fixed and mobile broadband networked environments. This may be for a traditional network operator or a virtual network operator. A fundamental characteristic of our architecture is the capability to deliver multiple services with observed reductions in elapsed effort to bring these services online; with a concomitant reduction in cost and speed to market. The architecture is based upon several experiences globally, is refined further here, and presented as a blueprint.

Categories and Subject Descriptors
H.5.1 [Information Systems]: Multimedia Information Systems - Audio input/output, Video, Hypertext navigation and maps.

General Terms: Design.

Keywords: Digital Media, Reference Architecture, Service Delivery Platform, IP Multimedia Systems, Web Service Gateway, Tripleplay.

1. INTRODUCTION
Broadband networks continue to expand their reach, providing both the fixed and mobile user with access to high bandwidth intensive applications. Market research continues to predict growth trends in all broadband segments, with distinct activity in mobility. A key factor contributing to the increased network demand for bandwidth is the delivery of multimedia services such as video, music, peer-to-peer, and more recently (wideband) voice. In particular, consumers are interested in downloading digital media content in order to subsequently use this content anywhere at a time of their choosing. Due to competitive pressures, commercial entities function under constraints that require consideration of time-to-market, the cost of service introduction, and a capability to trial new services rapidly and inexpensively. Traditional approaches to deploying new services typically involve deployment of a solution that attempts to solve a discrete problem, such as a video streaming service. Often, due to prevailing business constraints, such solutions are deployed as point solutions, specifically developed to address the identified need or problem. A good example of such a constraint is the need to integrate with existing billing and settlement systems, which are often too costly to replace, in order to give the appearance of a seamless integrated set of products and services for access and subsequent charging.

An alternative approach to the point-style deployment is the construction of a platform that hosts a number of applications. Under this scenario each application is responsible for providing a discrete, but related, set of services or content. For example, one application is responsible for video streaming service, another for multimedia news clips, and others may provide entertainment products or services such as online and offline games. The platform is tasked with the responsibility of providing a set of digital media services which are common to such applications, whereas the applications extend this inherent capability.

This approach to service delivery forms the basis of our architecture. We base our design on our experiences deploying digital media and entertainment services, which preserve the architectural principles of a platform approach to service delivery. Hence we present a proven blueprint for service design. The reference architecture caters for third generation mobile, fixed wireline broadband, and wireless broadband networks.

2. RELATED WORK
The related work has dealt with multimedia solution components in a discrete way or may focus on the telecommunications network in service delivery. We present a collaborative architecture that integrates the various components, in a manner that provides seamless interaction with digital media and entertainment services. In particular, we consider further the needs of the existing IT environment of the institution deploying the platform.

In [1] a mobile internet platform is presented that outlines an environment to deliver multimedia content and services to the mobile phone user. The platform is capable of delivering content and services in second generation mobile networks. However this does not address the broader domain of service delivery in both fixed and 3G mobile broadband networks, virtual network operator (VNO) support, and peer-to-peer services.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSC’05, November 11, 2005, Singapore.
Copyright 2005 ACM 1-59593-245-3/05/0011…$5.00.
An alternative framework is proposed for the creation and execution and management of multimedia services [2], however this is based upon a prototype implementation with focus on the mapping of SIP functionality to Parlay services.

The objective of the Parlay group is the definition of network application programming interfaces (APIs) to support creation of telecommunications services [3]. The underlying Parlay standard defines a range of network oriented functions such as call control, messaging, and mobility services [4]. The Parlay X APIs are a subset of the Parlay APIs and are defined as Web Services [5] in order to support the broader population of developers who may extend this network capability to provide enhanced services. The differentiating aspect of Parlay X is simplicity over Parlay [6]. Parlay has an intrinsic reliance on network capabilities; therefore developers wishing to make use of additional features of the prevailing IT environment must further integrate and develop these functions. This may include user registration and profile management, billing and bundling of products and services, and service creation and management.

The IP Multimedia Subsystem (IMS) is a comprehensive standard and architecture developed by 3GPP consortium for the delivery of multimedia services over traditional networks [7]. IMS focuses upon the telecommunications network treating the circuit switched and IP networks as discrete networks [8]. The IMS architecture deals more with network architecture, presenting an approach to abstract the telecommunications network to deliver multimedia services. The IMS extends the range of supported value-added multimedia services beyond Parlay, such as peer-to-peer, video streaming, and SIP services.

In [9, 10] a service delivery platform, that extends IP Multimedia systems, has been developed. The authors outline a test-bed architecture that integrates SIP and Parlay with the telecommunications network to deliver multimedia services. A further platform for delivery of mobile games over the IMS has also been described [11].

Several performance and quality of service (QoS) approaches for multimedia service composition are outlined [16, 17, 18]. In our architecture we assume existence of such underlying frameworks and refer to these works for approaches in QOS and performance optimization techniques.

In this paper we draw our attention to the additional requirements presented by the prevailing IT environment. In particular, we accommodate the needs of existing business processes, associated with legacy IT systems, of the operator and the VNO (such as mobile/fixed game service providers and content service providers) in multimedia service delivery. This enables additional capabilities such as bundled financial models for services using post and pre-paid payment instruments, revenue settlement with service providers and VNOs, and consumer and service provider relationship that support service access and management.

### 2.1 Contribution

In this paper, we present an architecture that delivers digital media and entertainment services in both a mobile and fixed broadband environment. The consumer of these digital media services, has the capability to roam between mobile networks, and is able to utilize additional modes of user interaction that are bestowed with third generation mobile networks (such as multical supplementary service). This work is based upon our experience deploying several multimedia service delivery platforms globally. Hence we provide a reference point for the deployment of multimedia services, whilst preserving principals in business model flexibility, application implementation speed, and user consistency. We view the key contributions as the following:

- we outline a reference architecture for digital media and entertainment service delivery that integrates with existing IT environments for mobile/fixed network operators and VNOs;
- define the core web service components required to support service provider delivery of digital media applications; and
- specify an architecture that combined services of telephony, data, and video, (referred to as tripeplay in the industry).

The fundamental notion for such an architecture is now explored further where we discuss further the benefits of this deployment approach.

### 3. PHILOSOPHY OF SERVICE DESIGN

There are several capabilities necessary for developing new multimedia applications. This includes presentation services, security services such as authentication and authorization, registration and subscription; and integration with established IT systems (i.e. billing) and network channels. The philosophy of our approach is based upon the notion that a portion of the software logic that is developed for one multimedia application is common to a range of multimedia applications. Furthermore, the unique business logic of the new service builds upon a set of cumulative common services. The following diagram illustrates this general notion (see Figure 1) where each capability layer often builds upon the functions available in lower layers.

![Figure 1. Capability Layers of a Developed Service](image)

The Portal services typically include registration, provisioning, and subscription functions for consumer service access and activation. In addition, self care and reporting functions are also available to the consumer and service provider. The infrastructure layer typically includes security services such as authentication and authorization, auditing, error management, database repository, and usage tracking for billing and reporting purposes. The network and system integration layer comprise the effort associated with integrating with billing systems, core user registries, settlement systems, and the network elements and channels. We reviewed several digital media and entertainment implementations and observed general trends in the level of effort attributed to each of these areas when deploying a new service. The table below summarizes these observations, see Table 1.
The eco-system for service delivery is composed of several key entities, refer to Figure 3, these are the:

- digital media and entertainment service delivery platform;
- integration with existing IT systems (e.g. financial, reporting, and settlement);
- integration with content and service providers; and
- telecommunication network and gateway channel integration for consumer and service provider access.

Consumers are the customers (or subscribers) who access the platform and gain admission to the various externally hosted content and services delivered by the Digital Media and Entertainment (DME) platform. Several devices may be used by these users including mobile phone, PDA, laptop over wireless networks, personal computers over fixed line broadband, and televisions equipped with set-top units or gaming consoles. Consumers also interact with the DME platform to perform registration, personalization, and interactive media functions.

Service providers represent the external 3rd parties (also referred to as merchants, service owners, and content developers) responsible for developing and managing externally hosted applications that deliver content, services and games to the consumer base. The service providers require access to the DME platform to conduct functions such as requesting and viewing reports, performing self care activities, and registering new services (i.e. games, content, and digital media).

Platform administrators include customer service representatives, administrators, and reporting analysts. Customer service representatives maintain the system, manage disputes with external service providers, and assist in customer satisfaction issues. Administrators perform tasks associated with the upkeep of the DME platform.

<table>
<thead>
<tr>
<th>Domain</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business logic of service</td>
<td>~30%</td>
</tr>
<tr>
<td>Portal Services</td>
<td>10-15%</td>
</tr>
<tr>
<td>Infrastructure Support Functions</td>
<td>15-20%</td>
</tr>
<tr>
<td>Network &amp; Systems Integration</td>
<td>30-40%</td>
</tr>
</tbody>
</table>

With distribution of deployment effort in mind we elaborate upon an underlying principal of the service delivery platform approach, see Figure 2. This notion is not new and although not explicitly stated is evident in the various contributions discussed in section 2.0.

Figure 2. Leveraging common functions in service delivery

The above diagram illustrates that multiple applications may be developed which provide digital media and entertainment services. These services extend a core set of capability that is made available as Web Services by the service delivery platform. Applications are accessed directly by consumers, however the application leverages the capability of the SDP in order to fulfill the service requested by the customer. Later in section 5.0 we provide a concrete example of how such an interaction occurs.

4. Reference Architecture for Service Delivery

In this section we outline the specifics of the reference architecture that enables delivery of broadband content, such as digital media and entertainment, to consumers. The reference architecture may be deployed by institutions such as telecommunications operators or VNO’s. This architecture is based upon several implemented solutions and is inclusive of those components most common to production solutions deployed. As such the intention is to articulate an architecture that may be extended further with additional functions and capabilities. This may be viewed as a proven blueprint that may be used as reference architecture.

4.1 System Context

The eco-system for service delivery is composed of several key entities, refer to Figure 3, these are the:

- recall the service design philosophy for aggregation, the DME environment hosts the core logic of the platform that provides the portal services, infrastructure support functions, network integration, and IT systems interfaces. The following diagram depicts the associated physical topology, see Figure 4.
The physical topology comprises several key sub-systems, these are the:

- network channel nodes;
- master authentication server;
- multcall session manager;
- web service gateway;
- SIP service manager;
- portals (Consumer, VNO, SRM, Admin);
- broadband management engine; and
- integration of legacy IT systems.

The network channel nodes consist of those elements that interface directly with the telecommunications network. This includes the Parlay gateway, messaging gateways (to SMI/SMI), WAP gateways (to GGSN/SGSN/PDSN), and the location and telephony servers. We now elaborate further on each of the remaining sub-systems listed above which directly comprise the DME platform architecture. In section 5.0 we further describe a detailed scenario of how these entities collaborate in the delivery of a multimedia service.

4.3 Master Authentication Server
The Master Authentication Server is the consolidated access point for all external entities accessing the services made available by the DME platform. Several functions are performed by this component, this includes authentication of users, authorization of service requests, and single sign on across a range of current and future network channels (2G/3G, wireless broadband such as WiMax, and fixed broadband Internet).

As the central security node, this component performs the authentication of consumers accessing the DME platform and of external multimedia applications that make use of the published web services. For scalability, the master authentication server may also be partitioned so that dedicated nodes authenticate SIP requests, web service requests, and requests from consumer devices. Following authentication, consumers interact with the consumer portal to select the desired multimedia product or service. During this selection, consumer authorization is confirmed, to confirm access to the service, before being redirected to the multimedia application. Since services are provided by external applications, a trusted relationship must be instituted between the platform and application to allow access by the consumer. This is accomplished by employing a federated security identity model.

4.3.1 Federated Identity Management
Federated identity management, developed by the liberty alliance project [12], specifies how independent organizations are able to share user identities for trusted access to applications. The security assertion markup language (SAML) has been defined by the Organization for the Advancement of Structured Information Standards (OASIS) [13]. The SAML specifications define several types of assertion, including identity authentication, attribute authentication, and user authorization. As such, a SAML authority, trusted by external applications, is required that prepares SAML (redirection) requests when the user selects a service. This is carried out by the service redirector.

4.3.2 Service Redirector
The service redirector is fundamentally a mapping registrar, where the selected service is translated into an external URL representing...
the application responsible for providing the digital media or entertainment service. In constructing the redirection, additional information is supplied within the URL request so that the receiving application is able to verify the identity of the consumer and that the consumer is authorized to access the service. Service redirection is conducted at a coarse grain level, meaning that if the external application wishes to enforce additional constraints, to access content provided by the application, then these checks are performed locally. Generally, such authorization checks are used to determine if the user has subscribed to access and pay for the service being requested.

4.4 Multicall Session Manager
In addition traditional session management, the multicall session manager performs two key functions within a multiple broadband network infrastructure. This includes the seamless transfer of a session when moving between networks and the management of multiple concurrent connections to a mobile device in a 3G mobile network for multimodal interaction.

4.4.1 Multimodal Interaction Support
Multimodal applications are bestowed with the capability to manage several modes of user interface, most typically voice and web [14]. Such applications are considered more versatile due to the flexibility of using combined modes of user interaction.

The multicall supplementary service provides a capability to establish multiple transmission channels to the mobile handset [15]. This service also permits the transmission of voice and data simultaneously over 3G networks, providing support for multimodal applications. This is achieved by establishing both a voice (circuit switched) and data (packet switched) connection between a mobile phone and an application, allowing a user to interact with an application with either voice or data commands, or both simultaneously. In order to support simultaneous connections to the mobile phone, the multicall session manager provides $M \times N$ session management capability (i.e. $M$ user session by $N$ connections per user).

4.4.2 Roaming Agent
As the various networks increase their coverage and users become increasingly more mobile, greater demand will exist to remain connected whilst traversing several types of networks. The roaming agent is responsible for allowing seamless network roaming, thus an application connection established over a 3G network will not be disrupted when moving to a wireless hotspot. This roaming agent must be present on the mobile device, while the multicall Session Manager contains the server logic for this connection. For example, a PDA with the agent installed establishes a secondary IP connection stack that overlays an initial IP stack created over a 3G network. The mobile device must also be equipped with dual network interfaces; such devices are now readily available. When a WiFi network is detected by the device, a new lower level IP connection is established and the agent redirects transmission over the alternate network. Since the secondary IP stack remains unchanged during this process, seamless roaming is guaranteed.

4.5 Web Service Gateway
The Web Service Gateway (WSG) is responsible for providing services through published interfaces to authorized multimedia applications. The key observation is that these web services are used by the applications and indirectly used by consumers as they interact with an application. The multimedia application may also be charged for use of the web service, alternatively the application may be an in-house requestor. The WSG collaborates with three primary entities in order to fulfill a service request. This includes the various network services through the channel gateway nodes, the legacy IT enterprise systems, and the core broadband engine.

The WSG also serves as an ecosystem for developers, by providing an environment to rapidly test and trial new digital media and entertainment services. A sample of the set of web service typically offered is shown below in Table 2.

<table>
<thead>
<tr>
<th>Web Service</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetDevice</td>
<td>Return further device information.</td>
</tr>
<tr>
<td>CheckBalance</td>
<td>Check prepaid account balance.</td>
</tr>
<tr>
<td>ServiceCharge</td>
<td>Charge consumer (postpaid).</td>
</tr>
<tr>
<td>GetLocation</td>
<td>Retrieve location of device.</td>
</tr>
<tr>
<td>SendMultimedia</td>
<td>Send message via network.</td>
</tr>
<tr>
<td>UploadGame</td>
<td>Load game to DME platform.</td>
</tr>
<tr>
<td>Authenticate</td>
<td>Authenticate consumer identity.</td>
</tr>
<tr>
<td>IsAuthorized</td>
<td>Is consumer authorized to access service?</td>
</tr>
<tr>
<td>UploadStream</td>
<td>Load streaming content to DME platform.</td>
</tr>
</tbody>
</table>

Another key feature of the WSG is the ability for multimedia applications to combine various web service functions for flexibility. Using an example, a theme service that sets ring-tone, caller tune, wallpaper, is made available to post paid subscribers, and may be purchased half-price for the duration of a movie launch. This example uses a combination of several network, legacy, and platform features, demonstrating greater flexibility for application developers. Such as network location, messaging, and billing.

4.6 SIP Service Manager
The Session Initiation Protocol (SIP) is an application layer protocol that manages the establishment, control, and termination of multimedia sessions. The emerging and popular application in use today is Internet telephony, or rather Voice over IP (VoIP). Additionally, SIP supports the capability for users to maintain a single identifier regardless of their network location.

The SIP gateway supports the use of key multimedia applications such as VoIP, video conferencing, and other peer-to-peer multimedia services. The SIP gateway provides a set of core services, such as name mapping and redirection, to enable connections between external applications. For instance, it is possible to attach a video recorder to a TV to watch a personal recording of some family event. The consumer may then decide to dial in other family members (who view the same digital recording remotely) to share the viewing experience. The SIP service gateway functions in a similar way to the WSG, with the exception...
that the external multimedia application is actually a client application invoked by the consumer.

Peer-to-peer services are a comparatively new paradigm, and are an important constituent of IP telephony as a tripleplay offering. The example above also briefly illustrates the potential for consumers to also act as content providers.

4.7 Portal Services
The term portal is used to signify the set website pages, accessible from a mobile phone or personal computer (i.e. WML or HTML), that provides a set of functions to consumers. The portal provides an interface by which the users of the platform access the multimedia services and the platform functions available. Given the diverse needs of the consumer and the service provider, i.e. one is the purchaser of the multimedia product or service and the service provider the vendor, dedicated portals are required to support the needs of each of these user communities. The internal operational and maintenance functions to manage the platform are performed with the administrative portal. Each of these portals are now described in further detail.

4.7.1 Consumer Portal
There are essential two instantiations of the consumer portal, one for mobile (WAP) access and a second for Web access from fixed devices. In both cases, there is a content portal and a self-care portal. The Web portal is more comprehensive, offering additional self-care capabilities. The WAP portal has general restrictions due to the form factor of mobile devices.

The consumer content portal is the means by which all consumers access the multimedia services. This predominantly includes a data (Web/WAP) interface, accessible by mobile or fixed device, but also may include a voice portal for multimodal support. Whilst the portal is device independent, certain function are sometimes only made available via the web portal versus the mobile portal, such as IP TV. The key features of the consumer portal include consumer registration functions, to enable new users to register to the platform, personalization and profile maintenance, and content or service catalogue menus.

The service catalogue provides the ability for consumers to navigate a menu for selecting the desired service or product. Each product or service is supplied by a corresponding multimedia application. Whether the application responsible for the content or service is situated locally with the platform or externally, is indistinguishable to the consumer.

4.7.2 Service Relationship Management Portal
The Service Relationship Management (SRM) portal provides the set of functions that enable an external service provider to register a new multimedia application that provides digital content, entertainment services, or a product. It is important to observe that the platform is focused on maintaining information at a coarse level with regards to external multimedia applications. Each external application is therefore responsible for maintaining the specific range of content available at a more granular level. For example, a service provider would register a polyphonic ring-tone application with the DME platform, rather than register each ring-tone that the application provides. Hence the application is synonymous with the general notion of a service, in this case providing ring-tones.

The SRM portal provides other functions such as service usage reporting, payments due to the service provider, and profile management (self-care) activities. This enables the service provider to supervise their portfolio and decide where new services may be introduced or to deprecate ones which are no longer actively used or financially cost effective. Note that when new services are registered and introduced an approval or publish process is initiated. This is carried out by the administrators of the platform.

4.7.3 Virtual Network Operator Portal
The VNO portal is essentially a white-label portal, meaning that it possesses no branding, that may be adopted and branded by an external business wishing to function as a virtual network operator. The VNO then on-sells telecommunication enhanced multimedia services. This business model is recently gaining significant momentum within several industries attempting to leverage growth in mobile and fixed broadband multimedia applications such as peer-to-peer podcasting, mobile blogging, and on-line gaming. This includes consumer electronics, game service providers (GSP), Mobile Virtual Network Operators (MVNO), and traditional Application Services Providers (ASP).

The federated security model is crucial to enabling this relationship to function, allowing the customers of the VNO to access the portal. Whilst the portal is illustrated as being deployed within the platform, this may be physically situated within the VNO site, whilst the remaining DME SDP components remain deployed to the telecommunications operator site. This VNO approach is generally suited to smaller institutions.

Alternatively, the entire platform may be deployed to the VNO. In this deployment mode, dual web service gateways exist: a WSG that resides with the platform at the VNO site, which is directly accessed by the multimedia applications, and a secondary WSG that is situated at the operator site. In effect the first WSG is a proxy for the second gateway. From our experience this is only practical for large organizations that have account management systems in place.

4.7.4 Administration Portal
The administration portal is the means by which the operator, or VNO, manages the platform. This contains the functions to review and publish new multimedia applications submitted to the platform, resolve disputes between consumers and service providers, and perform operational maintenance and support activities.

4.8 Broadband Engine
The broadband management engine is supported by several components that deliver and charge for broadband content. This includes a download manager and online gaming service, video streaming service, revenue collections, and digital rights management. The underlying architectural framework fosters extensibility, such that additional components may be added to the engine to support further capabilities as they arise.

4.8.1 Repositories
The key repositories include the content store, service registry, consumer profile, and service relationship management (SRM) profile.
In general, multimedia content will be supplied by external service providers to consumers. However, content may also be provided (and on-sold to 3rd parties for reproduction) by the operator or VNO. The content store is accessible by multimedia applications via the WSG. An example is a location service where the application obtains mapping information from the content store provided by the DME platform; this is information is retrieved in addition to location details.

The service registry contains details of all multimedia applications registered with the platform. This is reproduced as a service menu (or catalogue) which the consumer browses. When a service is selected a SAML redirection takes place as mentioned previously.

The Consumer and SRM profile contain secured details regarding registered consumers and service providers respectively. Both communities require personalization, reporting, and audit capabilities to manage and resolve disputes. This is particularly critical for service providers to operate their business effectively.

4.8.2 Streaming Server
The streaming server is responsible for streaming content to the mobile or fixed device. Due to high bandwidth loads that streaming content incurs, this would normally be performed from within the firewall enclosure, (dotted link to MAS shown in Figure 4). This is also a consistent requirement for other services to ensure that response time and performance is not compromised. While an external multimedia application may perform this function, the DME platform is able to provide this as a common service that may be leveraged by external applications. Hence applications need only pre-load their content for later streaming.

The streaming server also supports IPTV, a key tripleplay offering. The IPTV service is provided in the form of multicasting, using the internet group management protocol, and video on demand with conventional real time streaming protocol.

4.8.3 Gaming Server
There are two aspects that require consideration in support of the gaming server. This includes a download capability for mobile devices and an on-line gaming arena. That later appears to be increasingly important and is perhaps strategic for VNOs wishing to establish a consumer household presence using the living room television as the portal.

The download manager generally supports the JSR-190 specification, which standardizes the events/transactions between the MIDlet on handset and server. In addition, this component should support a wide variety of billing and pricing models. This includes usage or time limit, trial and subscription, and payment methods such as prepaid and postpaid.

The gaming arena includes several functions including waiting room (lobby server), tournament management, multicast for on-line gaming, and ancillary services to support gaming communities such as chat, high-score, and messaging.

4.8.4 Revenue Collection
The revenue collection engine is responsible for aggregating and dispatching all chargeable events related to services rendered and used. This component provides the point of integration between legacy IT systems responsible for account based billing, financial networks, and settlement systems. Revenue settlement is the process by which revenue is collected by the operator or VNO and is distributed to external service providers under the financial model agreed upon. Clear audit trails and accounting practices are required for dispute resolution. In addition, the user is able to designate the desired payment instrument consisting of payment card (i.e. credit or debit), post paid account, or pre-paid accounts.

4.8.5 Digital Rights Management
Digital rights management (DRM) is becoming more widespread however this is largely dependant upon mobile device capability. The three classes of rights management are forward lock, separate delivery, and combined delivery. The DRM component provides the external multimedia application with an optional capability to enforce one of these rights management schemes, which may be specified when the relevant web service is invoked to deliver content.

4.9 Legacy IT Systems
While not formally part of the DME service delivery platform, the legacy systems provide a set of capabilities that are made available by the platform. Specifically, these functions are advertised to multimedia applications as part of the available web services.

An integration middleware layer acts as the convergent point-of-control for systems integration. Such technologies are generally incumbent elements of an enterprise IT infrastructure. We now briefly outline the key legacy IT systems required to support the functions of the DME platform.

Post-paid. The account based billing system is used to confirm post-paid customer details during consumer registration and potentially for notifications of account suspension or account closure. All recurring and non-recurring charges are sent to the post-paid billing system. The revenue collection engine forwards mediated charges, generally as daily batch, for processing.

Customer Repository. The central customer repository is the main user database managed by the platform owner. This system will be interrogated during the consumer registration process to confirm the identity of new users. Note that in the case that a central customer repository does not exist the post-paid or pre-paid billing systems may be interrogated to verify the customer identity during registration.

Pre-paid. The pre-paid system is used for account checking during payment authorization, payment capture (i.e. debit), and refund processing. This may also be used to verify pre-paid customer registrations, in the case that no central customer repository is available to perform this function. Pre-paid systems are traditionally deployed as part of the intelligent network (IN) however they are also becoming available as an IT systems implementation.

Financials Systems. Financial systems are those accounts payable and receivable systems used to invoice consumers and to manage distribution of settlement funds, or refunds, to external service providers.

5. Web Service Interaction
To illustrate how the platform delivers multimedia services to consumers we now present a use case scenario as an interaction diagram. In this use case, a pre-paid mobile phone customer requests multimedia content from an external service provider, where billing for the service occurs at the point of access. The
following diagram illustrates the sequence of events that occur when interacting with the DME platform, see Figure 5.

Briefly, the pre-paid customer logs into the DME platform, selects an external 3rd party service that provides multimedia content, the external service subsequently delivers the content via MMS and/or SMS. Please refer to the solution components of Figure 4, for interacting entities.

Figure 5. Web Service Interaction

The first action occurs when the customer switches on the mobile phone and is authenticated by the mobile (GSM or CDMA) network. The customer browses the portal site, using a 2.5/3G data service and is automatically authenticated by the GGSN and associated radius authentication server. The WAP request is then forwarded onto the WAP gateway for conversion to an HTTP request. The HTTP request is forwarded onto the Master Authentication server, which is configured to operate with the WAP gateway and GGSN in trusted mode. This means that the user is not prompted for a username password, since this authentication has already taken place. The Master Authentication server checks if the customer has previously logged in. If no session is present a new one is created. The MSISDN (or MDN for CDMA networks) is used to lookup the customer details to populate the session stored with a temporary identifier.

In step two, the Master Authentication server maintains a cookie on behalf of the handset (using a cookie proxy usually as a plug-in extension) and forwards the request to access the platform onto the portal. At this point the user may interact with the portal (this is not shown), performing self care functions and other activities. At some point the user decides to select a content or service, this is done by browsing the menu of services provided by the multimedia applications.

The third action occurs where the mobile customer decides upon a particular service and selects this from the menu item. This results in a WAP request to the Master Authentication server. The Master Authentication Server resolves the specific URL of the external multimedia application, by looking up the database of applications, and returns a SAML redirection to the mobile device. The redirection string includes a substitute of the identity of the customer with a temporary identified or alias, this facilitates customer anonymity.

After selecting the desired service the customer is redirected to the external service provider application (step 4). The customer browses the catalog of content and services at the redirected site and selects a particular item (i.e. content, product, or service). The multimedia application returns a description of the goods selected to the customer along with the associated price, (note that this rated price may also be retrieved from the DME platform).

When the customer decides to purchase the selected product, e.g. today’s hot news supplied via SMS/MMS. The purchase request is submitted to the external application. The application now prepares a web service request using the customers’ temporary identifier and the product cost and sends this to the web service gateway to check for sufficient funds in their pre-paid account. The web service gateway forward the request onto the revenue collection engine in order to check for sufficient funds in their pre-paid account. The pre-paid legacy system returns to the revenue collection engine a result indicating if sufficient funds remain for the amount requested. The revenue engine may apply a set of additional business rules to determine if the request is to be approved, i.e. special tariffs.

In step six, approval is returned to the external multimedia application. When received, the external application invokes a web service, upon the Web Service Gateway, to deliver the content via SMS/MMS; the content is included within the web service request. The Web Service Gateway generates a charge record, so that charging of web service usage may be performed if required by owner of the DME platform, and then invokes the relevant core network element interface. In this case either an SMS or MMS request, resulting in an SMPP message to the SMSc, or an HTTP message for an MMS request. On receipt of the request, the SMS/MMS gateway returns an acknowledgement to the Web Service Gateway, which in turn forwards a similar response confirming dispatch back to the multimedia application.

The external application then issues a capture request to revenue collections. Revenue collection issues a fund capture against the back-end pre-paid system in real-time. An acknowledgement is returned, which is then forwarded onto the external application via the web service gateway.

At step eight, the customer is notified of the purchase confirmation and that funds have been deducted from their pre-paid account. Step nine shows that at a later point the SMSc (MMSc) then fulfills the content request by sending the message to the customer device. The SMSc also generates a delivery record on completion.

A final action is performed (step 10) when records of delivery, or rather delivery receipts, are retrieved in batch and sent to the revenue collection engine. A batch process is then run to reconcile delivery receipts with the charge detail records generated by the revenue collection process. This may be used for dispute resolution, and if appropriate to re-instate deducted funds to the customer for undelivered content.

This fundamental approach is used to interact with other services such as games, video streaming, and location based services. The customer is redirected to the relevant service application at step 4. Furthermore, the application would make the relevant web service calls for addition information such as location and billing. For example, in the case of video streaming, the customer is first
We present an architecture based upon our experiences deploying digital media and entertainment service delivery platforms. A fundamental feature of the platform is the use of web services for enabling external multimedia applications. The platform also caters for recent trends to support triple play (telephony, data, and video) services. Given the dynamic nature of service evolution, we observe that constant attention and refinement is required in the needs of the consumer, service provider, and service delivery platform owner.

6. Summary and Conclusions

Whilst it is possible to deploy a service delivery platform that employs the Parlay or IMS architecture, this does not cater for the broader needs of the VNO. Furthermore, we have observed that the operator is still required to develop those IT capabilities for each application for management of consumers and services providers, to establish the financial models in those relationships, and to integrate with existing business processes encapsulated within existing IT systems of the organization.

We present an architecture based upon our experiences deploying digital media and entertainment service delivery platforms globally. The architecture may be viewed as a blueprint. A fundamental feature of the platform is the use of web services for enabling external multimedia applications. The platform also caters for recent trends to support triple play (telephony, data, and video) services. Given the dynamic nature of service evolution, we observe that constant attention and refinement is required in such architectures so that platforms are adapted appropriately to the needs of the consumer, service provider, and service delivery platform owner.

7. ACKNOWLEDGMENTS

We thank Neil Cherry and Doug Regan for sponsoring this study.

8. REFERENCES


Supporting Meetings with a Goal-Driven Service-Oriented Multimedia Environment

Karin Anna Hummel, Wolfgang Jochum, Stefan Leitich, Bernhard Schandl

University of Vienna, Faculty of Computer Science
Department of Distributed and Multimedia Systems
Liebiggasse 4/3-4, A-1010 Vienna, Austria

{karin.hummel,wolfgang.jochum,stefan.leitich,bernhard.schandl}@univie.ac.at

ABSTRACT

During the last decade the number of personal multimedia-enabled devices has increased significantly in everyday usage. Additionally, high-end devices for multimedia environments enable multi-modal human computer interaction and in particular advanced collaboration. Most multimedia environments focus on efficient provisioning of multimedia services but show a lack of user-centric aspects, like explicit reasoning about the users' wishes and fulfillment of requirements.

In this paper, we present a novel goal-driven approach for multimedia service composition which is able to reason about the users' demands and to adapt to new context situations in a non-intrusive manner. The fulfillment of goals is assured by hierarchical goal structuring, service provisioning, and evaluation of service fulfillment degrees. We apply this approach to a typical multimedia-enriched meeting scenario described by means of Semantic Web ontologies. A flexible and modular service-oriented software architecture demonstrates the usability of the envisioned companion-like smart meeting room.

Categories and Subject Descriptors
H.1.2 [Information Systems]: Models and Principles—User/Machine Systems; H.5.1 [Information Systems]: Information Interfaces and Presentation (e.g., HCI)—Multimedia Information Systems; I.2.4 [Computing Methodologies]: Artificial Intelligence—Knowledge Representation Formalisms and Methods

General Terms
Design, Human Factors

Keywords
Goal-Driven Architecture, Semantic Web, Service-Oriented Architecture

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MSC'05 November 11, 2005, Singapore
Copyright 2005 ACM 1-59593-245-3/05/0011 ...$5.00.

1. INTRODUCTION

In the field of user interactive systems, five major trends have emerged during the last few years: (1) increasing pervasiveness of computing power and sensors, (2) the user-orientation implemented by services and configurable devices, (3) the increasing digitalization of business workflows, such as meeting situations, (4) the recognition of semantics and semantically-enriched data and process models as an enabling factor for interoperable systems, and (5) the emergence of self-adapting and learning systems.

Due to the complexity of computing and hardly manageable direct interactions, the approach of goal-driven user interaction will – in our belief – gain more and more importance. To increase system acceptance, users should only specify what they want to do while the complexity of how to operate the service should be hidden to a high degree. In our vision, we substitute the simple interactive selection of application functions by an omnipresent, but non-intrusive and non-disturbing assistant for multimedia environments.

We propose a system design where three types of knowledge are used to support user requirements: (1) general knowledge about life situations, such as a meeting, (2) situation-related knowledge (e.g., the agenda of a particular meeting), and (3) live knowledge, i.e. data gathered by sensors, external events, and user interaction, including the data usually denoted as context. The sources of these knowledge types and some examples are depicted in Figure 1.

Figure 1: Types of knowledge

Based on these knowledge types, we propose a system to integrate services, devices, and users in a flexible and adaptive manner. In order to achieve adaptability of the system, means for modeling the success of operations are required. Therefore, we propose a goal-driven system archi-
tecture which allows to model workflows as goal hierarchies. Services are used to fulfill goals which cannot be decomposed any further. A matching algorithm selects the service capable of fulfilling the particular goal in a given context best. This separation between complex goal hierarchies and simple services exhibits some major advantages. First, the approach allows to model situations where different services are capable of fulfilling a specific goal, which makes the approach robust. Services can easily be added without additional system changes, which makes our approach scalable. Similarly, services do not have to be altered in case new goals are introduced. Finally, since services are evaluated depending on the given context, the system further becomes context-aware without changing its purpose expressed by the goal hierarchy.

For demonstration purpose, we apply the approach to a goal-driven, service-oriented smart multimedia environment. We choose the meeting application domain because it is an ideal prototype and demonstration object for various interesting aspects. For example, in a meeting, persons with different organizational backgrounds collaborate for a foreseeable period of time. The meeting participants often bring in their own personal devices, which are probably unknown to the system. The semantic models of meetings (i.e. data objects, processes, artifacts, etc.) are well-known and can be reliably expressed and modeled. Furthermore, the concept of a goal is inherent in the meeting situation, since economics prohibits to hold meetings that do not produce results, or at least strongly aim to do so. The envisioned smart meeting room will be capable of tracking activities, suggesting appropriate alternatives, and automating user interactions. It will play the role of a meeting companion.

Synopsis

The remainder of this paper is organized as follows: After discussing related work in section 2, Section 3 introduces the proposed architecture of our system and its main system components. In Section 4, we present a formal model to describe goals, their attributes, and their relations. Section 5 describes how services are modeled, while Section 6 describes the process of matching between goals and services. To illustrate our concept, Section 7 presents thoughts for the application of the presented approach in specific meeting situations.

2. RELATED WORK

The ubiquitous enrichment of working and home environments with multimedia-enabled devices has been investigated in various studies. For such environments, the architecture DCOD/VDSG [9] proposes to achieve two main benefits: (1) The devices available in a multimedia environment can be controlled by the use of a single mobile device, like a PDA, and (2) services and functionalities are composed by combining the environment’s devices into virtual devices. In contrast to the user interaction driven control, we propose an approach based on the companion metaphor, which allows to hide system complexity from the user to a high degree. Bandara et al. [2] propose an ontology for the semantic description of devices. This approach addresses the physical level of device description (including hardware and software properties) and does not consider the problem of matching device descriptions to actual user or system requirements. In this work, we describe how the matching between user requirements modeled by goals and services can be supported by using extended ontologies.

The RUBI framework [11] emphasizes the similarity between resource discovery and network routing, since both fulfill the need to discover and locate services and resources available over the network. In contrast to service discovery in a highly dynamic network, we focus on a system architecture supporting service matching to a particular requirement and context. Heider and Kirste [12] present a promising approach to goal-based interaction with complex devices in the home infotainment area. They state the important point that a user is not primarily interested in functions, but in goals. Interaction with devices should be modeled in order to support this assumption. In [7], the authors present a smart meeting room based on a context broker architecture, which emphasizes the context aspect of a meeting scenario. Similar to our approach, the authors use the Semantic Web as source for key technologies. The system architecture presented in this article extends these approaches by selecting services while considering both the goals and the context.

3. ARCHITECTURE OVERVIEW

The success of a smart multimedia meeting room relies on the fulfillment of the meeting participants’ requirements in terms of services provided. Although general meeting requirements may persist, services may change due to technological progress, and new services might be needed over time. Therefore, flexibility and ease of enhancement are major quality criteria of the proposed system. In our architecture, we combine two perspectives. First, we base the architecture on general meeting workflows as described in Section 1. Second, we introduce multimedia meeting services following the approach of Service Oriented Architectures (SOAs) [15]. The services of this smart meeting room are distributed and accessible via different stationary and mobile devices. For the latter, the meeting room should provide wireless access, for example, via WLAN and Bluetooth.

The multimedia environment system consists of four main components as depicted in Figure 3: (1) the agenda preparation module, (2) the goal definition engine, (3) the runtime environment, and (4) the learning & self-adaptation module. Additionally, a repository stores and manages all data related to the system such as, for example, sensor data, items of the meeting agenda, and human computer interactions during the meeting. The object space serves as a virtually shared meeting memory, and various services provide multimedia meeting support.

3.1 Agenda Preparation

The agenda preparation module supports the person who conducts the meeting (typically the moderator) in all activities preceding the meeting. A proactive assistant helps the moderator to create and structure the agenda. We further include tasks that are not directly related to the agenda, like the invitation of meeting participants or the booking of required equipment.

The structured agenda is the source of the definition of goals. These goals can be derived from the agenda items or from combinations of such items. For instance, based on the descriptive agenda item “presentation of annual revenue numbers” we can infer the goal “presentation of annual revenue numbers is successfully finished” and related subgoals,
like “presentation is displayed on the video-wall” and “every participant owns a copy of annual revenue numbers”. Note that we model only goals that are relevant to the technical aspects and the execution support of the system, instead of modeling business goals.

The agenda is specified using vocabulary of the agenda ontology, which is a representative of the first type of knowledge (general knowledge). The agenda ontology defines a vocabulary that allows to model all relevant aspects of a meeting, including participants, topics, and additional documents. It defines concepts for several well-established meeting agenda items, like presentation, discussion, negotiation, or coffee break. The agenda ontology and corresponding individuals are defined in OWL DL\(^1\) in a flexible, extensible, and exchangeable manner.

### 3.2 Goal Definition

Using the input from the agenda preparation module, the goal definition engine defines a goal hierarchy, which includes additional information provided by the meeting initiator and lessons learned from previous meetings extracted from the repository. The goal hierarchy may directly be modified by the initiator, guided by a proactive assistant agent. Together with the agenda the goal hierarchy is used by the runtime environment as script for the actual meeting. We define a goal ontology, which provides a vocabulary to define goals and their relations. The formalism of goal modeling is described in Section 4.

### 3.3 Runtime Environment

The runtime environment supports the meeting participants during the entire meeting. Its versatile functionalities are divided into several submodules, which communicate by means of an event-based distribution system.

The agenda processing component uses the previously defined agenda, and assists the meeting moderator in managing the meeting. It allows for short term changes of the agenda and associates events with the appropriate agenda items. It keeps track of the time budget for each agenda item and permanently informs about the meeting progress. The goal supervision component acts as counterpart to agenda processing. It uses the meeting’s goal hierarchy and controls the degree of fulfillment for each defined goal. Similar to agenda processing, goals can be changed during the meeting. However, late changes cause heavy-weight management operations due to the restructuring of the goal hierarchy. Furthermore, participants and the moderator might want to overrule automatic goal fulfillment, which is also achieved by human initiated changes during runtime. The goal supervision and agenda processing modules are closely related.

The moderator is further supported by the moderation support component, which manages interactions with the meeting moderator and provides the interface to all functionalities available within the runtime environment. The matching of goals to multimedia services provided by the smart meeting room is performed by the goals/services broker. It infers the meeting’s requirements from the agenda and the goal hierarchy (and includes additional requirements defined by the meeting moderator and/or meeting participants). For each goal the broker searches for a multimedia service currently registered and capable of solving this goal. Such a multimedia service consists of features combined by device and software capabilities and exhibits supporting sub-services. A detailed introduction to multimedia services is given in Section 5, the matching algorithm is described in Section 6.2.

The sensor tracking module is used to monitor environmental changes. Physical meeting context is included by means of sensors, like temperature or luminance sensors, proximity sensors (for example based on RFID technology), or location systems like the Ekahau positioning engine for indoor location tracking. The generated traces allow the system to react to environmental changes, that is, to become context-aware, to log information for post-meeting processing reasons, and to learn from observations.

Finally, the privacy and security module guarantees access control by means of authorization based on the meeting definition (agenda and goal hierarchy) and a role concept.

### 3.4 Learning & Self-Adaptation

In order to design a satisfactory proactive meeting assistant, the architecture includes a learning and self-adaptation module. Traces stored in the repository represent the lessons learned. In subsequent meetings, the system is able to make use of the digital memory created and can adapt itself to the requirements of the user in a more adequate way.

### 3.5 Object Space

The object space serves as a virtual shared memory for objects that are processed during the meeting. These objects include, but are not restricted to files, persons, devices, and physical objects. For example, a presentation file can be inserted into the object space which will be used by the keynote speaker of a meeting. The objects are assigned to goals and accessed by the matching multimedia services as proposed by the goal/service broker. Service access to objects is restricted by these assignments. The object space, for example, is used to realize chat or file-sharing applications for meeting participants. Object characteristics are described by means of the Resource Description Framework (RDF).\(^2\)

For every object a description based on an OWL ontology is plugged into the system.

---

\(^1\)http://www.w3.org/2004/OWL/  
\(^2\)http://www.ekahau.com/  
\(^3\)http://www.w3.org/RDF/
4. GOAL MODELING

The goal-driven meeting support approach uses the abstraction of goals in order to separate logical relations of meeting items from the services realizing and fulfilling the goals depending on the physical context of the meeting. This approach is able to support human requirements and preferences in a meeting better because of the following advantages: (1) goals model human preferences and requirements better than services, (2) the specified goals persist although the services might be adapted or changed, (3) the computation of goals and their relations hides the complexity from embedded services in the environment, which are expected to be numerous but simple in terms of processing and reasoning capabilities.\(^4\)

Based on the implicit and explicit knowledge about human meeting goals, room goals are derived. The following subsections refer to these room goals.

4.1 Goal Representation

We model the goal-driven framework by means of a goal hierarchy and goal relations influenced by first-order logic calculus [14] and the following vocabulary:

**Predicate variables:** \(G_i\) denotes a (sub)goal of the set of Goals \(G\) which can evaluate to a fulfillment degree \(f(G_i) : G \mapsto [-1; 1]\) indicating the estimated degree of fulfillment for each goal. For example, 1 indicates that this goal has been optimally reached, 0 indicates that the goal has not been reached, and \(-1\) indicates that a multimedia service has caused malign effects on a goal.

**Goal variables:** \(v_i\) denotes a variable used in goal statements.

**Goal constants:** from the goal evaluation perspective, \(C_i\) denotes a constant value. However, these constants are defined by the runtime environment in terms of attributes.

**Logical operators:** The logical operators used are defined by \(!\) (similar to not), \&\& (similar to and), and \(||\) (similar to or).

**Iterator:** The quantifiers are expressed by means of the other logical operators. Additionally, we introduce a new iterator operator \(\oplus\), which is an adaptation of the universal quantification in terms of degree calculation.

We include neither conditional nor bi-conditional operators, since these logical operators can be expressed by the other logical operators and have no special merit in our approach. In addition to the possibilities of hierarchical modeling, we use metadata to reason about ordering of goals in terms of time, that is, deadlines. These metadata can be further enhanced, for example, to model fail-over behavior in case a goal cannot be met. Goals do not describe whether they can be executed in parallel or sequential, which is up to the runtime environment (i.e. the broker).

Each goal consists of either a combination of subgoals termed composite goal or a fact termed atomic goal. Leaves of the goal hierarchy, i.e. atomic goals, are mapped to services. No further restrictions of goal decomposition are included in our model. Thus, each goal is denoted by the following \(n\)-tuple

\[
G_i = (L_i, C_i, T_i, \ldots, Op_i),
\]

where \(L_i\) denotes the goal’s label, \(C_i\) the goal’s attributes, \(T_i\) the goal’s deadline derived from the meeting agenda if specified, and \(Op_i\) the operator and its arguments in case of a composite goal. Figure 3 depicts a typical goal hierarchy, the corresponding tuples are given as follows:

\[
G_0 : = (L_0, C_0, T_0, \ ||(G_{0.0}, G_{0.1}))
\]

\[
G_{0.1} : = (L_{0.1}, C_{0.1}, T_{0.1}, \ &&& (G_{0.1.0}, G_{0.1.1}, G_{0.1.2}))
\]

\[
G_{0.1.2} : = (L_{0.1.2}, C_{0.1.2}, T_{0.1.2}, \oplus (v, C_v, G_{0.1.2.0}))
\]

For a composite goal \(G_i\), the degree of fulfillment is calculated according to its goal hierarchy. A weight \(a_{ij}\) is assigned to each goal \(G_{ij}\) used for self-adaptation. Depending on the calculated goal fulfillment degree the system suggests multimedia services and monitors user feedback and interactions. In case the recommended services were rated beneficial, the weights will be increased. Otherwise they will be decreased.

For the operations specified, the fulfillment degree is calculated as follows:

\[
\&\& (G_{i.0}, \ldots, G_{i.n}) : f(G_i) = \min_{j=0}^n a_{ij} f(G_{i,j})
\]

\[
|| (G_{i.0}, \ldots, G_{i.n}) : f(G_i) = \max_{j=0}^n a_{ij} f(G_{i,j})
\]

\[
!(G_{i.0}) : f(G_i) = -G_{i.0}
\]

\[
\oplus (G_{i.0}) : f(G_i) = \sum_{i=1}^n a_{ij} f(G_{i,n}) / n
\]

(where \(a_{ij} \in [0; 1]\), \(\sum_{i=1}^n a_{ij} = 1\).

4.2 Goal Ontology

The goal ontology is defined by means of OWL DL. OWL provides superior means for describing both vocabularies and relationships between terms in a machine-interpretable manner. This characteristic is used to describe the previously introduced goal hierarchy. Goals and operators are defined as depicted in Listing 1.

\[\text{Figure 3: Goal hierarchy}\]
Composite goals (and subgoals) which allow to structure the goals hierarchically are realized via arguments the operator accepts. These arguments can be individuals of the classes Operator or Goal, as is shown in Listing 1.

The operators are all defined in a similar manner. Listing 2 describes how the most complex operator, that is, the iterator, is described in OWL.

5. SERVICES
When acting in an unfamiliar environment it can be obstructive to have a set of rich multimedia devices and no operating experience. In place of the user, the environment can manage these devices and services minimizing the personal administration overhead. In order to access and control a diverse set of components, we propose a service oriented architecture. In principle, this architecture provides an abstraction from the necessary environmental communication. Therefore, the service structure, communication protocols, as well as context features are encapsulated and hidden from the runtime environment. In contrast to using the broker for reasoning about the context (such as proposed in [6]), services aggregate the context parameters by means of the fulfillment degree for a specific goal. Since the broker matches services against goals, the service ontology has to be accessible by the broker. Additionally, the broker transfers information about attributes defined by the goals and bound to the services during runtime. The following subsections detail these concepts.

5.1 Service Architecture
For the design of our system, we choose Web Services\(^5\) as interface to components of the service architecture which are depicted in Figure 4. Due to the communication abstraction layer, there is no need for the runtime environment to deal with service specific problems like service discovery or protocols, but the service scheduling and invocation is controlled by the runtime environment. Referring to the goal driven approach all services are considered to be atomic. Service composition is realized through goal composition and matching each atomic goal to one service. Every service accessible by the runtime environment has to provide four functions: (1) estimated degree of fulfillment, (2) estimated time of execution, (3) actual fulfillment after execution is completed, (4) associated service class(es) based on the service ontology.

Listing 1: Goal and operator definition

```
Listing 2: Iterator operator

Listing 2: Iterator operator

Figure 4: Device communication architecture

\(^5\)http://www.w3.org/2002/ws/

59
We distinguish between two types of service communication protocols. Either a service may be accessed by means of an smart meeting room specific communication protocol, or a standard communication protocol is used. While the first option is beneficial for specific domain dependent services, the latter option allows to include additional services based on well-known frameworks in the field, like Jini\(^6\) or UPnP\(^7\). Standard protocols will be supported in our architecture by means of proxies.

5.2 Service Semantics

We choose to support the broker by defining an upper ontology that describes the main concepts of services in the environment. Although OWL-S\(^8\) is currently investigated as a possible standard for the semantic description of web services [16], there are still drawbacks [1]. For the purposes of our goal/service broker we favored a proprietary semantic description of services in OWL DL.

5.3 Attributes

The broker supplies the services with goal attributes from the object space based on a standard upper attribute ontology. The services use these attributes not only to calculate fulfillment degrees for goals, but also for service provisioning purposes, like e.g. an MPEG-4 coded video (and its structure) required by a video streaming service. The information transferred from the broker to the service further includes information about the impact to the fulfillment degree each attribute exhibits.

Services use domain specific data structures expressed in RDF to exchange object data based on different standards and formats. Although perfectly suitable for our approach we are planning to verify the MPEG-21 framework [4] as a possible substitution or extension. The benefits of using the MPEG-21 framework as internal attribute representation would be the standardized integration of desired qualities like capability and profile information similar to CC/PP\(^9\), Digital Item Declaration, Intellectual Property Management and Protection, and Digital Item Adaptation.

5.4 Capabilities and Profiles

In order to calculate a significant fulfillment degree the service has to be aware of the subsystem, that is, its capabilities and profiles. We assume that services will use standard technologies for this purpose like CC/PP, which—like many other current approaches—only provides syntax descriptions. However, for a seamless integration of diverse devices, networks, and services, it would be important to standardize semantics as well [5]. We propose to hide these semantically heterogeneous capability descriptions by encapsulating these aspects by the service.

6. ONTOLOGY-BASED MATCHING

The usefulness of well-defined ontologies, the expression of ontologies using Semantic Web technologies [3], and their usage for matching of requirements and services have been described, e.g. in [13, 6, 10, 8]. In our scenario, we use the power of combining the open world assumption with extensibility. Our architecture only defines a set of upper ontologies, which provide a common vocabulary for the modeling of concrete domain ontologies. The role of these upper ontologies in the matching process is depicted in Figure 6.

[Figure 5: Ontology-based matchmaking]

We define only the top-level classes of ontologies to provide a lowest common denominator for the modeling of goals, objects, and services. These upper ontologies are sufficiently expressive. Still, they are sufficiently general so that domain-specific features can be integrated smoothly. For the meeting scenario, an ontology of meeting goals is proposed, which makes use of the common goal ontology. For the definition of meeting objects, there will be ontologies for file objects, person objects, etc. which are dynamically integrated into the system on demand. For services within a multimedia environment, there will be an ontology that describes, e.g. digital white-boards or location sensor technology. Services may describe themselves using the service ontology. However, to keep components small and efficient, they must only implement and understand the parts of the ontologies required for their functionality. The following subsections describe in detail how goal attributes are bound to values and how matching services are found based on these ontologies.

6.1 Attribute Binding

As described in Section 4, attributes are used to describe goals in more detail. Before a goal can be matched to a service, all its attributes must be bound to objects residing in the object space. As described above, every object in the object space is described by a metadata record. During the definition of the agenda, the meeting moderator defines a set of matching criteria for goal attributes. For example, for a variable describing a Powerpoint presentation by Niko Popitsch, the matching criteria may be \{ dc:type = ".ppt", dc:title = "METIS in a multimedia environment", dc:creator = "niko.popitsch@researchstudios.at" \}. If no such objects are available, or the appropriate object cannot be determined unambiguously, the meeting moderator is prompted to provide the object, or to select one out of the set of appropriate objects.

6.2 Goal Matching

All goals whose attributes are bound to objects are organized into a goal priority list. According to this processing the result of this processing is a list of services \( B_G \) that are able to fulfill goal \( G \) with a degree of at least \( q \), ordered by their estimated execution time.
In detail, the algorithm for matching a set of goals $AG$ to a set of available services $S$ is depicted below. Each service $S \in S$ must provide a function $S.fulfills(\cdot) : AG \rightarrow [-1; 1]$, indicating the estimated degree of fulfillment for each atomic goal, where 1 indicates perfect support for this goal, 0 indicates that the service has no effect on this goal or is not aware of the goal’s description, and $-1$ indicates that the service has a destructive effect on this goal. Furthermore, each service must provide a function $S.time(\cdot) : AG \rightarrow \mathbb{N}$, which indicates the amount of time that this service requires to process the request. This function is necessary for timely invocation of services. In order to increase efficiency, services may describe themselves in terms of classes of the service ontology: $S.classes(\cdot) : \{\} \rightarrow SC^n$, with $SC$ being the set of service classes. Using this function, the broker is able to restrain the set of services to be queried to those belonging to specific classes, namely the classes that were able to fulfill the goal in previous situations.

The algorithm can be configured using a quality threshold $q$. If possible, only services with $S.fulfills() > q$ are selected. If none of the services can provide an adequate degree of fulfillment, the matching returns an empty set. Subsequently, the user is asked to reduce $q$ so that a matching service can be found.

**Algorithm 1** Match($G$, $S$, $q$)

```plaintext
B_G \leftarrow \{\}

for all $S \in S$ do

  if $\exists S.classes(\cdot) \in classes(G)$ and $S.fulfills(G) \geq q$ then

    add $S$ to $B_G$

  end if

end for

order elements $S \in B_G$ according to $S.time(G)$
```

In a second step, the list of eligible services is further processed, considering the goal trees for the situation. In particular, goals that are outperformed by their alternatives within one $||()$ term are removed from the list. Goals that are argument to an iterator ($\oplus$) must be multiplied so that the corresponding services repeatedly occur in $B_G$, identified by different parameter instances. If there are fulfillment conflicts (conflicts that occur because one capability supports one goal, but obstructs another one), they must be resolved so that the overall goal fulfillment is maximized.

Finally, if there is only one service in $B_G$, it is scheduled for execution. Otherwise, the meeting moderator may select one service that is appropriate for him, or the goal/service broker may select a service based on global configuration options. Moreover, the self-adaptation module might help at this point. Preceding user selections and interactions will be analyzed to present alternatives.

### 6.3 Meeting Processing

The first cycle of goal/services matching is executed at the beginning of the meeting. For every scheduled goal/service relation, there exists an optimal start time of execution. This value is calculated from the estimated time that a service requires to fulfill a goal and the time that the agenda schedules for this goal to be fulfilled. Thus, the goal/service broker is able to invoke services timely.

However, the execution schedule must be revised by the goal/service broker regularly because of various reasons. The execution of a service may fail, e.g. because of technical problems. In this case, the corresponding goal must be reassigned to the second best service which is able to fulfill it. Additionally, the goal hierarchy may change during the meeting because of short-term modifications of the agenda. Then, the new or modified goals must be (re)matched to services, or cancelled goals must be removed from the execution schedule. Finally, the service environment may change due to context dynamics. New services may register themselves (in this case, the goal/service matching must be revised to check whether there exist new services that may fulfill goals better), or services may disappear from the system. Parameters for goal fulfillment regarding the meeting context (e.g. the persons which are present in the room, the lighting conditions, or the temperature) may change; hence, goals that were assigned to services must probably be re-matched to other ones.

#### 7. APPLICATION SCENARIO

To give an impression how a real meeting (Figure 6) in a multimedia environment is conducted, the following textual description of an application scenario illustrates the interaction between meeting attendees and the system as well as the collaboration of the system parts. This use case is presented in a descriptive manner from a user’s point of view, yet still mentioning system internal processes triggered by user interaction. (A formal use case description is omitted, because it would exceed the size of this paper.)

**7.1 Premeeting Phase**

In the premeeting phase the person responsible for arranging and preparing the meeting defines the agenda of the meeting. This is realized by an user interface of the agenda preparation module. It provides the possibility to enter parameters like the list of participants ($P_1...5$) and the agenda items ($A_1...5$). The agenda items for this sample meeting are:

![Figure 6: Typical meeting situation, participants, and their devices: in detail (1) information adapted for the PDA, (2) augmented presentation on notebook, and (3) RFID enabled PDA for proximity-based interaction](image_url)
• \((A_1)\) Welcome message
• \((A_2)\) Presentation of project 'Sample Project' by \(P_1\)
• \((A_3)\) Discussion
• \((A_4)\) Coffee break
• \((A_5)\) Collaborative editing of 'Sample Document'

The agenda items are internally represented as instances of classes of the agenda ontology. The ontology itself is formulated in OWL DL, as well as the other ontologies in the system (e.g., goal ontology, service ontology, attribute ontology). For essential fragments of the sample meeting’s representation see Listing 3, containing the instances of the meeting, the moderator, a remotely located participant and two agenda items. Omitting URIs in favor of short names as identifiers in the following sample listings is intentional to make them more readable.

Listing 3: The agenda’s representation depicting the sample meeting in abbreviated form

```
...<Meeting rdf:ID="Sample_Meeting">
  <moderator>
    <Person rdf:ID="P1">
      <name name_P1/>
      <location>
        <local rdf:ID="Meeting_Room_R"/>
      </location>
    </Person>
  </moderator>
  <participant>
    <Person rdf:ID="P2">
      <remote rdf:ID="Office_P2"/>
      <location>
        <attends rdf:resource="#Sample_Meeting"/>
        <name name_P2/>
      </location>
    </Participant>
  </participant>
  <agenda_item>
    <Presentation rdf:ID="my_Presentation">
      <topic>Project XYZ</topic>
      <duration>30</duration>
    </Presentation>
  </agenda_item>
  <agenda_item>
    <Break rdf:ID="my_Coffee_Break">
      <topic>none</topic>
      <duration>15</duration>
    </Break>
  </agenda_item>
</Meeting>
```

Through analyzing the list of participants and their contact information known to the system, participants are informed about the meeting (e.g., for participant \(P_2\) an entry in her Outlook calendar is placed, for participant \(P_3\) a notification via email is sent, ...). Additionally, required resources are reserved (e.g., meeting room \(R\) on date \(d\) from \(t_B\) to \(t_E\), ...).

If the user authorizes the agenda, it is transformed into a goal hierarchy by the goal definition module. The goal hierarchy is created based on the predefined goal hierarchies related to a certain kind of agenda item. In the first stage a full hierarchy is built, which is further manipulated through the learning & self adaption module in a second step. However, branches the user has marked as non-relevant in goal hierarchies of former meetings are omitted. Finally, if necessary, the goal hierarchy can be manipulated directly by the user herself. In Listing 4 the essential fragments of the goal hierarchies’ representation depicting the agenda item “Presentation of Project ‘Sample Project’” are shown. It is assumed that this goal representation has already passed all stages of post-processing mentioned before. The listing contains a branch of the goal hierarchy describing the goal to convey presentation \(Q\) to the meeting participants \(P_i\). This goal \(G_0\) should be fulfilled by the subgoal \(G_1\) (i.e., visualize the presentation) and \(G_2\) (i.e., deliver a copy of the slides); \(G_1\) can be fulfilled by its subgoals, either \(G_{10}\) (i.e., visualize the presentation via a shared output device), or \(G_{11}\) (i.e., visualize the presentation via an individual output device iterated over the list of participants).

Listing 4: The goal hierarchies’ representation depicting the agenda item “Presentation of Project ‘Sample Project’” in abbreviated form

```
...<Goal rdf:ID="G0">
  <variable rdf:resource="#var_Presentation"/>
  <variable rdf:resource="#var_Participants"/>
  <label>convey presentation \(Q\) to participants \(P_i\)</label>
</Goal>
<composite>
  <And rdf:ID="And_G0">
    <argument rdf:resource="#G1"/>
    <Goal rdf:ID="G2">
      ...
    </Goal>
    </argument>
  </And>
</composite>
<is_part_of rdf:resource="#Sample_Meeting"/>
</Goal>
<Goal rdf:ID="G1">
  <composite>
    <Or rdf:ID="Or_G1">
      <argument rdf:resource="#It_G11"/>
      <Goal rdf:ID="G10">
        <label>display presentation \(Q\) on shared visual output device</label>
        <variable rdf:resource="#var_Participants"/>
        <is_part_of rdf:resource="#Sample_Meeting"/>
        <variable rdf:resource="#var_Presentation"/>
      </Goal>
      <has_argument/>
    </Or>
  </composite>
  <variable rdf:resource="#var_Presentation"/>
  <variable rdf:resource="#var_Participants"/>
  <label>visualize Presentation \(Q\) to Participants \(P_i\)</label>
  <is_part_of rdf:resource="#Sample_Meeting"/>
</Goal>
<Iterator rdf:ID="It_G11">
  <argument>
    <Goal rdf:ID="G11">
      <label>display presentation \(Q\) on personal output device</label>
      <is_part_of rdf:resource="#Sample_Meeting"/>
      <variable>
        <Variable rdf:ID="var_Participants"/>
      </variable>
    </Goal>
  </argument>
</Iterator>
```

\(^{10}\) The branch describing \(G_2\) was stripped out of the listing because of lack of space.
the concept of goals. We introduced a goal-driven service-oriented architecture for the integration of simple multimedia services and applied the approach to the use case of a smart meeting room. The proposed software service architecture is based on the separation of goals and multimedia services, which reduces complexity for services and increases flexibility. We described how Semantic Web-based ontologies can be used for the modeling of such goals and services, and how the services are matched to goals in order to fulfill user requirements while the context changes dynamically. Finally, we demonstrated our concept for a multimedia-enriched meeting scenario. In future work, we plan to exploit our architecture for new multimedia services and self-adaptive goal composition.

9. ACKNOWLEDGMENTS

We thank Werner Winiwarter, Niko Popitsch and Bernhard Haslhofer for their valuable comments and suggestions.

10. REFERENCES


7.2 Conducting the Meeting

The main responsibility for a successful conduction of the meeting is taken by the goals/services broker. In a first step, the variables of goals are bound to their instances (e.g. for $G_1$ the variables are the participants $P_i$ and the presentation $Q$ itself). Subsequently, according to the deadline parameter of the goals, they are sorted into a goal priority list and further processed by the broker. Considering goal $G_{10}$ (visualize the presentation via a shared output device) as an example, the service $S_{23}$, which is capable of displaying the presentation $Q$ in its format on a beamer, returns an estimated fulfillment degree of 0.9, while the service $S_5$ returns a degree value of 0.01, because it has only access to a very small output screen (e.g. smart-phone display). Following the algorithm defined in Section 6.2, the broker would decide for service $S_{23}$ to fulfill goal $G_{10}$. After the presentation was effectively displayed the service returns the actual fulfillment degree back to the broker. See Listing 5 for a service description.

Listing 5: A service description in abbreviated form

```xml
...<Service rdf:ID="#S23">
  <service_function>
    <Output rdf:ID="#Output_23">
      <function_of_service rdf:resource="#S23"/>
    </Output>
  </service_function>
  <service_of_device rdf:resource="#Monitor_42"/>
  <attribute rdf:ID="#presentation_23"/>
  <status>
    <Online rdf:ID="#Status_Online_S23"/>
    <attribute rdf:ID="#scope_23"/>
  </status>
</Service>
Device rdf:ID="#Monitor_42">
  <device_service rdf:resource="#S23"/>
</Device>
...```

This procedure is executed for all atomic goals of the agenda items the meeting consists of. Hence, the successful execution of the meeting is ensured. The recorded data about the actual fulfillment degree of goals is stored in the system’s repository and used as input data for the learning & self-adaptation module and for evaluating the performance of the system for the conducted meeting.

8. CONCLUSION

In this paper, we presented an approach to model the user requirements in multimedia-enriched environments based on


Author Index

Balke, W.-T. ...................................................... 3
Götz, S. ........................................................... 31
Herborn, S. ...................................................... 21
Hummel, K. A. ................................................ 55
Ito, M. ............................................................. 37
Jochum, W. ...................................................... 55
Kalasapur, S. ................................................... 11
Kumar, M. ....................................................... 11
Leitich, S. ...................................................... 55
Lopez, Y. ........................................................ 21
Nahrstedt, K. .................................................... 3
Pavlovski, C. J. ................................................ 47
Polet, Q. S. ....................................................... 47
Schandl, B. ...................................................... 55
Seneviratne, A. ................................................ 21
Shibata, N. ...................................................... 37
Shirazi, B. ....................................................... 11
Staes-Polet, Q. ............................................... 47
Sun, T. ............................................................ 37
Tamai, M. ....................................................... 37
Wehrle, K. ....................................................... 31
Yamaoka, S. .................................................... 37
Yasumoto, K. .................................................. 37
Zimmerman, R. ................................................. 1
NOTES