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.

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, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

\textcopyright{} 2005 ACM 1-59593-245-3/05/0011...$5.00.

![Figure 1: Variations of composed media service paths](image)

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.
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 entail 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.

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. introduce 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 an 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 organizing 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 median 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. Since 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:

\[
\Gamma = \{ l_i \mid i \in M \}, \quad \text{where} \quad \forall l_i \exists l'_i \mid l_i \prec l'_i, i = 1 \ldots (m - 1),
\]

\[
\forall k \exists l_i \mid l_i \prec l'_i, k = 2 \ldots m, \quad | l_i | = | l'_i | = h = 1 \ldots m, h \neq i
\]

for all \( l_i \in \Gamma \), where \( M = \{1 \ldots m\} \) and \( m \) is the number of classes in the description scheme, in our case four. The notation \( x \prec y \) indicates that \( x \) is a child of \( y \), \( x\prec y \) is a parent of \( y \). Each class \( l_i \) contains zero or more elements \( \xi_i \), i.e. \( l_i = (\xi_1^i, \ldots, \xi_n^i) \) for \( n \geq 0 \), where \( n \) is the number of elements in to \( l_i \) 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 \( \gamma \subseteq \Gamma \), where \( | \gamma | = 1 \). 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 λ denotes a given irresolvable MediaClient description and γ denotes an irresolvable multimedia content description we define Diff(λ, γ) = Ψ to represent the end-to-end mismatches between the client and the content,. Thus, Ψ denotes a set of adaptations required on the path between the content and the client, e.g. no adaptations are needed if Ψ = ∅. If for a given MediaPort P, Ψ denotes the set of adaptations required after passing the media through P, then the condition |Ψ| < |Ψ| 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 1: Notation

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>A unresolved media client description</td>
</tr>
<tr>
<td>Γ</td>
<td>A unresolved media content description</td>
</tr>
<tr>
<td>λ</td>
<td>An irresolvable media client description</td>
</tr>
<tr>
<td>γ</td>
<td>An irresolvable content description</td>
</tr>
<tr>
<td>Ψ x,y</td>
<td>List of changes needed for x to become y</td>
</tr>
<tr>
<td>P t</td>
<td>A MediaPort input/output description</td>
</tr>
<tr>
<td>Diff(Γ, A)</td>
<td>Difference function</td>
</tr>
<tr>
<td>ξ &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>Ω</td>
<td>An end-to-end service graph</td>
</tr>
<tr>
<td>ω</td>
<td>A serial sub-path belonging to Ω</td>
</tr>
</tbody>
</table>

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 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 <codec> and <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.

```xml
<user name="bob">
    <audio:language value="french">
        <audio:rating value="[G,PG]" />
        <codec value="DIVX" />
    </audio>
    <target name="mediaplayer">
        <app name="mediaplayer">
            <audio:language value="french">
                <audio:rating value="[G,PG]" />
                <codec value="DIVX" />
            </audio>
        </app>
    </target>
    <ifc name="LAN">
        <video:screen value="[160*120,320*240]" />
        <bitrate value="(0,1000)" />
    </ifc>
    <ifc name="GPRS">
        <video:screen value="[620*480,1024*768]" />
        <bitrate value="(0,10000)" />
    </ifc>
</ifc>
</user>
```

**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...
translation service is needed on all audio flow sub-components of the media content.

\[
\text{From Figure 4 it can be seen that we model MediaPorts as simple irresolvable descriptions.}
\]

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

**Figure 3:** Sample XML description of content endpoint

### 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 ‘Splitters’ or ‘De-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.

**Figure 4:** MediaPorts

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 is output ports.

\[
P_I = \{i_1, \ldots, i_n\} \text{ and } P_O = \{o_1, \ldots, o_k\} \text{ where } k \geq 1, \ k' \geq 1, \text{ and } k > 1 \Rightarrow k' = 1, \text{ and } k' > 1 \Rightarrow k = 1.\text{ From the figure it can also be seen that as long as MP2 can receive the output of MP1 then the two MediaPorts may be chained in serial during a service path search. A node is said to be complete if, during the execution of the service graph routing algorithm, there exists a completed path from the MS for all outputs of a de-multiplexer, or all inputs of a multiplexer. A node is required to be complete before it can be included on a composed service path. We make use of this definition in the routing algorithm detailed in section 5.}
\]

### 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 5:** Sample output of comparison

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. \( \text{[Diff(bob, starwars)]} > 0 \). 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: English audio \( \rightarrow \) French audio, R-rated audio \( \rightarrow \) G/PG rated audio, DIVX encoding to MP/RM encoding. The video screen resolution and bitrate values necessitate no specific end-to-end adaptation.

As a rule, we order the possible combinations according to the number of end-to-end mismatches, though this may not be an optimal approach and is thus an area for future work.
We now identify and specify a set of three functions that need to be provided in order to construct directed service graphs, including Compatible, Adapt, and Useful. The function Compatible is used to determine whether or not a given input port is able to receive a media flow in its current state. The Adapt function generates a description of the result of applying a given MediaPort to a media flow. The Useful function, the most complex of the three, is used to determine whether or not a given MediaPort is ‘useful’ in the context of a certain media session, i.e. whether or not the MP brings the media closer to its goal state. We provide expressions of the three functions below.

The function Compatible is defined below as a logical expression. The precise implementation of the ‘intersection’ operator varies depending on the type of value being compared. For example, the intersection of range values or enumerated list values is straightforward, however the intersection of a single value element depends on whether or not it is mandatory. Intersection of two mandatory elements will return an empty set if their values are not equivalent, whereas intersection of non-mandatory values will always return a singleton set.

\[ \text{Compatible}(\gamma, \mathcal{P}) \equiv \forall \exists \gamma' : \{N, t, v\} \in \gamma \exists \gamma : \{N, t, v\} \in \mathcal{P} : N_i = N, t_i = t, v_i \cap v_i \neq \emptyset \]

Expression 1: the Compatible function

The expression below is a logical representation of the Adapt function that is used to determine the result of passing a media stream through a given MediaPort. We use the underscore character to denote a wildcard.

\[ \text{Adapt}(\gamma, \mathcal{P}) \rightarrow \gamma' \text{ where} \]
\[ \gamma' = \{e \mid e(\{N, t, \_\}) \in \gamma, e(\{N, t, \_\}) \notin \mathcal{P}\} \]
\[ \cup \{e \mid e(\{N, t, \_\}) \notin \gamma, e(\{N, t, \_\}) \in \mathcal{P}\} \]
\[ \cup \{e(\{N, t, v\}) \mid e(\{N, t, v\}) \in \gamma, e(\{N, t, v\}) \in \mathcal{P}, v \neq v'\} \]
\[ \cup \{e(\{N, t, v\}) \mid e(\{N, t, v\}) \in \gamma, e(\{N, t, v\}) \in \mathcal{P}\} \]

Expression 2: The Adapt function

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. \(\text{Diff}(\gamma, \gamma^\text{goal}) \cap \text{Diff}(\gamma, \gamma^\text{goal}) = \emptyset\) for the purpose of this expression.

\[ \text{Useful}(\gamma, \gamma', \gamma^\text{goal}) \Rightarrow \text{Diff}(\gamma, \gamma^\text{goal}) \Rightarrow \text{Diff}(\gamma', \gamma^\text{goal}) \cap \text{Diff}(\gamma, \gamma^\text{goal}) \]

Expression 3: the Useful function

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

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

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 \(\text{find_services}(\cdot)\), 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 \(\text{find_services}(\cdot)\) 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

![Comparison of service discovery mechanisms](image)

Figure 7: Scope comparison of service discovery mechanisms

### 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 iterate return results in batches, as is the present practice 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:**
\[
\forall \text{MediaPort } MP \in \text{service graph } \Omega, MP' \succ MP \\
D(MP, MS)/D(MP, MC) \geq D(MP', MS)/D(MP', MC) - \varepsilon
\]

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

Where \(D(A,B)\) is a distance function, and \(\varepsilon\) 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.

![Service path discovery algorithm](image)

Figure 8: Service path discovery algorithm

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(\(\gamma\), \(\gamma^{goal}\))\(\neq\)0
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 \(\omega' \triangleq \omega\) 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 (Useful(\(A\), Adapt(\(\gamma\), \(F^x\)))
14. define \(\omega' \triangleq \omega\) plus \(x\)
15. define \(A' \triangleq A\) less operations done by \(x\)
16. define \(\gamma' \triangleq \text{Adapt}(\gamma,y)\)
17. find_path(\(F^{A'}, A', \gamma, \gamma^{goal}\))
18. else if (\(x\) can split this flow)
19. for each (\(F^x\))
20. define \(P \triangleq P\) plus \(x\)
21. define \(\gamma' \triangleq \text{Adapt}(\gamma, F^{A'})\)
22. find_path(\(P', A', \gamma', \gamma^{goal}\))
23. return \(P^* = \Omega\)
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 an 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).

In the 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 search 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
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 active participants of 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