AN ARCHITECTURE FOR COMPONENTIZED, NETWORK-BASED MEDIA SERVICES

Michael Harville, Michele Covell, Susie Wee
Streaming Media Systems Group, HP Labs, Palo Alto, CA 94304 USA

ABSTRACT

We present MSA (Media Services Architecture), a flexible, general architecture for requesting, configuring, and running services that operate on streaming audio and video as it flows through the network. MSA decomposes requested media services into modular processing components that may be distributed to servers throughout the network and which intercommunicate via standard streaming protocols. Use of standard protocols also affords seamless inter-operability between MSA and media content delivery networks. MSA manages media services by monitoring the networked servers and assigning service components to them in a manner that uses available computational and network resources efficiently. We describe some implemented example services to illustrate the concepts and benefits of the architecture.

1. INTRODUCTION

One of the most significant trends in computing today is the migration of applications from local platforms into the network, in the form of componentized, web-based services. Using inter-operable, reconfigurable components delivered by the network, powerful services may be dynamically constructed and made available to end-users on demand, without their investing heavily in the underlying computational resources, software integration, and system administration. This model allows developers to easily maintain and rapidly improve their software, and it allows end-users to easily obtain the most current applications, or to switch application vendors entirely. Many major technology companies recognize the tremendous appeal of componentized, web-based services and have begun rebuilding their offerings in this form [1].

A second important trend is the increasing demand for digital rich media, such as audio, video, and images [2]. Digital media is easily shared between one's own devices, between different people, and between widespread or mobile locations. Streaming-media overlays are central to meeting media timing constraints. Mobile streaming media content delivery networks (MSM-CDNs) address scalability, user mobility, error-prone wireless channels, and real-time constraints [3]. In MSM-CDNs, self-managed "edge servers" work within today's IP infrastructure to ensure high-quality delivery performance through adaptive routing, caching, coding, and monitoring of media streams.

In this paper, we present a Media Services Architecture (MSA), that builds on both of these trends in computing. The MSA extends componentized, web-based services to the domain of streaming rich media by decomposing complex media services into flexibly configured, network-based parts. This approach allows rapid development and simple maintenance of powerful new applications, and promotes scalability to large numbers of users. All of this is achieved without sacrificing ease-of-use from the perspective of the media service clients.

2. NETWORK-BASED MEDIA SERVICES

Many types of analysis performed on audio, video, and other media in standalone systems today easily integrate into a networked-processing architecture. For example, speech recognition, face detection and recognition, and audio de-noising could simply move off the local desktop to networked server machines with available bandwidth and processing power. In addition, the MSA makes practical new, high-value services including:

Video compositing: Two or more video streams may be blended, image by image, according to masks to produce a single video stream with content from multiple sources. "Picture-in-picture" and "blue-screening" special effects are among the many applications. Video transcoding may be needed to overcome mismatched formats, resolutions, and frame rates of the input streams.

Meeting summarization and transcription: When cameras and microphones are present in a meeting, the incoming audio and video streams can be collected in the network and processed with video and audio segmentation and voice and face recognition to produce an indexed record of the meeting. This record can be used to quickly recall the meeting content at a later time.

Dynamic view selection: In live teleconferencing and webcast lecture applications, multiple cameras are often needed for adequate coverage. The best camera view typically changes many times during the event. Analysis of the video and audio streams from the event can be used by a network-based service to automatically select the best video feed.

These types of media analysis are available today through local desktop processing. However, componentized services operating on media streams in the middle of the network offer many advantages over the traditional desktop model, including:

- **Improved application offerings:** Developers can quickly distribute improved services by simply updating the MSA. New services are quickly created by mixing and matching components. Applications are available whenever users can reach the network, not just when they can access their own machines.
- **Reduced system administration:** Because processing is performed in the network, end users need not worry about continuous installation and update difficulties on their own machines.
- **Facilitation of multi-stream processing:** Many media-based applications, such as meeting summarization, require multiple streams to be gathered for joint processing. When these streams do not arise from the same machine, it is usually much more efficient to process them mid-network.
- **Controlled computational environment:** While individual users' machines may vary widely in their compute power, memory capacity, and operating systems, MSA machines can be standardized to a narrow range of specifications. Service components can be developed and optimized for these specifications, leading to more reliable overall application performance.
- **Efficient sharing of results:** In many situations, such as the meeting summarization context, the processed media and analysis results desired by multiple users are nearly the same or identical. Rather than duplicate this processing on each user's machine, mid-network processing can perform overlapping computations once only, and then distribute the results to each user.

In short, network-based media processing services offer users the potential of much greater flexibility and functionality than current, local, media-centric applications, with reduced maintenance and reliability concerns. In the next section, we describe how media services can be delivered in an efficient and scalable manner.

3. MEDIA SERVICES ARCHITECTURE

The key design goals of the MSA are 1) integrating with the media delivery architecture, and 2) enabling media services in a highly flexible manner. The main features of the MSA are:

- **Interoperability:** seamless streaming interconnections between components using open interfaces and standards
- **Modularity:** modular service components allowing dynamic media service construction in the middle of the network
- **Manageability:** efficient assignment of media services to computation and storage resources in a scalable manner

The means by which the architecture provides each of these features are discussed in the three subsections below.

3.1. Seamless Interconnects for Streaming Inter-Operability

All inter-machine transport of media streams within the MSA, as well as between elements of the MSA and components of media content delivery networks (CDNs), is conducted via uniform input and output modules that we have dubbed "Ears". The Ears rely on standards-based media streaming protocols, thereby easing integration of the MSA with CDNs and other streaming media applications. Both the input and output Ears communicate with other networked machines via 1) the SDP protocol for describing multimedia, 2) the RTSP protocol for session management and media playback control, and 3) the RTP/RTCP protocol for transport of data under real-time constraints. A given Ear manages one end (send or receive) of flow for a single media stream, but multiple Ears can be linked into the same, synchronized streaming session.

The Ears also provide data compression and decompression functionality, so that multimedia flowing through the architecture can be inter-converted between the compressed formats typically used for network transmission and the uncompressed format often demanded by media processing and analysis algorithms. Input Ears automatically detect the format of incoming streams and recruit the appropriate decompression module to convert the data into forms suitable for media analysis. Output Ears convert raw data streams into compressed formats suitable for network transport. Standard compression schemes supported include MPEG-1, -2, and -4 video and AMR and WAV audio.

Finally, because media processing algorithms may not operate at the same rate as the streaming media, the Ears implement data buffering and flow control methods to smooth data rate mismatches. Circular buffering minimizes expensive data copying, and multi-threading efficiently services data requests from the network, the application, and the (de)compression routines. Buffer overflow is handled by selectable policies for dropping frames.

3.2. Flexible, Modular Service Decomposition

An MSA service is initiated by contacting a Service Portal with a simple, high-level Media Service Request. These requests can be made directly by the user via the web, or they may be generated by applications run by the user either locally or within the MSA. Each Request contains the name of the service, such as "video compositing", along with any necessary service parameters, such as source and destination URLs.

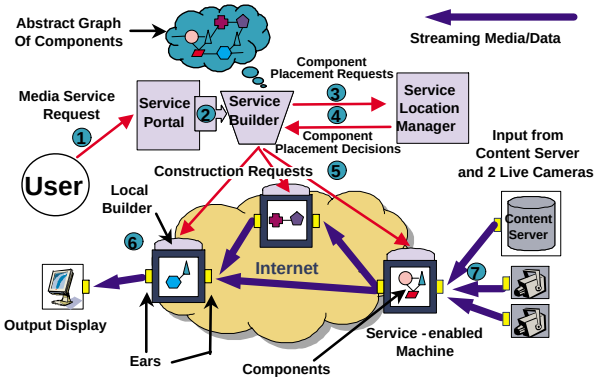

Fig. 1. Steps in starting a distributed media service: 1) User makes Media Service Request; 2) Service Portal runs Service Builder for named service; 3) Service Builder makes Component Placement Requests to SLM; 4) SLM returns Component Placement Decisions to Service Builder; 5) Construction Requests sent to Local Builders on selected service-enabled machines; 6) Local Builder instantiates and connects service Components; 7) media flows and service runs

These simple Media Service Requests hide the complexity of most media services. For example, meeting summarization can employ speech recognition, face detection, video motion analysis, and voice identification, and each of these component algorithms can, in turn, be divided into several sub-components. A given processing algorithm, on the other hand, may be a useful component in many different services. For these reasons, it is highly desirable to encapsulate media processing algorithms into modular, re-usable components that are flexibly and dynamically combined.

We therefore structure each media service as a graph of independent "Components" communicating through data streams. Each Component encapsulates one or more "Sub-Component" processing algorithms working tightly together. The Components for one media service can be dynamically placed on a single machine or distributed across the network. Since Components are well encapsulated, each operates without concern for this distribution.

The steps by which the MSA decomposes and distributes services are depicted in Figure 1. First, after receiving a Media Service Request, the Service Portal starts up a Service Builder to manage the Request's fulfillment. Each named media service is associated with a different Service Builder, and each Service Builder knows the structure of an abstract graph of Components that will implement that service. For each Component in this graph, the Service Builder sends a Component Placement Request to a "Service Location Manager" (SLM) to determine, as discussed in Section 3.3, the networked machine on which to run the Component. The SLM returns Component Placement Decisions, which include specific URLs (with port numbers) for each input and output stream of each Component, back to the Service Builder. The Service Builder groups these Decisions by selected machine, and sends one Construction Request, listing desired Components and their input and output URLs, to each machine.

A "Local Builder" runs on each MSA machine to service Construction Requests. For a given Request, the Local Builder creates each of the named Components, and uses the input and output URLs to instantiate Ears to send and receive data between these Components and those on other machines. The Local Builder also attempts to optimize each collection of inter-communicating Components running on a single machine, by eliminating identical Sub-Component processing done by more than one Component.

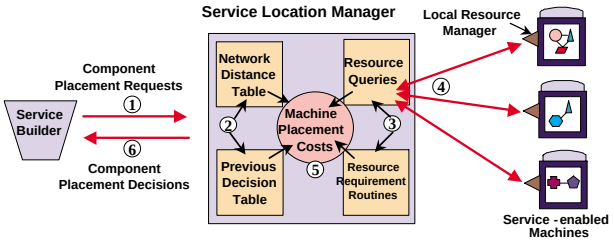

Fig. 2. Service location management steps: 1) Service Builder sends Component Placement Request to SLM; 2) SLM consults tables of network distances and previous decisions to select host pool; 3) SLM computes Component resource needs, and sends queries to host pool machines; 4) Local Resource Managers send back resource status and port numbers; 5) final Machine Placement Costs computed; 6) machine with lowest cost, and reserved ports and URLs, returned in Component Placement Decision

Such duplication sometimes occurs when services are divided into reasonably-sized, reusable Components. This cost of service modularity is thus mitigated by the Local Builder's optimizations.

After all Construction Requests are fulfilled, the service is ready to run. Components in the service graph closest to the data destination request media via an RTSP PLAY command, thereby pulling data through the entire graph of connected Components.

3.3. Dynamic Service Location Management

Many individual machines in the MSA network are capable of performing the underlying processing for media services. Therefore, for each Media Service Request, decisions must be made as to how to allocate MSA resources to best fulfill the request. To avoid unduly increasing the network load, these decisions are based in part on the (network) proximity of various service-enabled machines to good paths between sources and destinations of the media streams. To provide services with minimal delay and highest quality, these decisions also take into account the current processing load carried by each MSA media processor. Finally, when some Components of a service share Sub-Component processing, it may be preferable to group them on the same machine.

To make these decisions intelligently, we use "service location management" (see [4]). The MSA contains Service Location Managers (SLMs) that determine where to place the individual Components that comprise a service. For a given Media Service Request, an SLM places Components of the service one at a time in an order defined by the associated Service Builder. Placement Decisions for Components are not made simultaneously, through joint optimization over all factors and all Components, as this is likely to be a complex, time-consuming procedure for even moderately sophisticated services. However, Placement Decisions for different Components are not entirely independent, as this could lead to inefficient data paths and duplicate Sub-Component processing. Instead, SLMs maintain tables of recent Component Placement Decisions, and base each new decision in part on this history.

Our service location management methodology is shown in Figure 2. For each Component Placement Request, the SLM first selects a pool of potential host machines, based on network locality and previous Component Placement Decisions. To assess the former, the SLM consults a table of network "distances" between server machines, to determine which machines are near the service data sources and destinations, or the path(s) between them. The table distances are determined by measured network delays and bandwidths. Machines on which other Components of the service have previously been placed may be given greater preference by the SLM for placement of the current Component, particularly if those previously placed Components are to be connected directly to, or are indicated to potentially share Sub-Component processing with, the current one. All of this information is combined into calculating "Machine Placement Costs" for each potential host.

The required computational and memory resources of the Component are determined by the SLM by supplying service parameters, such as media resolution and frame rate, to a Resource Requirement Routine associated with that type of Component. Resource availability on potential hosts is determined by the SLM through consultation of Local Resource Managers (LRMs) resident on those machines. Each LRM monitors that machine's state by direct inquiry to the operating system. LRMs also track pending and recently fulfilled requests from the machine's Local Builder, as these may not yet be reflected in current processor load statistics. Each LRM returns the machine status back to the SLM, along with network port numbers reserved for use by the Component if it is placed there. The SLM increments all Machine Placement Costs in inverse proportion to the machine's resource availability.

The machine with the lowest Placement Cost is selected as the Component host. A Component Placement Decision, specifying this host and containing Component input and output URLs and reserved ports, is returned by the SLM to the Service Builder. The SLM's table of recent Placement Decisions is also updated.

4. EXPERIMENTAL RESULTS

We have implemented a prototype of the MSA, along with Components from which a variety of services may be built. To better illustrate the operation and benefits of the MSA, we discuss services supported by three Components operating on video media:

- **Resizing:** Changes the width and/or height of the video; for instance, a high-resolution video may be downsampled for better transmission and display on a PDA.
- **Background Removal:** Extracts the dynamic or "interesting" objects in a scene, such as people, while suppressing other, unchanging aspects of the scene, such as walls and furniture. Our Background Removal Component is based on the method of [5]. It attempts to replace background in a scene with a constant color (such as white), while leaving the foreground unchanged.
- **Compositing:** Uses a mask to replace pixels in a video stream with pixels from another image or video stream. Our Compositing Component replaces video stream pixels having a special color (such as white) with pixels from another image or stream, while leaving the other pixels unchanged.

Transcoding of video to lower resolutions suitable for mobile clients, such as PDAs and cellphones, is important to modern CDN design [6, 7], and can be achieved via the Resizing Component. By combining this with Background Removal, "intelligent compression" services for even lower target bit-rates, without loss of the most "interesting", foreground part of the video, may be provided. We focus here on a "Mobile Private Video Phone" (MPVP) service that uses all three of the above Components. MPVP allows video teleconferencers to prevent others from seeing the details of their surroundings, by using Compositing to replace their background with an image or video of their choice. For instance, a person calling from the beach may prefer to use a background image of his office. For users receiving video on mobile devices, downsampling (via Resizing) is also used, for bit-rate reduction.

The MPVP service may be started within an IP telephony application that has already opened an audio channel to a remote

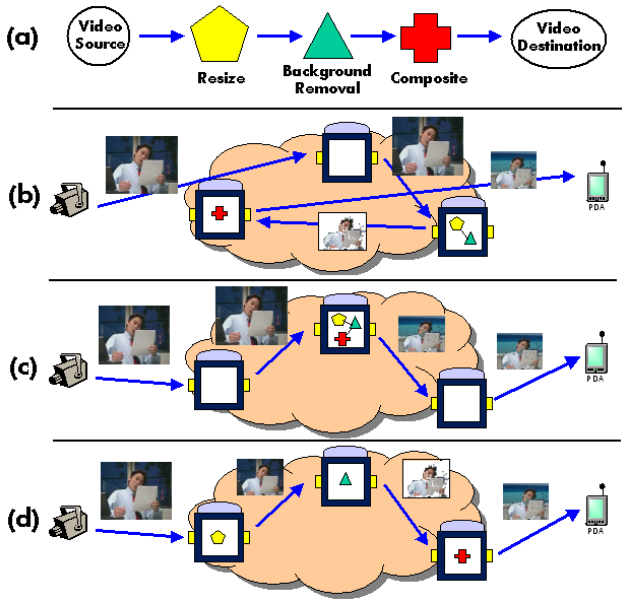

Fig. 3. Mobile Private Video Phone service results, where a person in an office wishes to appear as if he is at the beach. **a)** Abstract graph of Components describing service; **b) - d)** Three possible distributions of Components on a small network. The three servers and the video source and destination are arranged to reflect their relative network distances. Images represent the (possibly processed) video flowing on each link. Machines with no processing Components simply forward the media.

participant, and now wants to add video. The application sends a request for the "MPVP" service, along with parameters such as the destination IP address and desired video resolution, to an MSA Service Portal. The Portal starts the MPVP Service Builder, which knows the abstract graph for the service (Figure 3**a**): Resizing feeding into Background Removal feeding into Compositing.

The Service Builder sends Component Placement Requests for each of the three Components, in the order they appear in the abstract graph, to an SLM. For illustration, we assume the network contains three service-enabled machines on which the SLM can place Components. Also, the SLM knows how much computation can be reduced if two or more of the Components are placed on the same machine. The SLM considers the potential computation savings, the current computational load levels on each machine, the processing requirements of each Component, and the network topology and load levels, in order to arrive at a decision as to how to distribute the Components. Three possible distributions are shown in Figure 3**b-d**.

The first distribution (**b**) is not favored by our SLM because its long data path will result in high latency for the service. Such a distribution might be selected by simpler placement algorithms that do not account for network topology and placement history. The second configuration (**c**) places all Components on the same machine. This results in computational savings not just through elimination of redundant Sub-Component processing, but also by removing extra video decompression and compression steps, performed by the Ears, that would be needed if the Components were on separate machines. Configuration **(c)** thus greatly reduces the overall computational load introduced to the service network, and may be preferred when system load levels are high, as when many services are in progress. A disadvantage of placing all Components on one machine, however, is that their combined processing is less likely to keep up with the frame rate of the streaming video. For instance, it may be difficult to do Resizing, Background Removal, and Compositing all on the same machine at 30 frames/sec, so that some frames need to be dropped and the resultant video quality diminishes. By spreading the Components across three different machines, on the other hand, as in Figure 3**d**, all three Components are more likely to run smoothly, without dropped frames, at 30 frames/second, particularly if these machines were selected because they were relatively unloaded.

The Placement Decisions made by the SLM are returned to the Service Builder, which groups them by machine and sends out Construction Requests to the Local Builders resident on those machines. The Local Builders start up the requested Components, and direct them to send and receive data according to the URLs specified in the Construction Requests. When all Local Builders have notified the Service Builder that their Components are ready, media flow through the service is started via an RTSP "PLAY" command. The images on the links between machines in Figure 3 show examples of the processing done to a real video stream as it flowed through the various service topologies.

This service example illustrates some basic aspects of the MSA. Clearly, this approach can be extended to incorporate additional types of Component processing, as well as branching of processed streams to multiple users, each of whom may request different, further processing along his own branch. Also, many of our other service Components may employ video and audio analysis to produce non-media data streams such as text (e.g. from speech recognition) or event summaries and time indices (e.g. from vision-based person tracking and activity analysis).

5. CONCLUSIONS

Many advanced algorithms in video and audio analysis and processing have yet to make their way into widely-used applications. This is due, in part, to the difficulties of configuring complex media processing applications, in obtaining the substantial processing resources they often require, and in connecting these applications to interesting sources of media and desirable output locations. By enabling flexible media processing that lives in the network itself, we believe our Media Services Architecture has the potential to bring advanced, media-rich applications into mainstream, widespread usage. This architecture integrates easily with media CDNs, allows for modularity of services for easy reconfiguration and reuse, and promotes efficient allocation of scarce network resources, while reducing maintenance, compatibility, and availability issues for end-users.

6. REFERENCES

[1] Intel, "Media Processing for the Modular Network." see http://www.intel.com/network/csp/resources/white_papers/7786web.htm.
[2] H. Houh, J. Adam, M. Ismert, C. Lindblad, and D. L. Tennenhouse, "VuNet Desk Area Network: Architecture, Implementation, and Experience," *IEEE Journal of Selected Areas in Communications*, 1995.
[3] S. Wee, J. Apostolopoulos, W. Tan, and S. Roy, "Research and design of mobile streaming media content delivery network," in *ICME*, 2003.
[4] S. Roy, M. Covell, J. Ankcorn, S. Wee, M. Etoh, and T. Yoshimura, "A system architecture for mobile streaming media services," in *Intl. Wksp. on Mobile Distrib. Computing*, May 2003.
[5] M. Harville, G. Gordon, and J. Woodfill, "Adaptive background subtraction using color and depth," in *ICIP*, Oct 2002.
[6] H. Sun, W. Kwok, and J. Zdepski, "Architectures for MPEG compressed bitstream scaling," *IEEE Trans. Circuits and Sys. for Video Tech.*, vol. 6, April 1996.
[7] S. Wee, B. Shen, and J. Apostolopoulos, "Compressed-domain video processing," *HP Labs Tech Report*, October 2002.