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 for real time data stream applications, both in terms of lower level messaging and higher level service, filter and session structures. In our approach, messaging systems act as a Grid substrate that can provide qualities of service to various streaming applications ranging from audio-video collaboration to sensor grids. The messaging substrate is composed of distributed, hierarchically arranged message brokers that form networks. We discuss approaches to managing systems for both broker networks and application 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.

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, in which events are often buried in the application. 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]. 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 Network Centric Operations and Warfare Architecture [3, 4].

We have developed general purpose, open source software to support distributed streams, described in Sec. 2. NaradaBrokering [5] 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.

Our architecture supports the interesting concept of hybrid streams where multiple "simple streams" are intrinsically linked: examples are linkages of a stream of annotation white boards with original audio/video stream [7] and the combination of lossless and lossy codec streams (using perhaps parallel TCP and UDP respectively) to represent a large dynamic shared display.

Several applications 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 [6] and the linkage of annotations to video streams showing how composite streams can be supported for real-time annotation [7]. 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

The messaging infrastructure supports the application services with their filters, gateways and sessions reflecting both 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. Section 3 describes the integration of the management of messaging and higher-level services.

2. NaradaBrokering: a Distributed Messaging Substrate

NaradaBrokering [5, 9] is a messaging infrastructure that is based on the publish/subscribe paradigm. The system efficiently routes messages [10] 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. 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 [11]. 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 [12] 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, WS-ReliableMessaging, and WS-Reliability. Work on implementing the WS-Notification 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 are also enforced.

In our support for SOAP within NaradaBrokering we have introduced *filters* and *filter-pipelines.* 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.

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 example, 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 [14] 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.

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.

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. 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.

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) [14]. 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 [13] thus enabling guaranteed delivery for data streams.

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, whiteboards 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 the 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.

Streaming Filters: 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 filters, they can be divided into three categories: one-in-one-out filters, multiple-in-one out filters, and one-in-multiple-out. In addition, there is a final "sink" filter category. We discuss each of these below.

One-In-One-Out filters implement 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, also known as *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 that 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, or *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 graphs consisting of multiple filter chains.

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.

Global-MMCS Workflow Management: There is substantial literature on Grid and Service-based workflow [16]. 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.

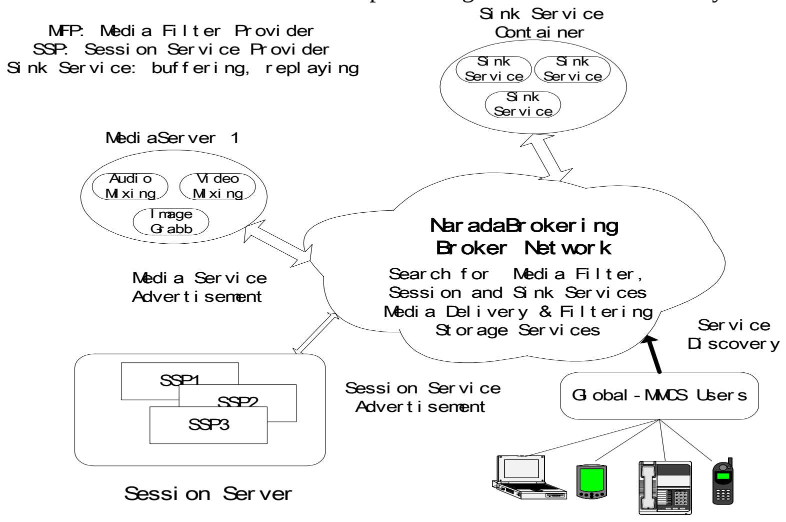
MFP: Media Filter Provider
SSP: Session Service Provider
Sink Service: buffering, replaying
Sink Service Container
Sink Service
Sink Service
Sink Service
MediaServer 1
Audio Mixing
Video Mixing
Image Grabb
NaradaBrokering Broker Network
Search for Media Filter, Session and Sink Services
Media Delivery & Filtering
Storage Services
Media Service Advertisement
Service Discovery
SSP1
SSP2
SSP3
Session Service Advertisement
Global-MMCS Users
Session Server

Fig. 1 Workflow and filters in GlobalMMCS.

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.

Session Management and NaradaBrokering Integration: As shown in Figure 1, 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 restarts it or locates 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 down-sampler 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 been unified 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 services, 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 Fig. 2, we depict our work to develop filters on live Global Positioning System data. OCRTN, RICRTN, and

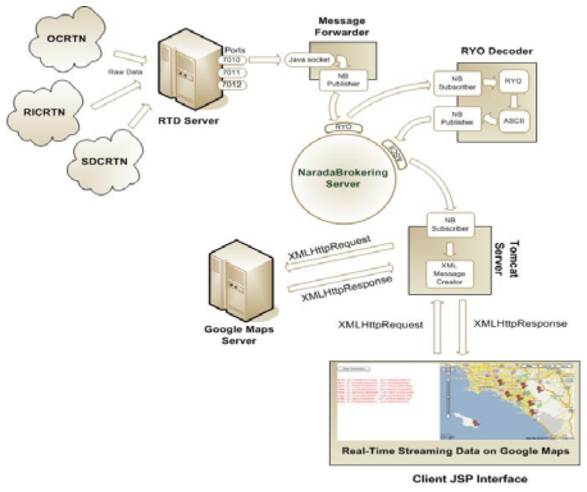

Fig. 2 Naradabrokering may be used to support filters of real-time GPS data.

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) and are published to an RTD server, where they are made publicly available. 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. The data can be further subdivided into individual station position measurements.

We are currently developing more sophisticated real-time data filters for data mining. Tools such as RDAHMM [17] 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.

6. Future Work

Conventional support of SOAP messages using the verbose "angle-bracket" representation is too slow for many applications. Thus we and others are researching [6, 18] 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 and 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 not trivial and require substantial further research.

References

1. Qiu, X.: Message-based MVC Architecture for Distributed and Desktop Applications Syracuse University PhD March 2 2005
2. Wu, W., Bulut, H., Uyar, A., and Fox, G.: Adapting H.323 Terminals in a Service-Oriented Collaboration System. In special "Internet Media" issue of IEEE Internet Computing July-August 2005, Vol 9, No. 4 pages 43-50 (2005)
3. Fox, G., Ho, A., Pallickara, S., Pierce, M., and Wu, W.: Grids for the GiG and Real Time Simulations Proceedings of Ninth IEEE International Symposium DS-RT 2005 on Distributed Simulation and Real Time Applications (2005)
4. Birman, K., Hillman, R., and Pleisch, S.: Building Network-Centric Military Applications Over Service Oriented Architectures. In proceedings of SPIE Conference Defense Transformation and Network-Centric Systems (2005) http://www.cs.cornell.edu/projects/quicksilver/public_pdfs/GIGonWS_final.pdf
5. Fox, G., Pallickara, S., Pierce, M., and Gadgil, H.: 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).

6. Oh, S., Bulut, H., Uyar, A., Wenjun Wu, Geoffrey Fox Optimized Communication using the SOAP Infoset For Mobile Multimedia Collaboration Applications. In proceedings of the International Symposium on Collaborative Technologies and Systems CTS05 (2005)
7. For discussion, see http://grids.ucs.indiana.edu/ptliupages/presentations/DoDGridsAug25-05.ppt
8. Aktas, M. S., Fox, G., and Pierce, M.: An Architecture for Supporting Information in Dynamically Assembled Semantic Grids Technical report (2005)
9. Pallickara, S. and Fox, G.: NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids. Proceedings of ACM/IFIP/USENIX International Middleware Conference Middleware (2003).
10. Pallickara, S. and Fox, G.: On the Matching Of Events in Distributed Brokering Systems. Proceedings of IEEE ITCC Conference on Information Technology, Volume II (2004) 68-76
11. Uyar, A. and Fox, G.: Investigating the Performance of Audio/Video Service Architecture I: Single Broker Proceedings of the International Symposium on Collaborative Technologies and Systems CTS05 (2005)
12. Bulut, H., Pallickara, S., and Fox, G.: Implementing a NTP-Based Time Service within a Distributed Brokering System. ACM International Conference on the Principles and Practice of Programming in Java. (2004) 126-134.
13. Pallickara, S. and Fox, G.: A Scheme for Reliable Delivery of Events in Distributed Middleware Systems. In Proceedings of the IEEE International Conference on Autonomic Computing (2004).
14. Gadgil, H., Fox, G., Pallickara, S., Pierce, M., and Granat, R.: 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)
15. Wu, W., Fox, G., Bulut, H., Uyar, A., and Altay, H.: Design and Implementation of a Collaboration Web-services system. Journal of Neural, Parallel & Scientific Computations, Volume 12 (2004)
16. 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. See also Gannon, D. and Fox, G.: Workflow in Grid Systems.
17. Granat, R.: Regularized Deterministic Annealing EM for Hidden Markov Models, Doctoral Dissertation, University of California Los Angeles (2004)
18. Chiu, K., Govindaraju, M., and Bramley, R.: Investigating the Limits of SOAP Performance for Scientific Computing, Proc. of 11th IEEE International Symp on High Performance Distributed Computing HPDC-11 (2002) 256.