Management of Real-Time Streaming Data Grid Services

Geoffrey Fox, Galip Aydin, Harshawardhan Gadgil, Shrideep Pallickara, Marlon Pierce, and Wenjun Wu

Community Grids Laboratory
Indiana University
501 North Morton Street, Suite 224
Bloomington, IN 47404
{gcf,gaydin,hgadgil,spallick,marpierc,wewu}@indiana.edu

Abstract: We discuss the architectural and management support of real time data streams, both in terms of lower level messaging and higher level service, filter and session structure. Our messaging system acts as a substrate that can be used to provide qualities of service to various streaming applications ranging from audio-video collaboration systems to sensor grids. The messaging substrates are composed of distributed, hierarchically arranged message broker networks. We discuss the role of management systems for both broker networks and filters: broker network topologies must be created and maintained, and distributed filters must be arranged in appropriate sequences. These managed broker networks may be applied to a wide range of problems. We discuss applications to audio/video collaboration in some detail and also describe applications to streaming Global Positioning System data streams. These provide specific application filters that can transform and republish message streams to the broker system.

1. Introduction

A growing number of applications involve real-time streams of information that need to be transported in a dynamic, high-performance, reliable, and secure fashion. Examples include sensor nets for both science and the military applications, mobile devices on ad-hoc networks, and collaborative applications. In the latter case the streams consist of a set of “change events” for a collaborative entity multicast to the participating clients. They could be the frames of audio-video streams, encoded changed pixels in a shared display, or high level semantic events such as signals of PowerPoint slide changes. Here we describe our research into ways of managing such streams, which we think are a critical component of both sensor nets and real time synchronous collaboration environments.

We develop real-time streaming technology assuming that the sources, sinks, and filters of these streams are Web or Grid services. This allows us to share the support technology between streaming applications and benefit from the pervasive interoperability of a service oriented architecture. Further, this allows a simple model of collaborative Web and Grid services gotten by “just” sharing the input or output ports. As services expose their change by explicit messages (using what we call a message-based Model-View-Controller architecture [1]), it is much easier to make them collaborative than traditional desktop applications whose change is often buried in the application. We have shown how traditional collaborative applications can be made service oriented with in particular a set of services implementing traditional H.323 functionality and interoperating with Access Grid and Polycom systems. This required development of an XML equivalent of the H.323 protocol [2, 3]. Our other major motivation is the sensor networks of military, scientific and social infrastructure. These are well suited to a service architecture as exemplified by the US military Global Information Grid with its service-based NCOW (Network Centric Operations and Warfare Architecture) [4, 5, 6, 7].

We have developed general purpose, open source software to support distributed streams, described in Sec. 2. NaradaBrokering [8, 9] forms a distributed set of message brokers that implement a publish-subscribe software overlay network. This environment supports multiple protocols (including UDP, TCP, and parallel TCP) and provides reliable message delivery with a scalable architecture. The infrastructure can supply message oriented middleware support for Web services with support of WS-Eventing, WS-Notification, WS-Reliable Messaging, and WS-Reliability. We note the architecture links both Peer-to-Peer and Grid paradigms and has been used in either mode although the Peer-to-Peer experience is limited.

We have used several applications that drive the development of our technology. These include collaboration services with audio, video, and shared display streams, as well as linkages of real-time Global Positioning System sensors to Geographical Information Systems implemented as Web services. Other examples include integration of hand-held devices to a Grid [10] and the linkage of annotations to video streams showing how composite streams can be supported for real-time annotation [11]. The first two applications are described in sections 4 and 5 and illustrate the need the high level session and filter infrastructure on top of the messaging infrastructure

There are several technical challenges to building infrastructure that is efficient and can support time-sensitive operations such as the instant replay and annotation of sensor streams in an emergency. One needs to archive events (such as video frames) to support the replay. The location of replicated (for fault tolerance) archives and the linkage to annotation material illustrate system metadata which is very dynamic but must be accessible to the system with low latency. This requires metadata approaches that we discuss separately [12] optimized for dynamic responsiveness in modest sized subsystems rather that approaches like distributed hash tables optimized for scalability.

Our architecture supports the interesting concept of hybrid streams where multiple “simple streams” are intrinsically linked; examples are linkage of a stream of annotation white boards with original audio/video stream [11] and the combination of lossless and lossy codec streams (using perhaps parallel TCP and UDP respectively) to represent a large dynamic shared display.

The messaging infrastructure supports the application services with their filters, gateways and sessions reflecting both the collaborative and workflow functions.

However we have found the need for a set of services that manage the messaging itself and so control broker deployment and quality of service. This is discussed in section 3 which describes the integration of the management of messaging and higher-level services.

2. NaradaBrokering: a Distributed Messaging Substrate

NaradaBrokering [8, 9, 13] is a messaging infrastructure that is based on the publish/subscribe paradigm. The system efficiently routes messages [14] from the originators to the consumers that are interested in the message. The system places no restrictions on the size and the rate at which these messages are issued. Consumers can express their interests (or specify subscriptions) using simple formats such as character strings. Subscriptions may also be based on sophisticated queries involving XPath, SQL, or regular expressions. Support for these subscription formats enables consumers to precisely narrow the type of messages that they are interested in. The substrate incorporates support for enterprise messaging specifications such as the Java Message Service [15]. The substrate also incorporates support for a very wide array of transports (TCP, UDP, Multicast, SSL, HTTP and ParallelTCP among others), which enable the infrastructure to be leveraged by entities in a wide variety of settings. To cope with very large payloads the system leverages ParallelTCP at the transport level and services such as compression and fragmentation to reduce individual message sizes. The fragments (compressed or otherwise) are reconstituted by appropriate services (coalescing and de-compression) prior to delivery to the application.

The most fundamental unit in NaradaBrokering is a message. A stream can be thought of as being composed by a series of messages, each with causal and ordering correlations to previous messages in the stream. The inter-broker latency for routing typical messages is around 1 millisecond. In a controlled cluster setting a single broker was found to support up to 400 UDP-based A/V clients concurrently with adequate latency [16]. Among the services most relevant for collaboration within the system are the following.

1. Support for a replay and recording services: The recording service is used to store messages reliably to the archival system. The recording is done in such a way that all events issued by the recording entity are stored in the order that they were published. The replay service facilitates the replay of these previously stored messages. The replay service support replays in multiple flavors. Entities may request replays based on sequencing information, timing information, content of the message or based on the topics that these messages were published to. In some cases one or more of the parameters can be combined in a single request.
2. Support for consistent global timestamps [17] through an implementation of the Network Time Protocol (NTP). This implementation ensures that timestamps at the distributed entities are within a few milliseconds of each other. This allows us to ensure that we can order messages based on these global timestamps. This is especially useful during replays when we can precisely determine the order in which messages should be released to the application.

3. Support for buffering and subsequent time-spaced release of messages to reduce jitters. The typical lower bound for time space resolution is a millisecond. However, we have also been able to successively time-space events in the order of several microseconds. By buffering and releasing messages we reduce the jitters that may have been introduced by the network.

More recently, we have incorporated support for Web Services within the substrate. Entities can send SOAP messages directly to the brokers that are part of the messaging infrastructure. We have incorporated support for Web Service specifications such as WS-Eventing [18], WS-ReliableMessaging [19] and WS-Reliability [20]. Work on implementing the WS-Notification [21] suite of specifications is currently underway. The implementation of these specifications also had to cope with other specifications such as WS-Addressing and WS-Policy that are leveraged by these applications. In addition to the rules governing SOAP messages and the implemented protocols, rules governing WS-Addressing were also enforced.

In our support for SOAP within NaradaBrokering we have introduced Filters and Filter-Pipelines (Figure 1). A filter is smallest processing unit for a SOAP message. Several Filters can be cascaded together to constitute a Filter-Pipeline. Here, the Filters within a Filter-Pipeline can be dynamically shuffled and reorganized. The system allows a Filter-Pipeline to be registered for every role that the node (functioning as a SOAP intermediary) intends to perform.

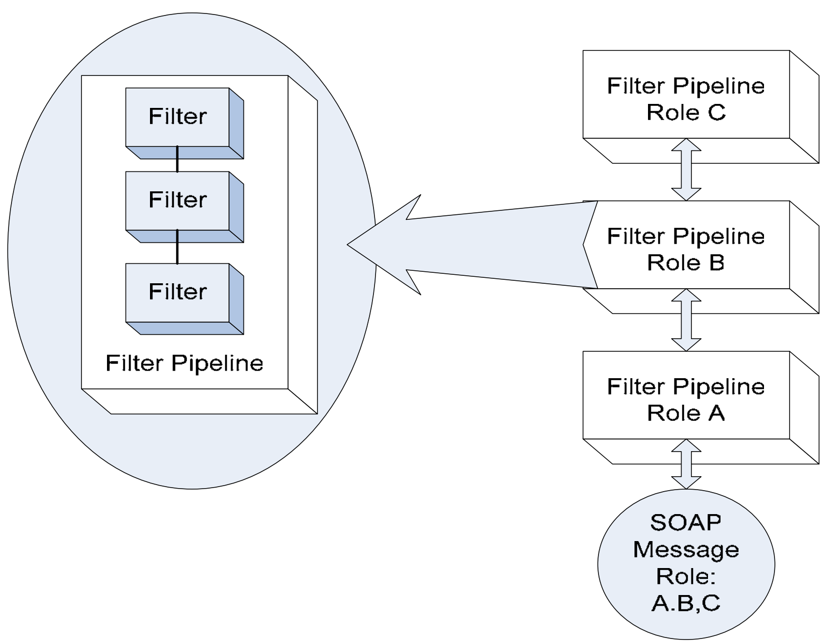

Figure 1 Filter pipelines are used to process SOAP messages in NaradaBrokering.

Upon receipt of a SOAP message that is targeted to multiple roles (as indicated by the SOAP 1.2 role attribute) the corresponding Filter-Pipelines are cascaded so that the appropriate functions are performed. The SOAP message is first parsed to determine the roles that need to be performed. Next, we check to see if there are any pipelines registered for a specific role. The scheme allows developers to develop their own Filters and Filter-Pipelines and target them for specialized roles. For e.g. in some cases a developer may wish to develop a Filter that performs message transformations between the competing notification specifications: WS-Eventing and WS-Notification. By providing an extensible framework for the creation of Filters and the registration of roles sophisticated applications can be built.

3. HPSearch: Managing Broker Networks and Service Grids

As discussed in the previous section, NaradaBrokering provides a software messaging infrastructure. In a related project, we have been developing HPSearch [22] as a scripting based management console for broker networks and their services. At one end of the spectrum are services which help manage the messaging middleware, while at the other end are services that leverage capabilities of the middleware (WSProxy). The management of both sets of services is handled by a scripting medium that binds Uniform Resource Identifiers (URI) to the scripting language. By binding URI as a first-class object we can use the scripting language to manage the resource identified by the URI. We discuss these functions in detail below.

Management of messaging middleware

In order to deploy a distributed application that uses NaradaBrokering, the middleware must be setup and a broker network topology must be deployed. Broker network topology may also be changed at runtime using HPSearch by adding or deleting links between brokers. Once the middleware is setup, we leverage the broker network to deploy the distributed application as detailed in the next section.

To fulfill this requirement we have been developing a specialized Web Service called the Broker Service Adapter (BSA) that helps us deploy brokers on distributed nodes and setup links between them. The BSA is a Web Service that enables management of the middleware via WS-Management [23]. Further, the BSA network is a scalable network that periodically restructures itself to achieve a tree based structure. A management engine simply sends the appropriate commands to the root BSA node which is then appropriately routed to the correct BSA. Errors and other conditions are similarly handled and notified to the management engine using WS-Eventing [18].

Management of Data Streams and Services

HPSearch uses NaradaBrokering to route data between components of a distributed application. This data transfer is managed transparently by the HPSearch runtime component, the Web Service Proxy (WSProxy) [24]. Thus, each of the distributed components is exposed as a Web Service which can be initialized and steered by simple SOAP requests. WSProxy can either wrap existing applications or create new data processing and data filtering services. WSProxy handles streaming data transfer using NaradaBrokering on behalf of the services thus enabling streaming data transfer for any service. The streaming data is enabled using NaradaBrokering middleware, a distributed routing substrate. Thus there are no central bottlenecks and failure of a broker node routes the data stream through alternate routes if available. Further, NaradaBrokering supports reliable delivery via persistent storage [25] thus enabling guaranteed delivery for data streams.

All interactions with the WSProxy are made using simple SOAP calls, and thus the service wrapped by the WSProxy can be steered using any client that can talk SOAP. The WSProxy service has been written using Apache Axis and can be deployed in any Web Service container such as Apache Tomcat.

HPSearch adds initialization and steering of WSProxy based Web Services using a scripting interface. HPSearch creates temporary topics using the NaradaBrokering system, and correctly ties up all components thus establishing a data flow between various components of the distributed application. Reference [26] presents more details on how a distributed application can be modeled using WSProxy and deployed and managed using the HPSearch system.

4. Global-MMCS: Audio and Video Stream Services and Management

Global-MMCS, as a service-oriented multimedia collaboration system, mainly processes multimedia streams: video, audio, whiteboard and so on. “Events” in video or audio are usually called video frames or audio samples. Generally speaking, there are a lot of similarities between multimedia streams and other data streams such as sensor data. All streaming data require significant Quality of Service (QoS) constraints and dynamic filtering. These are both particularly demanding and well-understood for multimedia streams for both communication and processing. Because of high bandwidth generated by raw multimedia bit-streams, complicated codecs must be used to compress the streams and transmit them over Internet. Further, multimedia streams are typically used collaboratively and so stress the infrastructure needed to support the efficient software or hardware of multicasting required by the delivery of a given stream to multiple clients. Due to the diversity of collaboration clients supported by Global-MMCs, the services for multimedia streams need to adapt the streams to different clients. We note that many relevant web service specifications like those for reliable messaging and notification appear not well designed for scalable efficient multicast as needed by Global-MMCS. Thus we suggest that multimedia collaboration is an excellent proving ground for general streaming data grid infrastructure.

A media service or filter is a functional entity, which can receive one or multiple media streams, perform some processing, and output one or multiple media streams. Each service is characterized by a set of input and output stream interfaces and a processing unit. According to the number of fan-in and fan-out of filters, they can be divided into three categories: one-in-one-out filters, multiple-in-one out filters, and one-in-multipe-out. In addition, there is a final “sink” filter category. We discuss each of these below.

One-in-One-Out Filters

Such a filter implements the basic transformation operation. For instance, a filter can receive as input a video stream in YUV4:1:1 format, resize it and deliver the modified video as output. Each filter provides a very basic adaptation on a stream in an intermediate format. Complex stream transformations can be built by combining several basic filters and creating a filtering workflow pipeline. Below are examples of one-in-one-out filters:

Decoder/Encoder transcoder filters aim at compressing/uncompressing the data into a chosen intermediate format (e.g. RGB24, YUV4:1:1, Linear Audio). Common codecs include H.261, H.263, MPEG1, MPEG2, MPEG4, H.264, and RealMedia. Transcoding generates a new stream which is encoded in the format wanted by the user. For examples, if a RealPlayer user needs to receive a video encoded in H.261 RTP, a RealStream producer is needed to first decode the H.261 video and generate a new RealFormat stream. ***Image-scaling filters*** resize video frames, which is useful to adapt a stream for devices with limited display capacities. They are sometimes required to enable transcoding operations, for example MPEG videos may be transmitted in any size while H.261 videos require predefined sizes such as CIF, QCIF or SQCIF. ***Color-space-scaling filters*** reduce the number of entries in the color space, for example from 24 to 12 bits, gray-scale or black-and-white. ***Frame-rate filters*** can reduce the frame rate in a video stream to meet low-end clients like PDA. For example, we can discard B-frame or P-frame in a MPEG-4 video stream with 24 fps to create a new stream with a lower frame rate.

Multiple-in-One-Out Filters

Mixer Filters combine multiple streams. A video mixer can create a mixed video streams resulting from several input sources. Each element of the resulting mixed video (typically displayed as a grid of images) results from an image-scaling adaptation of a particular stream. An audio mixer can create a mixed audio stream by summing up several input sources. Audio mixing is very important to those clients that can’t receive multiple RTP audio streams and mix them. Video mixing service improves the visual collaboration especially for those limited clients, which can only handle a single video stream. ***Multiplexors / Demultiplexors*** are used to aggregate/separate audio and video data in a multimedia stream. For instance, an MPEG multiplexor allows merging an MP3 audio and an MPEG-1 video in a MPEG2 stream. Multiplex and demultiplex are quite useful for guaranteeing stream synchronization in unpredictable network environments.

One-in-Multiple-Out Filters

Duplicator filters are used to replicate an output media stream. Duplication is useful when a stream has different targets with different requirements. In most cases, multiple simple media filters should be organized in a media filter chain. Filters can be either as simple as bit-stream parsing, or as complicated as decoding and encoding. Composite media services are usually acyclic computation graph consisting of multiple filter chains.

Sink Filter Services

There is also another type of bit-stream service, called sink service, which doesn't change bits in the stream. Examples of sink services include buffering and replaying services. These can buffer real-time multimedia streams in memory caches or disk storage, and allow users to reply or fast-forward these streams through RTSP session. Sink filters can handle single or multiple streams. When multiple streams flow into a sink entity, all the streams can be synchronized and replayed. Based on such a composite sink service, an annotation service can be developed. Through annotation, users can attach text and image streams to the original video and audio stream to convey additional meaning in collaboration. Stream annotation is discussed in [11].

GlobalMMCS Architecture

Figure 2 shows our architecture for managing streaming services and their workflow. It is built around NaradaBrokering which offers a powerful RTP event (message) delivery, which is quite critical to multimedia streaming. We have developed XGSP [27] as the framework to specify stream schema and offer offer support for sessions, end-points, filters, replay collaboration and their integration in streaming workflow. The stream schema use a similar syntax to SMIL [28], describing the source (URI), the format and QoS requirement of each stream. The function of filters can be defined by specifying the input and output stream streams for them. The whole workflow is a collection of filter chains and the available streams.

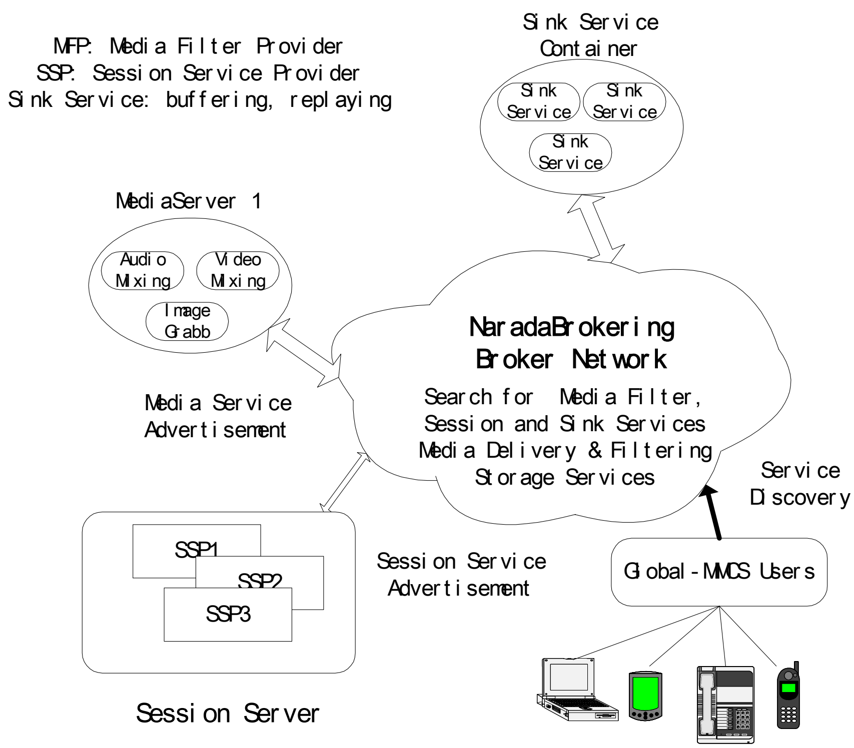

Figure 2. Global-MMCS Streaming Workflow Management

Media Service and Workflow

There is substantial literature on Grid and Service-based workflow [29, 30]. Unlike many of these systems, Global-MMCS's streaming workflow, especially conferencing workflow, is implicit and can be determined by the system at run time based on the specified (in XGSP) sinks and sources and their QoS. For example, when a PDA with limited network and processing capability wants to receive an H.261 encoded, 24 fps, CIF video stream, a customized workflow is need to transcode the H.261 stream to a JPEG picture stream or low-bitrate RealMedia Stream. An intelligent workflow engine can easily build a filter chain automatically based on the format description of the source stream and capability description of the PDA. Such an engine usually follows a graph search algorithm and tries to find a route from the graph node representing the format of the source stream to the destination node representing the format needed by the receiver.

No user involvement is needed for defining explicit workflow. Furthermore, in order to minimize the traffic and delay, most of one-in-one-out filter chain should be constrained in a single service container. One needs a distributed implementation to orchestrate multiple-in and multiple-out filters for different clients. Therefore the key issue in Global-MMCS media service management is how to locate the best service container based on streaming QoS requirement and make the service provider shared by participants in XGSP Sessions.

Computation and storage resources connected with NaradaBrokering brokers are service containers that can host both media processing and session management services. The XGSP framework specifies the XML scheme for describing the media processing and session management services. Each service provider can advertise its service XML description to distributed service registries such as we discuss in more detail in a companion paper [12].

Each broker may have a registered media service container called MediaServer, which hosts various computationally intensive media processing services. All service providers implement the interface to be able to run inside the service container. MediaServer can create, start and stop media service instances. We are currently investigating adding support for UDDI to allow the MediaServer to advertise these service providers and reports the status information to the distributed metadata registry regarding the load on that machine. XGSP audiovisual session servers can locate the best container and request a service instance to execute in the container.

Global-MMCS Meta Data Management

We need to define collaborative sessions describing both the the group of people and their clients as well as the associated media services. XGSP audiovisual sessions have five states: created, canceled, activated, deactivated and finished. The XGSP AV session service manages these states. A XGSP user can initiate an audio-visual session while the session server can activate this created session at the meeting time after the needed service instances are created. The meta-data or "context" for the session has both static and dynamic parts.

Static Metadata**:** Collaboration users need to know how many active sessions are available and their associated detailed information. The conference announcement can be implemented either in the XGSP session protocol or through the out-of-band communication. The XGSP framework divides the conference advertisement information into two levels: one is the collaborative conference calendar, which describes high-level meta-data about the organization of the conference including meeting time and topic. The other is the detailed information needed by audiovisual clients to join the conference as for example the session identification in the system and transport addresses associated with the session. The high-level conference calendar is implemented as a web-services running in the XGSP web server. Each active entry in the calendar has a link to the detailed session description.

Real-Time Metadata**:** As well as the static metadata described above, a XGSP session has much real-time context. There are three important entities in a XGSP session: participants, streams and services or filters. For each type of entity, a dynamic list has to be maintained. A participant list should keep the ID of each joined user and its multimedia capability and preference. A service or filter list should keep track of the activated services and their load. The stream list is more comples as it must keep track of source streams and filtered streams such as duplicated streams, transcoded streams and mixed streams. For buffering and annotated streams, it also keeps the description, of linkage in hybrid streams (see Sec. 1) and how they are stored. The latter is particularly critical for real-time replay [11].

For a real-time conferencing application, static and high-level meta-data should be organized using standard calendar models and allows this conference calendar service to interact with other public, group or private calendars. Since WS-Context can manage session metadata between multiple participants in Web-Service interactions, real-time metadata of XGSP sessions may also be managed by WS-Context style service and implemented in an efficient manner [12].

Global-MMCS Session and Workflow Management

NaradaBrokering can publish performance monitoring data in the form of XML on a topic which is subscribed to by the AV Session Server. From these performance data and broker network maps, the Session Server can estimate the delay and bandwidth between the service candidates and the requesting user. Based on the workload of the media service providers and estimated the performance metrics, the Session Server can find the best service providers and initiate a media service instance. Furthermore, the AV Session Server has to monitor the health of each media service instance. Through a specific NaradaBrokering topic, an active media service instance can publish status meta-data to notify the session server. If it fails to respond within a period of time, the AV Session Server is responsible to restart it or locate a new service provider and start a new instance. Note that the messaging infrastructure supports both TCP control and UDP media streams and their reliable delivery; the session can choose separate QoS for each type of stream.

Each session server may host limited numbers of active XGSP AV sessions. The exact number depends upon the workload and the computational power of the machine. The session initiator will firstly locate the right session provider to create a session service instance for a particular XGSP AV session. Then, this session server will locate the necessary media service resources on demand. In the current implementation, a default audio mixer is created to handle all the audio in the session. Private audio mixers can be created on-demand for private sessions supporting subgroups in the session. Further, multiple video mixers can be created by the session server on the request of the client. An image grabber (thumbnail) service is created when a new video stream is detected in the session. Further, customized transcoding services can be created when a user sends a request to access particular streams. For example, a mobile client like PDA connected to Wi-Fi, which only has limited processing power wants to choose a 24 4-CIF MPEG-4 video; then a transcoding process pipeline consisting of frame rate adapter , video size downsampler and color transformation, is needed to create this stream. Another example is an H.323 terminal, which can only handle H.261 and H.263 codecs, wants to display a MPEG-4 video, it will ask the session server to start a MPEG-4-to-H.261 transcoder.

Sink services like buffering, archiving and replaying services can also be initiated by real-time XGSP sessions. Buffering and archiving services store events into distributed cache and file storage attached to NaradaBrokering overlay networks. Once stream data flow into these "sinks", replaying service can pull the data flow out of the sinks and send to clients based on the RTSP request of the user. The events are accessed in an ordered fashion and resynchronized using their timestamps which have beed unfied using NaradaBrokers NTP service. The list with time-stamps of these archived and annotated streams is kept in the WS-Context dynamic meta-data service. Through the recording manager service, a component of AV session server, users can choose streams to be buffered and archived. And through replay and RTSP servicesr, users can initiate RTSP sessions and replay those buffered streams. After the streams are buffered, users can add annotations to the streams and archive the new composite steams for later replay.

5. Supporting Real Time Sensor Grid Services

The basic services needed to support audio-video collaboration, such as reliable delivery, multicasting and replay, can also be applied to problems in real-time delivery of sensor grid data. In Figure 3, we depict our work to develop filters on live Global Positioning System data. OCRTN, RICRTN, and SDCRTN represent GPS networks for Orange, Riverside, and San Diego Counties in southern California. These stations are maintained by the Scripps Orbit and Permanent Array Center (SOPAC) [31].

Data is published from these stations in the binary RYO format. By connecting a sequence of filters, we convert and republish the data as ASCII and as Geography Markup Language (GML) formatted data.

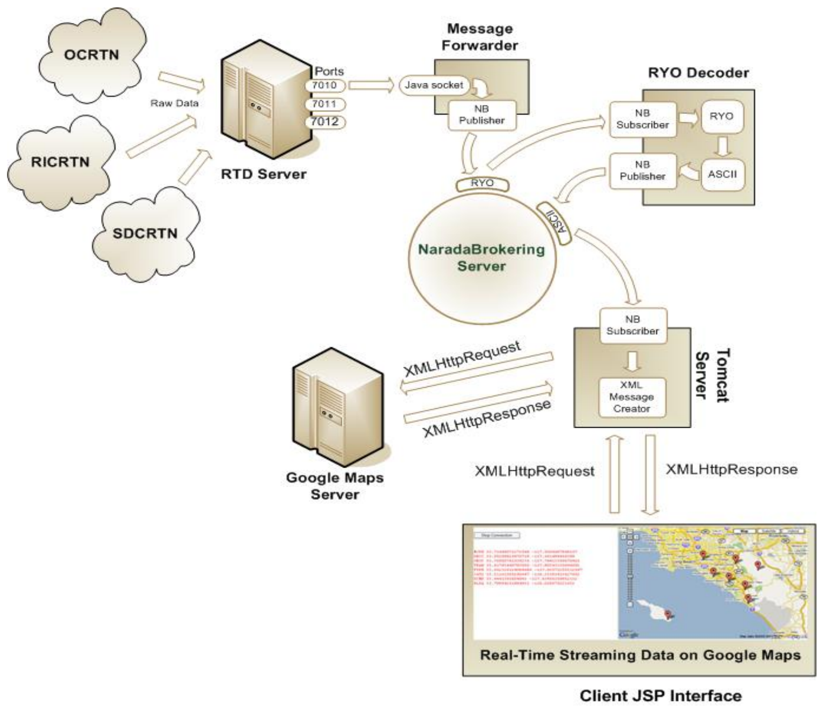

Figure 3 Naradabrokering may be used to support filters of real-time GPS data.

We are currently developing more sophisticated real-time data filters for data mining. Tools such as RDAHMM [32] may be used to detect state changes in archived GPS time signals. These may be associated with both seismic and aseismic causes. We are currently working to develop an RDAHMM filter that can be applied to real-time signals and link them in streaming fashion to the Open Geospatial Consortium standard services supporting integration of maps, features and sensors [33].

6. Future Work

The NaradaBrokering system has been recently augmented with the ability to create topics [34] and discover and allow secure access to topics and the data published on it. We plan to leverage this capability to create secure access to data streams. Further, implementing broker discovery [35] will allow us to select nearest broker when a WSProxy publishes/subscribes to data streams.

Security for real-time streams is also of course a critical challenge but the architecture supports message-based security like WS-Security [36] and we suggest that a form of WS-SecureConversation is natural for streams.

Conventional support of SOAP messages using the verbose "angle-bracket" representation is too slow for many applications. Thus we and others are researching [10, 37] a systematic use of "fast XML and SOAP" where services negotiate the use of efficient representations for SOAP messages. All messages rigorously support the service WSDL but transport the SOAP Infoset using the angle bracket form in the initial negotiation but an efficient representation where possible for streamed data

Another interesting area is structuring the system so that it can be implemented either with standalone services, message brokers and clients or in a Peer-to-Peer mode. These two implementations have tradeoffs between performance and flexibility and both are important. The core architecture "naturally" works in both modes but the details are non trivial and require substantial further research.

References

1. Xiaohong Qiu *Message-based MVC Architecture for Distributed and Desktop Applications* Syracuse University PhD March 2 2005 http://grids.ucs.indiana.edu/ptliupages/publications/qiuPhDthesis.pdf
2. Wenjun Wu, Hasan Bulut, Ahmet Uyar, Geoffrey Fox *Adapting H.323 Terminals in a Service-Oriented Collaboration System* Special "Internet Media" issue of IEEE Internet Computing July-August 2005, Vol 9 number 4 pages 43-50 http://grids.ucs.indiana.edu/ptliupages/publications/H323GW-IC0.pdf
3. GlobalMMCS Collaboration Environment http://www.globalmmcs.org
4. Geoffrey Fox, Alex Ho, Shrideep Pallickara, Marlon Pierce, Wenjun Wu Grids for the GiG and Real Time Simulations Proceedings of Ninth IEEE International Symposium DS-RT 2005 on Distributed Simulation and Real Time Applications' Montreal October 10-12 2005 http://grids.ucs.indiana.edu/ptliupages/publications/gig/DSRTOct05.pdf
5. Ken Birman, Robert Hillman, Stefan Pleisch, Building network-centric military applications over service oriented architectures SPIE Conference DEFENSE TRANSFORMATION AND NETWORK-CENTRIC SYSTEMS Orlando Florida 31 March 2005.http://www.cs.cornell.edu/projects/quicksilver/public_pdfs/GIGonWS_final.pdf
6. NCOIC Network Centric Operations Industry Consortium http://www.ncoic.org/ and W2COG World Wide Consortium for the Grid http://www.w2cog.org/
7. Bill Levitt, NCOW RM Development Group Update on Target Technical View - Emerging Net-Centric Standards - NCOW Reference Model v1.1 The Open Group Conference January 25, 2005, San Francisco http://www.opengroup.org/gesforum/uploads/40/6574/NCOW_TTV_V1.1_Open_Group.ppt
8. NaradaBrokering Messaging System http://www.naradabrokering.org
9. Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Harshawardhan Gadgil, *Building Messaging Substrates for Web and Grid Applications* to be published in special Issue on *Scientific Applications of Grid Computing* in Philosophical Transactions of the Royal Society of London 2005

http://grids.ucs.indiana.edu/ptliupages/publications/RS-CGL-ColorOnlineSubmission-Dec2004.pdf .

10. Sangyoon Oh, Hasan Bulut, Ahmet Uyar, Wenjun Wu, Geoffrey Fox Optimized Communication using the SOAP Infoset For Mobile Multimedia Collaboration Applications Proceedings of the International Symposium on Collaborative Technologies and Systems CTS05 May 2005, St. Louis Missouri, USA.
11. For discussion, see for example http://grids.ucs.indiana.edu/ptliupages/presentations/DoDGridsAug25-05.ppt
12. Mehmet S. Aktas, Geoffrey C. Fox, Marlon Pierce An Architecture for Supporting Information in Dynamically Assembled Semantic Grids Technical report August 2005 http://grids.ucs.indiana.edu/ptliupages/publications/SKG2005_Aktas.pdf.
13. Shrideep Pallickara and Geoffrey Fox. NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware-2003.
14. Shrideep Pallickara and Geoffrey Fox. On the Matching Of Events in Distributed Brokering Systems. Proceedings of IEEE ITCC Conference on Information Technology. April 2004. pp 68-76 Volume II.
15. Mark Happner, Rich Burridge and Rahul Sharma. Sun Microsystems. Java Message Service Specification, 2000. http://java.sun.com/products/jms.
16. Ahmet Uyar and Geoffrey Fox. Investigating the Performance of Audio/Video Service Architecture I: Single Broker Proceedings of the International Symposium on Collaborative Technologies and Systems CTS05 May 2005, St. Louis Missouri, USA.
17. Hasan Bulut, Shrideep Pallickara and Geoffrey Fox. Implementing a NTP-Based Time Service within a Distributed Brokering System. ACM International Conference on the Principles and Practice of Programming in Java. Pp 126-134.
18. Web Services Eventing. Microsoft, IBM & BEA, http://ftpna2.bea.com/pub/downloads/WS-Eventing.pdf
19. Web Services Reliable Messaging Protocol (WS-ReliableMessaging) ftp://www6.software.ibm.com/software/developer/library/ws-reliablemessaging200403.pdf
20. Web Services Reliable Messaging TC WS-Reliability. http://www.oasis-open.org/
21. Web Services Base Notification (WS-BaseNotification). IBM, Globus, Akamai et al. ftp://www6.software.ibm.com/software/developer/library/ws-notification/WS-BaseN.pdf
22. HPSearch project website: http://www.hpsearch.org
23. Web Service Management. Available from http://msdn.microsoft.com/library/en-us/dnglobspec/html/ws-management1004.pdf.
24. Harshawardhan Gadgil, Jin-Yong Choi, Bernie Engel, Geoffrey Fox, Sunghoon Ko, Shrideep Pallickara, Marlon Pierce, Management of Data Streams for a Real Time Flood Simulation, CGL Technical Report, June 2004
25. Shrideep Pallickara and Geoffrey Fox. A Scheme for Reliable Delivery of Events in Distributed Middleware Systems. Proceedings of the IEEE International Conference on Autonomic Computing 2004.

26. Harshawardhan Gadgil, Geoffrey Fox, Shrideep Pallickara, Marlon Pierce, Robert Granat, A Scripting based Architecture for Management of Streams and Services in Real-time Grid Applications, In Proceedings of the IEEE/ACM Cluster Computing and Grid 2005 Conference, CCGrid 2005, Cardiff, UK.
27. Wenjun Wu, Geoffrey Fox, Hasan Bulut , Ahmet Uyar , Harun Altay . Design and Implementation of A Collaboration Web-services system. Journal of Neural, Parallel & Scientific Computations, Volume 12, 2004.
28. SMIL, http://www.w3.org/AudioVideo/
29. Grid workflow is summarized in GGF10 Berlin meeting http://www.extreme.indiana.edu/groc/ggf10-ww/ with longer papers to appear in a special issue of Concurrency&Computation: Practice&Experience at http://www.cc-pe.net/iuhome/workflow2004index.html. Editorial is Dennis Gannon and Geoffrey Fox Workflow in Grid Systems http://grids.ucs.indiana.edu/ptliupages/publications/Workflow-overview.pdf
30. Jia Yu and Rajkumar Buyya, A Taxonomy of Workflow Management Systems for Grid Computing, Technical Report, GRIDS-TR-2005-1, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, March 10, 2005. http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf.
31. Scripps Orbit and Permanent Array Center (SOPAC): http://sopac.ucsd.edu.
32. Robert Granat, Regularized Deterministic Annealing EM for Hidden Markov Models, Doctoral Dissertation, University of California Los Angeles, 2004.
33. The Open Geospatial Consortium: http://www.geospatial.org.
34. Shrideep Pallickara, Geoffrey Fox and Harshawardhan Gadgil. On the Creation & Discovery of Topics in Distributed Publish/Subscribe systems. (To appear) Proceedings of the IEEE/ACM GRID 2005. Seattle, WA.
35. Shrideep Pallickara, Harshawardhan Gadgil and Geoffrey Fox. On the Discovery of Brokers in Distributed Messaging Infrastructures. (To appear) Proceedings of the IEEE Cluster 2005 Conference. Boston, MA.
36. Shrideep Pallickara et al. A Security Framework for Distributed Brokering Systems. Available from http://www.naradabrokering.org.
37. K. Chiu, M. Govindaraju, and R. Bramley, “Investigating the Limits of SOAP Performance for Scientific Computing”, *Proc. of 11th IEEE International Symp on High Performance Distributed Computing HPDC-11 2002*, July 2002, p. 256.