Towards an Integrated Multimedia Service Hosting Overlay

Dongyan Xu
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA
dxu@cs.purdue.edu

Xuxian Jiang
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA
jiangx@cs.purdue.edu

ABSTRACT

With the proliferation of multimedia data sources on the Internet, we envision an increasing demand for value-added and function-rich multimedia services that transport, process, and analyze multimedia data on behalf of end users. More importantly, multimedia services are expected to be easily accessible and composable by users. In this paper, we propose MSODA, a service-oriented platform that hosts a wide spectrum of media services provided by different parties. From the user's point of view, MSODA is a shared "market" for media service access and composition. For a media service provider, MSODA creates a virtual dedicated environment for service deployment and management. Finally, the underlying MSODA middleware performs the key functions of service composition, configuration, and mapping for users. We discuss key challenges in the design of MSODA and present preliminary results towards its full realization.

Categories and Subject Descriptors

C.2.5 [**Computer-Communication Networks**]: Local and Wide-Area Networks—Internet; C.2.4 [**Computer-Communication Networks**]: Distributed Systems—Distributed Applications

General Terms

Design, Management, Performance

Keywords

Media Service, Hosting, Composition, Virtualization

1. INTRODUCTION

Digital multimedia is permeating and enriching every aspect of our life. Rapid advances in multimedia hardware and software have brought about two trends: First, it becomes easier for end users to *consume* as well as to *generate* multimedia data. Webcams, surveillance cameras, on-line radio/TV portals, and media-enabled personal communication devices are being widely adopted. Meanwhile, modern computing platforms, ranging from desktops, PDAs, to digital appliances, are ready to process and display multimedia data of varying format and quality. Second, multimedia data become more *manipulatable* and the semantics, both within and between media streams, are increasingly being exploited. In addition to traditional format-driven media processing operations (e.g., transcoding), intelligent semantics-driven operations, such as scene change detection, pattern recognition, object tracking, and image synthesis can be performed on media streams in real-time. Following these trends, we envision the proliferation of *value-added* and *function-rich* media services that transport, process, and analyze multimedia data in an integrated fashion. More importantly, multiple media services may be composed into a new and customized service. As a motivating example, in disaster recovery, audio, video, and sensor data streams are aggregated, filtered, and analyzed (e.g., event correlation, victim/rescuer tracking) to assist in recovery operations. As another example, an outdoor event is captured and transmitted live to remote viewers, with background music added and camera jitter eliminated.

The potential of the above vision has not been fully realized by today's distributed multimedia applications. Real-time interactions between clients and servers or between peers are often in the simple form of streaming, while value-added media processing is not common. One reason is that, for both end users and media content providers, it is impractical to develop a full spectrum of media processing functions. Moreover, media processing can be highly computation intensive and requires high performance platforms. However, such platforms can not be uniformly deployed in every domain or organization.

To support the next-generation distributed multimedia applications, we propose MSODA, a service-oriented platform that *hosts* multimedia services for on-demand user access and composition. In MSODA, media processing and transport functions are packaged as *media services*. MSODA provides infrastructural resources - physical hosts called MSODA nodes - in the wide-area Internet, while individual media services are provided and maintained by various media service providers (MSPs). As such, MSODA federates a large variety of media services and forms a rich collaborative service hosting *overlay* network. With MSODA, users will be able to request, without having to develop or install by themselves, any media service MSODA offers. More importantly, users will be able to dynamically *compose* new customized media services, based on the basic services hosted by MSODA.

The design of MSODA includes three key aspects: (1) A *virtualization* approach is taken to decouple the provisioning of MSODA functions from individual media services. MSODA functions provide generic system support for the specification, configuration, and mapping of media service sessions. (2) MSODA enables flexible delivery of media services by suggesting alternative service

MM'04, October 10-16, 2004, New York, New York, USA.

configurations suitable for different resource conditions. By selecting from multiple service configurations, MSODA achieves high service admission rate. (3) MSODA performs cost-effective monitoring of end-to-end network conditions between MSODA nodes on behalf of *all* media services hosted. This approach leads to both low monitoring overhead and high service mapping success rate.

The rest of this paper is organized as follows. Section 2 describes the architecture of MSODA. Section 3 presents the service composition and configuration functions of MSODA. Section 4 presents MSODA's method of cost-effective overlay monitoring. Section 5 compares MSODA with related work. Finally, Section 6 concludes this paper.

2. MSODA: AN OVERVIEW

From the user's point of view, MSODA is a shared integrated "market place" for media service access and composition. Basic media services are provided by individual service providers and hosted in MSODA nodes. A media service can be as simple as a video transcoder or as complex as a multi-video stream correlator. MSODA creates a wide-area collaborative service hosting *overlay*, with the goal of high resource utilization, service quality, and service composability. Figure 1 shows the architecture of MSODA.

2.1 Virtualization-Based Service Hosting

Each MSODA node hosts a number of media services. Moreover, the same media service can be dynamically replicated or migrated within the MSODA overlay network, driven by service demand and resource availability. In an MSODA node, each media service instance runs inside a *virtual machine*. The virtual machine has its own IP address, customized operating system, and networking capability [17]. Such a virtualization-based approach brings the following advantages to MSODA:

- Decoupling of multimedia service management and MSODA platform management: It is desirable that an MSP has full administrator privilege of the virtual machine where its service is running. The MSP will thus be able to perform service-specific management and maintenance, without interfering with the management of the underlying MSODA platform and other services hosted on the same platform.
- Service installation isolation: Different media services may require the same software library, but *different* versions. They may also require the *same* port binding. The virtualization-based approach naturally resolves these conflicts among multiple services.
- Easy service relocation and replication: Virtual machines have highly efficient priming, resource scaling, and migration capabilities. As a result, media services can be easily replicated or re-located in the wide-area MSODA overlay.
- Isolation of fault and attack impact: Without virtualization, any fault or attack associated with one service will affect the MSODA platform and consequently other services. With virtualization, the negative impact will be contained within the virtual machine where the fault or attack takes place.

2.2 Three-Layer MSODA Middleware

The virtual machines for media service hosting are on the data plane of MSODA. On the control plane, MSODA adopts a layered approach to multimedia service composition, configuration, and mapping. These functions are performed by a three-layer middleware running in each MSODA node, as shown in Figure 1. Note that the MSODA middleware performs these functions for *all* media services hosted, so that the functions do not have to be implemented by individual MSPs, reflecting the advantage of decoupling MSODA management and the management of individual services. The three layers are as follows:

- The *service composition layer* facilitates the specification of new composite media services. Through a high-level API, the user will be able to specify temporal and data dependencies among the basic services involved in the requested composite service. A service specification may be independent of media format, service interface, and location of service instances.
- The *service configuration layer* accepts a service specification, resolves the basic services involved, and customizes the composite service according to client capability and user preferences/needs. More importantly, the service configuration layer is able to generate multiple candidate service configurations for one service specification. Each candidate has a different set up of basic services - with respect to their quantity and topology. If a candidate service configuration cannot be successfully mapped to the MSODA overlay (by the service mapping layer), the service configuration layer will suggest another candidate.
- The *service mapping layer* is responsible for mapping service configurations to the MSODA overlay. For each configuration, a *service delivery overlay* will be created. The service delivery overlay consists of virtual machines that provide the basic service instances in the configuration. To perform mapping of a service configuration to a service delivery overlay, the service mapping layer monitors (1) MSODA node capacity and (2) network performance between MSODA nodes. However, the monitoring of (2) for each pair of MSODA nodes will incur significant network probing overhead and is therefore undesirable. Instead, the service mapping layer dynamically selects a *subset* of end-to-end connections to monitor, which form a *mesh* - rather than a complete graph - to connect all MSODA nodes. A highly cost-effective method is designed for the maintenance of the mesh (to be presented in Section 4).

3. SERVICE COMPOSITION AND CONFIGURATION

3.1 Service Composition

The service composition layer of MSODA middleware defines an API for users to specify the basic services involved in a new composite service, as well as the temporal and data dependencies among the basic services. The temporal relation between two basic services can be sequential, concurrent, or alternative. Moreover, it is desirable that the user does not need to know specific service interfaces and media formats when specifying a composite service. To support the service composition API, the service composition layer middleware performs validity check on service specifications and provides feedbacks to users.

To check service specification validity, a service profile database is created, to be accessed by service composition and configuration layers. The database stores profiles of basic services provided by their respective MSPs. A service profile includes interface definition, media format, and resource requirements. Based on the service profile database, the service composition layer is able to check

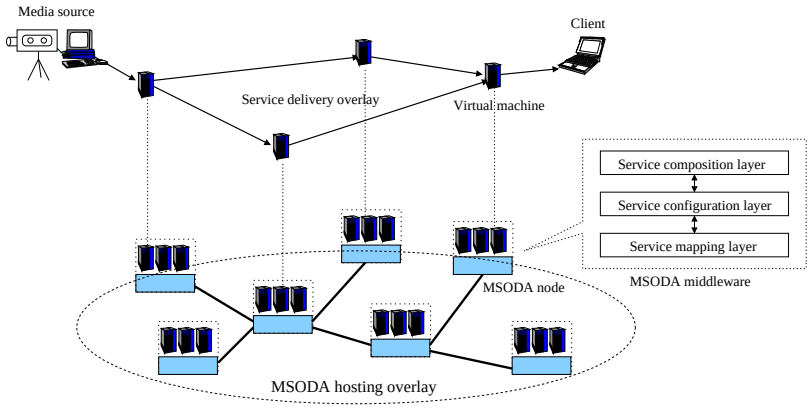


Figure 1: Architecture of MSODA

if any combination of the hosted services in MSODA is able to fulfill the composite service request. For example, if a user requests a "video transcoding + face recognition" service, the service composition layer will check if there exist instances of video transcoding and face recognition services, such that the output media format of the former can be accepted by the latter. However, if all face recognition services can only accept the *input* media format of the transcoding service, the service composition layer may suggest the user to switch the order between the two services.

3.2 Service Configuration

The service configuration layer accepts a valid service specification, and generates one or more candidate service configurations to be mapped to the MSODA platform. Given a service specification, the service configuration layer will resolve the abstract service instances and determine the interfaces, input/output media formats, and resource requirements of these service instances. The service configuration layer is expected to be intelligent: It may further refine and customize a service configuration according to user needs and client capability. If a service configuration cannot be mapped to the MSODA platform due to poor resource conditions, the service configuration layer will suggest an alternative service configuration, based on feedbacks from the service mapping layer about current MSODA resource availability. The service configuration layer middleware interacts with both service composition and mapping layers:

- Based on a service specification, the service configuration layer will generate a *customized* service configuration, which takes into consideration user preferences, client capability, and special assistance requirements. For example, if a hearing impaired user requests a service that involves a live-event video stream, the corresponding service configuration can be customized to add a "picture-in-picture" service, which merges a separate video stream showing the sign language interpreter into the original video stream. Such customization is highly service and user specific. The customization rules may be provided by users, MSPs, and the MSODA provider. The rules will be expressed in a uniform format and stored in the service profile database.
- Due to the dynamic node capacity and inter-node network performance in the MSODA overlay, the underlying service mapping layer may not be able to map a service configuration to the MSODA overlay. If so, the service configuration layer will suggest an alternative service configuration that leads to a different service delivery overlay. For example, in the original service configuration (Figure 2(a)), the bandwidth requirement between services S_1 and S_2 is 512Kbps. However, the service mapping layer reports that between any service instances of S_1 and S_2 in MSODA, the feasible data rate is no higher than 300Kbps. In response, the service configuration layer will suggest an alternative configuration as shown in Figure 2(b): Two instances of S_2 will split the output load of S_1, with each instance accepting a sub-stream of 256Kbps. The new configuration is more likely to be mapped successfully.

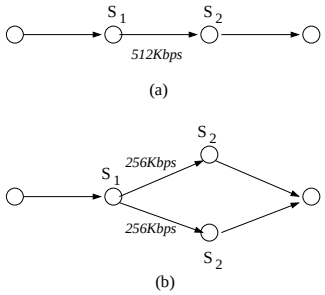


Figure 2: Original and alternative service configurations

One major challenge is to decide how to rank and select from the multiple candidate service configurations, in order to achieve high mapping success rate and low re-try overhead. A key property of the service configuration layer is *resource-awareness*: service configurations should always help to conserve resources. In the earlier example involving the hearing impaired user, suppose the service specification also includes a transcoding service. It is easy to see that the service configuration shown in Figure 3(b) is more likely to conserve bandwidth than the configuration in Figure 3(a).

To suggest resource-conserving service configurations, the service configuration layer is expected to have "knowledge" about many resource-saving techniques. In the example of Figure 2, the technique applied is *load partitioning* between two service instances. However, load partitioning is only applicable to services where the media data can be partitioned and re-assembled. Another resource-saving technique is *request aggregation*. If multiple service requests ask for the same media data, the requests can be aggregated into one shared service configuration, possibly with adaptation services for individual users [4].

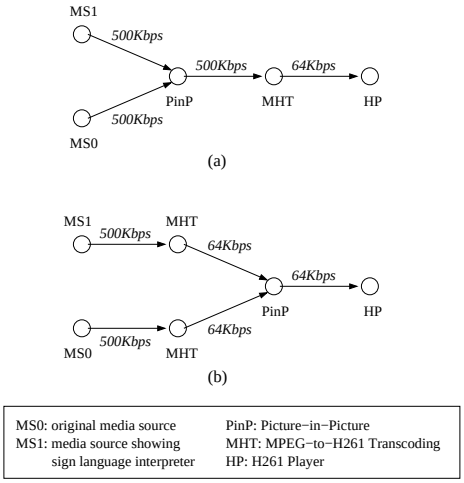


Figure 3: Two different service configurations for a hearing impaired user

4. OVERLAY MONITORING FOR SERVICE MAPPING

The service mapping layer maps service configurations, as *service delivery overlays*, to the MSODA service hosting overlay. As a result, multiple service delivery overlays will operate on top of the MSODA overlay. Service mapping involves two interrelated tasks: (1) to map the basic media services to MSODA virtual machines and (2) to route the media streams in the MSODA overlay network. A service delivery overlay is expected to satisfy both local and network resource requirements specified in the service configuration. A main challenge in finding service delivery overlays in MSODA is the monitoring of dynamic capacity of network and MSODA nodes. Our goal is to find a resource-sufficient service delivery overlay for each service configuration, in an effort to avoid service session failure due to insufficient resources.

In this paper, we propose a *mesh maintenance and augmentation* method for highly cost-effective MSODA overlay monitoring. Compared with a basic monitoring method in our earlier work [28], this method results in higher service mapping success rate *without* increasing the network probing overhead. For completeness of presentation, we briefly describe our approach in [28]: An algorithm is designed to find service paths, a special case of service delivery overlay with linear topology. The algorithm follows the "link-state" scheme. Each node in the MSODA overlay periodically probes (1) node capacity and (2) end-to-end bandwidth from this node to a selected *subset* of other nodes. The probing results are periodically propagated to other MSODA nodes. Therefore, each MSODA node is able to form a global view of the MSODA overlay, represented as a *mesh* consisting of all MSODA nodes and a *subset* of connections between them. The service path finding algorithm will be executed on the mesh to set up service delivery paths for service requests.

4.1 MSODA Mesh Maintenance and Augmentation

The service mapping layer of each MSODA node maintains the mesh. Let $C(P)$ denote the processing capacity of an MSODA node P. An edge (P, P') in the mesh corresponds to the overlay connection from P to another node P'. Let $B(P, P')$ denote the end-to-end bandwidth from P to P'. The mesh's effectiveness depends on (1) the number of MSODA overlay connections in the mesh and (2) the timeliness of monitoring results associated with the mesh. MSODA node capacity is monitored by querying the host OS of the MSODA node. MSODA overlay connection bandwidth can be monitored by probing at pre-determined data rates. Similar to [7], we use a set of discrete rates to estimate feasible bandwidth between MSODA nodes.

Overlay connection probing incurs non-trivial traffic and overhead. Therefore, it is critical to control the number of connections monitored by each MSODA node. However, the number of MSODA overlay connections in the mesh largely determines the success of service mapping. In Section 4.1.1, we first present the basic mesh maintenance method used in [28]. The new method of *mesh augmentation via cold-connection jumpstart* will be presented in Section 4.1.2.

4.1.1 Basic Mesh Maintenance

For each MSODA node P, we impose an upper bound on the number of actively monitored overlay connections, denoted as $d_{max}(P)$. We also denote the corresponding set of neighbor nodes as $Nbr(P)$. The elements in $Nbr(P)$ are initially set by the MSODA administrator. At runtime, P dynamically adjusts $Nbr(P)$ according to network probing results. To maintain the mesh, P periodically performs the following operations:

(1) **Probing and propagation** During every *neighbor-probing period* T_p, P probes the connection from P to each MSODA node $\in Nbr(P)$. T_p is a configurable system parameter. At the end of each neighbor-probing period, P propagates both local and network resource monitoring results to other MSODA nodes.

(2) **Mesh adjustment** During every *non-neighbor-probing period* T_q ($T_q > T_p$), P probes the connections from P to MSODA nodes *not* in $Nbr(P)$. To keep the probing overhead low, the non-neighbor-probing period is much longer than the neighbor-probing period. For each P_x not in $Nbr(P)$, P decides if P_x can be added to $Nbr(P)$ after the probing:

- Let $B(P, P_x)$ be the bandwidth from P to P_x via probing; and let $B(P \longrightarrow P_x)$ be the bandwidth on the path from P to P_x in the current mesh. If $B(P, P_x) \leq B(P \longrightarrow P_x)$, P_x will *not* be added to $Nbr(P)$.
- Otherwise, if the number of P's neighbors is smaller than $d_{max}(P)$, P_x will be added to $Nbr(P)$.
- Otherwise, P examines every MSODA node in $Nbr(P) \cup \{P_x\}$, and identifies such a P_v: In the mesh *plus* the edge from P to P_x, if edge (P, P_v) is removed, the end-to-end bandwidth from P to P_v will drop by the *least* percentage, compared with the current $B(P, P_v)$. P_v will be excluded from $Nbr(P)$. Note that the exclusion of P_v will not affect any on-going service sessions that involve edge (P, P_v).

4.1.2 Mesh Augmentation via Cold Connection Jumpstart

The basic mesh maintenance method in Section 4.1.1 is similar to methods proposed for application-level multicast [6, 7, 8]. However, we argue that the basic method is not sufficient for service mapping in MSODA because of the following problems:

(1) In an overlay for application-level multicast, the mesh is maintained for the construction of only one (or just a few) multicast tree. Therefore, even with a small value of $d_{max}(P)$ (i.e. each P only probes a small number of neighbors), the mesh will provide sufficient number of candidate edges to construct the multicast tree(s). However, in an MSODA overlay, a large number of service delivery overlays are running simultaneously. If the mesh is constructed using the basic method, it may not contain enough

MSODA connections to accommodate many service delivery overlays by satisfying their network resource requirements.

(2) In application-level multicast, between the source and each receiver, there may be a number of other receivers. In MSODA, however, if the mesh has a low $d_{max}(P)$ (and therefore limited connectivity), service delivery overlays computed based on the mesh may involve too many intermediate MSODA nodes (nodes that relay rather than process data), causing undesirable latency.

We propose our solution: mesh augmentation via cold-connection jumpstart that improves the basic mesh maintenance method. The new method is simple but highly cost-effective. It results in a highly connected mesh, without increasing bandwidth probing overhead. The key idea is to exploit the bandwidth probing results related to the "cold" connections, i.e. connections *not* currently in the mesh. The method involves the following operations:

(1) **Cold-connection jumpstart** Recall that in the "mesh adjustment" operation of the basic method (Section 4.1.1), if a "cold" connection (P, P_x) (P_x is not in $Nbr(P)$) is not added to the mesh, the bandwidth probing result will be discarded. We now give this cold connection a *jumpstart* by adding it to the mesh for one neighbor-probing period T_p.

(2) **Passive connection monitoring** If (P, P_x) is chosen to become part of a service delivery overlay before its expiration time, it is successfully jumpstarted and becomes a "warm" connection. P will keep track of the media data rate on (P, P_x) - as a passive bandwidth estimate in P's mesh, which will also be propagated to other MSODA nodes. The connection stays warm, until there is no media stream going through (P, P_x), or until a service delivery overlay fails to maintain its required data rate on (P, P_x).

By cold-connection jumpstart, the mesh of each MSODA node is augmented with warm connections that are passively monitored using the media streams going through them. Mesh augmentation complements the basic method in maintaining a highly connected mesh. Note that the mesh augmentation method does not incur *active* probing traffic. The effectiveness of mesh augmentation depends on system parameters including $d_{max}(P)$, the arrival rate of service mapping requests, and the duration of service delivery sessions. Intuitively, the lower the $d_{max}(P)$, the easier it is to jumpstart a cold connection. The greater the service mapping request rate and session duration, the easier it is to jumpstart a connection and keep it warm.

4.2 Performance Evaluation

In this section, we evaluate the effectiveness of the mesh maintenance method through both simulation and stochastic analysis.

4.2.1 Simulation Setup

The simulations are conducted in a virtual machine-based emulation testbed called vBET [18]. The simulated MSODA overlay and underlying IP network are shown in Figure 4. The connections between MSODA nodes in Figure 4 correspond to the *initial* mesh set by the MSODA provider. In each simulation run, we set d_{max} (the maximum number of neighbors for each node) to a different value. The total capacity of each node is randomly selected between 2500 units and 10000 units, while the total bandwidth of each underlying network link is randomly chosen between 400 units and 1600 units. At the IP level, routing between MSODA nodes follows the shortest path (with respect to hop-count) policy.

During each 3-hour run of the simulation, service mapping requests are generated according to a Poisson process. We use a different average request arrival rate λ_{req} in each run. For easy comparison with the basic mesh maintenance method used in [28], all service mapping requests are for linear service delivery *paths*: The source node P_s and destination node P_d of each request ($P_s \neq P_d$) are randomly selected from all MSODA nodes. The service session duration is exponentially distributed, with an average duration of 20 minutes. Each service configuration includes three basic media services. The MSODA node capacity required by each basic service is randomly chosen between 4 and 8 units, while the media data rate between services is randomly chosen from 1, 2, 4, and 8 units. The neighbor-probing period T_p is 5 minutes, while the non-neighbor probing period T_q is 30 minutes. In the simulation results, we will use the following notation for algorithms and their parameters: <algorithm name>(<with or without mesh augmentation>, d_{max}, T_p, T_q). For example, "SPF(w/, 3, 5, 30)" means "using the service path finding algorithm [28] *with* mesh augmentation method; $d_{max} = 3$; $T_p = 5$ minutes; and $T_q = 30$ minutes".

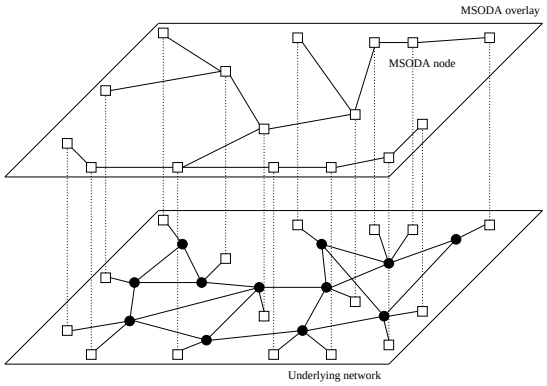


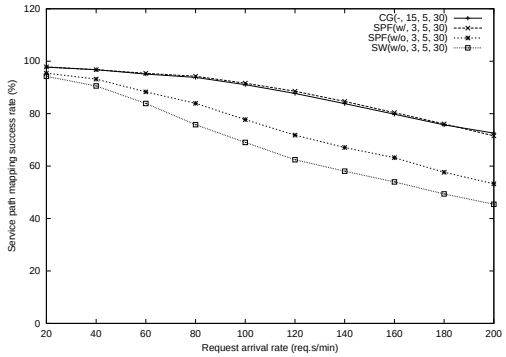
Figure 4: The simulated MSODA overlay and underlying network

4.2.2 Simulation Results

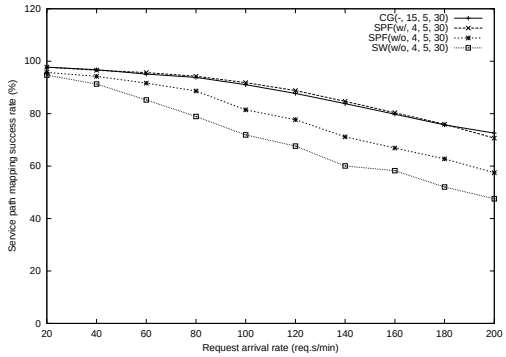
(1) Success rate of service path mapping A service path mapping is successful, if its resource requirements are satisfied *throughout* the service session. The following algorithms are compared:

- CG(-): The service path finding algorithm with a *complete graph* mesh, i.e. $d_{max} = 15$ for the MSODA overlay with 16 nodes.
- SPF(w/): The service path finding algorithm with the mesh augmentation method.
- SPF(w/o): The service path finding algorithm with the basic method only and without mesh augmentation.
- SW(w/o): The shortest-widest-path algorithm. The algorithm computes the shortest widest path from P_s to P_d, and determines the service-to-virtual machine mapping.

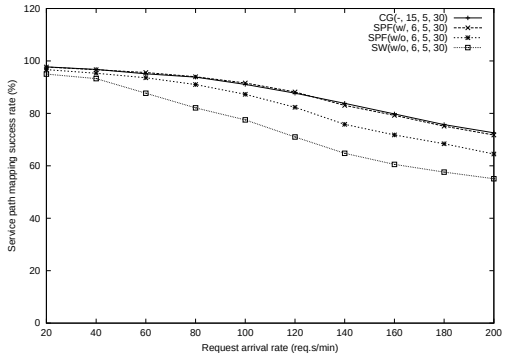
Figure 5 shows the service path mapping success rate achieved by the algorithms above, under request arrival rates ranging from 20 to 200 requests per minute. Each sub-figure shows the results under a different d_{max}. Algorithm SPF(w/) constantly achieves almost the same success rate as CG, and higher success rate than SPF(w/o), thanks to the highly connected mesh created by mesh augmentation. The lower the d_{max}, the larger the difference in success rate between SPF(w/) and SPF(w/o) and thus the greater the effect of mesh augmentation. Without mesh augmentation, SPF(w/o) constantly achieves higher success rate than SW(w/o), and the difference does *not* get smaller with the increase of d_{max}. This demonstrates the fundamental advantage of SPF over SW as have been shown in [28].



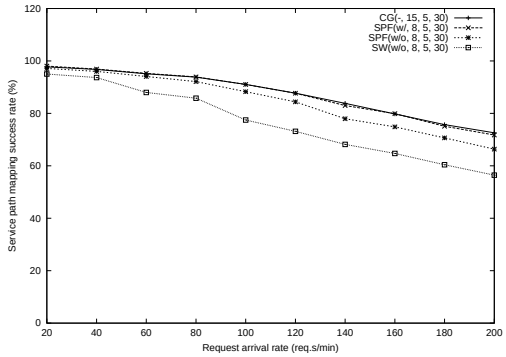
(a) $d_{max} = 3$



(b) $d_{max} = 4$



(c) $d_{max} = 6$



(d) $d_{max} = 8$

Figure 5: Service path mapping success rate under different request arrival rates

More importantly, we observe that SPF(w/) achieves the *same* high success rate as CG when d_{max} is only 3, indicating that SPF(w/) is highly cost-effective. In fact, this is due to a nice property of mesh augmentation: the lower the d_{max}, the more the cold connections to be jumpstarted and added to the mesh. This property is also confirmed by our stochastic analysis.

(2) Cost-effectiveness of mesh maintenance We evaluate the cost-effectiveness of a mesh maintenance method by the ratio between the total number of service path mapping successes and the number of active network probes (to *both* neighbors and non-neighbors) performed by each MSODA node. In other words, the cost-effectiveness is indicated by the number of successes "yielded" by each probe.

Figure 6 compares the cost-effectiveness of mesh maintenance - with and without mesh augmentation. For a fixed d_{max}, the cost-effectiveness is higher with mesh augmentation than without mesh augmentation (comparing the two sub-figures), because they incur the same number of probes, but the former leads to more service mapping successes. Furthermore, with mesh augmentation (Figure 6(a)), the lower the d_{max}, the *higher* the cost-effectiveness. This is because a lower d_{max} means fewer probes; but the number of successes are almost the same for different d_{max}, as previously shown in Figure 5. As a comparison, we also show the cost-effectiveness of a complete-graph mesh, which with no surprise is the lowest.

In summary, the simulation results show that our mesh augmentation method significantly improves the performance of the service path finding algorithm. For the general service delivery overlays, we expect that the rich mesh connectivity will also lead to improved service mapping success rate.

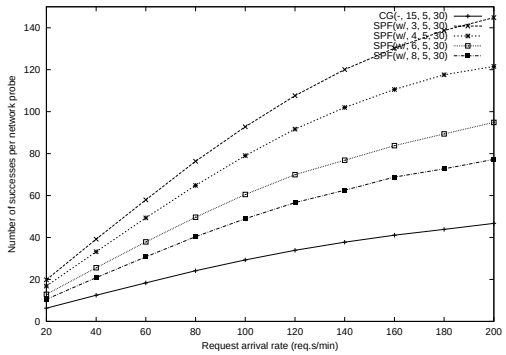
4.2.3 Analysis of Mesh Augmentation Method

Let N be the total number of MSODA nodes. For a given d_{max}, the number of non-neighbors of each node is $n = N - d_{max} - 1$. We assume that during each period T_q, the MSODA node probes a non-neighbor every fixed interval of $t_q = T_q/n$, and each non-neighbor is probed exactly once during each period T_q. The arrival of service path mapping requests follows Poisson distribution, with an average arrival rate λ_{req}. We assume that the P_s and P_d ($P_s \neq P_d$) of each service path request are both randomly selected from the MSODA overlay. The duration of each service session is an exponentially distributed random variable with a mean of $1/\mu$.

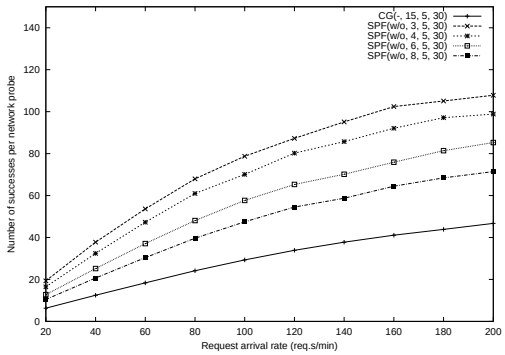
For each MSODA node, let $n(t)$ be the number of warm connections at time t. $n(t)$ may be any integer in interval $[0, n]$. Therefore, we have a state transition diagram for $n(t)$, as shown in Figure 7.

We first determine the transition rate μ_i in the diagram. For a warm connection, when the service delivery session that goes through it terminates, it becomes cold and the node loses a warm connection. In state $i+1$, there are $i+1$ warm connections, each with mean duration $1/\mu$. Therefore, the transition rate μ_i from state $i+1$ to i is:

$$\mu_i = (i+1) * \mu, (i = 0, 1, 2...n-1) \quad (1)$$



(a) With mesh augmentation



(b) Without mesh augmentation

Figure 6: Mesh maintenance cost-effectiveness measured by number of service mapping successes per network probe

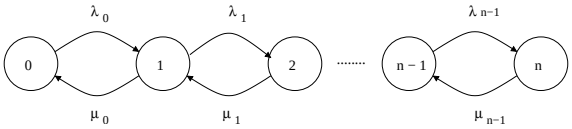


Figure 7: State transition diagram for the number of warm connections out of each MSODA node

We then determine the transition rate λ_i in the diagram. For the MSODA node in state i, in order to have one more warm connection, the following events should all happen: (1) A service path request arrives; (2) There is a non-neighbor probing result that has not yet expired - recall that a non-neighbor probing result expires after T_p (Section 4.1.2); (3) The connection from the MSODA node to the non-neighbor is currently cold; and (4) The connection is selected by the service mapping layer for this request, among $d_{max}+i+1$ connections, including d_{max} actively probed connections, i currently warm connections, and the cold connection itself. Therefore, the transition rate λ_i from state i to $i+1$ is:

$$\lambda_i = \frac{\lambda_{req}}{N} * min(\frac{T_p}{t_q}, 1.0) * \frac{n-i}{n} * \frac{1}{d_{max}+i+1}, (i = 0, 1...n-1) \quad (2)$$

The first factor in (2) is the request arrival rate at the MSODA node (recall that λ_{req} is the *overall* request arrival rate). The second factor is the probability that there is a not-yet-expired non-neighbor probing result. The third factor is the probability that the connection to this non-neighbor is *not* among the i warm connections. The last factor is the probability that the connection is selected by the algorithm for the request. For simplicity, we assume that the selection is random among the $d_{max}+i+1$ connections.

Let p_i ($0 \leq i \leq n$) be the steady state probability of state i. We will have the following balance equations for the state transition diagram:

$$\lambda_i * p_i = \mu_i * p_{i+1}, (i = 0, 1, 2...n-1) \quad (3)$$

Since $\sum_{i=0}^{n} p_i = 1.0$, we can solve p_i as:

$$p_0 = (1 + \frac{\lambda_0}{\mu_0} + \frac{\lambda_0\lambda_1}{\mu_0\mu_1} + ... + \frac{\lambda_0\lambda_1...\lambda_{n-1}}{\mu_0\mu_1...\mu_{n-1}})^{-1} \quad (4)$$

$$p_i = p_0 * \frac{\lambda_0...\lambda_{i-1}}{\mu_0...\mu_{i-1}}, (i = 1, 2...n) \quad (5)$$

Finally, the average number of warm connections out of each MSODA node can be computed as:

$$E(n(t)) = \sum_{i=0}^{n} i*p_i = \frac{1 * \frac{\lambda_0}{\mu_0} + 2 * \frac{\lambda_0\lambda_1}{\mu_0\mu_1} + ... + n * \frac{\lambda_0\lambda_1...\lambda_{n-1}}{\mu_0\mu_1...\mu_{n-1}}}{1 + \frac{\lambda_0}{\mu_0} + \frac{\lambda_0\lambda_1}{\mu_0\mu_1} + ... + \frac{\lambda_0\lambda_1...\lambda_{n-1}}{\mu_0\mu_1...\mu_{n-1}}} \quad (6)$$

Numerical results based on (6) match our simulation results reasonably well, although the analysis is more conservative estimating fewer warm connections per node. The reason is that we only consider the jumpstart of a cold connection by the MSODA node at the *starting end* of the cold connection. In a real-world MSODA overlay, since the probing results are also propagated to other nodes, the connection can be jumpstarted by *any* node that receives the results, leading to more warm connections in the mesh than what the analysis estimates.

5. RELATED WORK

Distributed software is evolving from monolithic applications to collections of open and flexible service components [15]. Based on service components provided by different parties across the Internet, new applications and services can be assembled rather than implemented from scratch. With the wide deployment of web services and the proposal of universal service description and discovery standards such as UDDI [2], application service composition and management is receiving increasing attention [14, 21].

A number of integrated frameworks for service composition have been proposed, such as Ninja [12], SAHARA [1], CANS [11], SPY-Net [28], and SpiderNet [13]. MSODA differs from these frameworks by introducing a virtualization-based service hosting platform that decouples service management and hosting platform management. As part of the Ninja project, the Automatic Path Creation (APC) service [23] supports automatic composition of application-level services. Each composed service is delivered by a service-level path along individual service providers. As part of the SAHARA project, service load balancing and stability issues in service path composition are studied in [24]. Recently, the concept of service multicast has also been proposed [4, 20], which involves the construction of a service delivery tree from the same media data source to multiple heterogeneous clients. MSODA aims at sup-

porting a more generic service delivery overlay, with its topology dynamically determined by the service configuration function.

Service overlay networks have recently attracted tremendous interests [10, 26, 27]. Overlay nodes are end systems instead of network routers, while overlay links are end-to-end transport-level connections between the overlay nodes. The function of overlays varies. For example, Narada [7, 8], Scattercast [6], and Overcast [16] are for application-level multicast and content distribution, while CFS [9], PAST [25], and OceanStore [22] are for wide-area networked storage. The overlay has also been proposed as a general infrastructure to support distributed applications running on top of it. Examples are the Resilient Overlay Network (RON) [3] and the Overlay Peer Utility Service (Opus) [5]. In Opus, distributed applications can be hosted in a subset of Opus nodes as application overlays. While sharing the same idea of "overlays on overlay", the MSODA architecture is virtualization-based and provides system support for service composition.

6. CONCLUSIONS

We have proposed the design of MSODA, an integrated multimedia service hosting overlay. Three key techniques in the development of MSODA are discussed: virtualization of media services, middleware support for service composition and configuration, and cost-effective monitoring of MSODA overlay for resource-aware service mapping. Our on-going work includes (1) the application of *virtual networking* techniques [19] to achieve isolation between service delivery overlays, (2) adaptive resource provisioning to service delivery overlays with *dynamic* load and topology, and (3) fully distributed service mapping *without* global system information.

7. REFERENCES

[1] The SAHARA Project: A Revolutionary Service Architecture for Future Telecommunications Systems. *http://sahara.cs.berkeley.edu/*.

[2] UDDI: Universal Description, Discovery and Integration. *http://www.uddi.org*.

[3] D. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris. Resilient Overlay Networks. *ACM SOSP 2001*, October 2001.

[4] S. Banerjee, Z. Xu, S. Lee, and C. Tang. Service Multicast for Media Distribution Networks . *IEEE Workshop on Internet Applications (WIAPP 2003)*, June 2003.

[5] R. Braynard, D. Kostic, A. Rodriguez, J. Chase, and A. Vahdat. Opus: an Overlay Peer Utility Service. *IEEE OPENARCH 2002)*, June 2002.

[6] Y. Chawathe. Scattercast: an Architecture for Internet Broadcast Distribution as an Infrastructure Service. *Ph.D. Thesis, University of California at Berkeley*, 2000.

[7] Y. Chu, S. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. *ACM SIGCOMM 2001*, August 2001.

[8] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. *ACM SIGMETRICS 2000*, June 2000.

[9] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica. Wide-Area Cooperative Storage with CFS. *ACM SOSP 2001*, October 2001.

[10] Z. Duan, Z. Zhang, and T. Hou. Service Overlay Networks: SLA, QoS, and Bandwidth Provisioning. *IEEE/ACM Trans. on Networking*, December 2003.

[11] X. Fu, W. Shi, A. Akkerman, and V. Karamcheti. CANS: Composable, Adaptive Network Services Infrastructure. *USENIX USITS 2001*, March 2001.

[12] S. Gribble, M. Welsh, R. von Behren, E. Brewer, D. Culler, N. Borisov, S. Czerwinski, R. Gummadi, J. Hill, A. Joseph, R. Katz, Z. Mao, S. Ross, and B. Zhao. The Ninja Architecture for Robust Internet-Scale Systems and Services. *Computer Networks, Special Issue on Pervasive Computing*, 2001.

[13] X. Gu, K. Nahrstedt, and B. Yu. SpiderNet: An Integrated Peer-to-Peer Service Composition Framework. *IEEE HPDC-13*, June 2004.

[14] R. Hauck. Architecture for an Automated Management Instrumentation of Component Based Applications. *IEEE/IFIP Int'l Workshop on Distributed Systems: Operations and Management*, October 2001.

[15] A. Ivan, J. Harman, M. Allen, and V. Karamcheti. Partitionable Services: A Framework for Seamlessly Adapting Distributed Applications to Heterogeneous Environments. *IEEE HPDC-11*, July 2002.

[16] J. Jannotti, D. Gifford, K. Johnson, F. Kaashoek, and J. J. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. *USENIX OSDI2000*, October 2000.

[17] X. Jiang and D. Xu. SODA: a Service-On-Demand Architecture for Application Service Hosting Utility Platforms. *IEEE HPDC-12*, June 2003.

[18] X. Jiang and D. Xu. vBET: a VM-Based Emulation Testbed. *ACM SIGCOMM Workshop on Models, Methods, and Tools for Reproducible Network Research*, August 2003.

[19] X. Jiang and D. Xu. VIOLIN: Virtual Internetworking on OverLay INfrastructure. *Department of Computer Sciences Technical Report CSD TR 03-027, Purdue University*, July 2003.

[20] J. Jin and K. Nahrstedt. On Construction of Service Multicast Trees. *IEEE ICC 2003*, May 2003.

[21] G. Kar and A. Keller. An Architecture for Managing Application Services over Global Networks. *IEEE INFOCOM 2001*, April 2001.

[22] J. Kubiatowicz, D. Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao. Oceanstore: An Architecture for Global-State Persistent Store. *ACM ASPLOS 2000*, November 2000.

[23] Z. Mao and R. H. Katz. Achieving Service Portability using Self-adaptive Data Paths. *IEEE Communications Magazine*, 40(1), 2002.

[24] B. Raman and R. Katz. Load Balancing and Stability Issues in Algorithms for Service Composition. *IEEE INFOCOM 2003*, April 2003.

[25] A. Rowstron and P. Druschel. Storage Management and Caching in PAST, a Large-Scale, Persistent Peer-to-Peer Storage Utility. *ACM SOSP 2001*, October 2001.

[26] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. OverQoS: An Overlay based Architecture for Enhancing Internet QoS. *USENIX NSDI 2004*, March 2004.

[27] S. Vieira and J. Liebeherr. Topology Design for Service Overlay Networks with Bandwidth Guarantees. *IEEE IWQoS 2004*, June 2004.

[28] D. Xu and K. Nahrstedt. Finding Service Paths in a Media Service Proxy Network. *SPIE/ACM MMCN 2002*, January 2002.