Keynote Speech
Collaborative Web Services and Peer-to-Peer Grids

Geoffrey Fox[1,2,4], Hasan Bulut[2], Kangseok Kim[2], Sung-Hoon Ko[1], Sangmi Lee[5], Sangyoon Oh[2], Shrideep Pallickara[1], Xiaohong Qiu[1,3], Ahmet Uyar[1,3], Minjun Wang[1,3], Wenjun Wu[1]
[1]Community Grid Computing Laboratory, Indiana University
501 N Morton Suite 224, Bloomington IN 47404
[2]Computer Science Department, Indiana University
[3]EECS Department, Syracuse University
[4]School of Informatics and Physics Department, Indiana University
[5]Computer Science Department Florida State University
gcf@indiana.edu,hbulut@cs.indiana.edu, kakim@cs.indiana.edu
shko@grids.ucs.indiana.edu, slee@grids.ucs.indiana.edu, ohsangy@cs.indiana.edu,
spallick@indiana.edu, xiqiu@syr.edu, auyar@syr.edu, mwang03@syr.edu, wewu@indiana.edu

Abstract
We define Peer-to-Peer Grids built around integration of technologies from the peer-to-peer and grid fields. Grids provide robust largely asynchronous shared resources for virtual organizations. We show how one can extend this to synchronous collaboration with a common peer-to-peer Grid architecture. Two building blocks are a common dynamic messaging environment and systematic use of Web Services including one to support session specification and control. We present our prototype audio-video conferencing Web Service and messaging environment NaradaBrokering. We describe a core collaboration Web Service defined with an XML protocol XGSP subsuming the capabilities of H323, SIP and JXTA.

1 Peer-to-Peer Grids
There are no crisp definitions of Grids [1,2] and Peer-to-Peer (P2P) Networks [3] that allow us to unambiguously discuss their differences and similarities and what it means to integrate them. However these two concepts conjure up stereotype images that can be compared. Taking "extreme" cases, Grids are exemplified by the infrastructure used to allow seamless access to supercomputers and their datasets. P2P technology is exemplified by Napster and Gnutella, which can enable ad hoc communities of low-end clients to advertise and access the files on the communal computers. Each of these examples offers services but they differ in their functionality and style of implementation. The P2P example could involve services to set-up and join peer groups, browse and access files on a peer, or possibly to advertise one's interest in a particular file. The "classic" grid could support job submittal and status services and access to sophisticated data management systems. Grids typically have structured robust security services while P2P networks can exhibit more intuitive trust mechanisms reminiscent of the "real world". Again Grids typically offer robust services that scale well in pre-existing hierarchically arranged organizations; P2P networks are often used when a best effort service is needed in a dynamic poorly structured community. If one needs a particular "hot digital recording", it is not necessary to locate all sources of this; a P2P network needs to search enough plausible resources that success is statistically guaranteed. On the other hand, a 3D simulation of the universe might need to be carefully scheduled and submitted in a guaranteed fashion to one of the handful of available supercomputers that can support it.

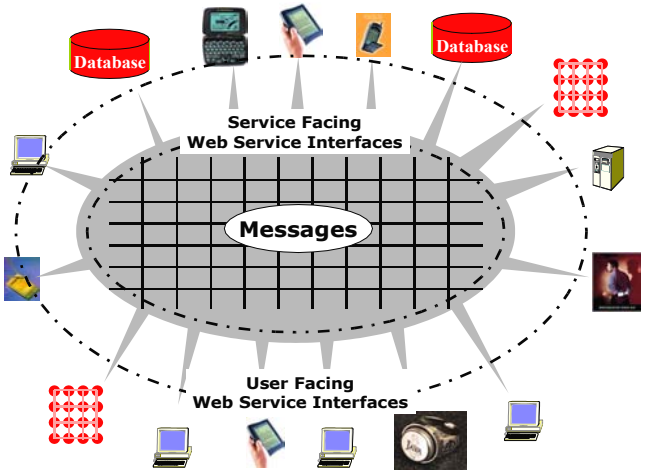

Fig. 1 Peer-to-Peer Grid of Resources linked by messages

In this article, we use the concept of a Peer-to-Peer Grid shown in figure 1 with a set of services that include those of Grids and P2P networks and support naturally environments that have features of both limiting cases. We will use the emerging Web Services model being used in both Internet and Grid communities and described in [4,5,6]. Here we capture the essence [7] of Web Services as using a distributed object model with all interfaces defined as "ports" interacting with XML specified messages.

2 Collaboration in P2P Grids
Both Grids and P2P networks are associated with collaborative environments. P2P networks started with ad-hoc communities such as those sharing MP3 files; Grids

support virtual enterprises or organizations – these are unstructured and structured societies respectively. At a high level collaboration involves sharing and in our context this is sharing of Web Services, objects or resources. These last three concepts are in principle essentially the same thing although today sharing "legacy applications" like Microsoft Word is not so easily considered as sharing Web services. Nevertheless we can expect that Web Service interfaces to "everything" will eventually be available and will take this point of view below where Word, a Web Page, a computer visualization or the audio-video (at say 30 frames per second) from some video-conferencing system will all be viewed as objects or resources with a known Web service interface.

There are many styles and approaches to collaboration. In asynchronous collaboration, different members of a community access the same resource; the Web has revolutionized asynchronous collaboration with in its simplest form, one member posting or updating a web page, and others accessing it. Asynchronous collaboration has no special time constraint and typically each community member can access the resource in their own fashion; objects are often shared in a coarse grain fashion with a shared URL pointing to a large amount of information. Asynchronous collaboration is quite fault-tolerant as each user can manage their access to the resource and independently accommodate difficulties such as poor network connectivity; further well-established caching techniques can usually be used to improve access performance as the resource is not expected to change rapidly. Synchronous collaboration is at a high level no different from the asynchronous case except that the sharing of information is done in real-time. The "real-time" constraint implies delays of around 10-1000 msec. per participant or rather "jitter in transit delays" of a "few" msecs. Note these timing can be compared to the second or so it takes a browser to load a new page; the several seconds it takes a lecturer to gather thoughts at the start of a new topic (new PowerPoint slide); and the 30 msec. frame size natural in audio/video transmission. These numbers are much longer than the parallel computing MPI message latency measured in microsecond(s) and even the 0.5-3 msec. typical latency of a middle-tier broker. We exploit this observation in section 5 to build powerful messaging subsystems at the cost of such middleware broker overhead. Nevertheless synchronous collaboration is much harder than the asynchronous case for several reasons. [8,9] The current Internet has no reliable quality of service and so it hard to accommodate problems coming from unreliable networks and clients. If your workstation crashes during a an asynchronous access, one just needs to reboot and restart one's viewing at the point of interruption; unfortunately in the synchronous case, after recovering from an error, one cannot resume where one lost contact because the rest of the collaborators have moved on. Further synchronizing objects among the community must often be done at a fine grain size. For asynchronous education, the teacher can share a complete lecture whereas in a synchronous session we might wish to share a given page in a lecturer with a particular scrolling distance and particular highlighting. In summary synchronous and asynchronous collaboration both involve object sharing but the former is fault sensitive, has modest real-time constraints and requires fine grain object state synchronization. The application of collaboration in research and education is discussed in refs. [10] and [11] respectively.

The sharing mechanism can be roughly the same for both synchronous and asynchronous case. One needs to establish communities (peer groups) by either direct (members join a session) or indirect (members express interest in topics and are given opportunity to satisfy this interest). The indirect mechanism is most powerful and is familiar in P2P systems with JXTA [12] using XML expressed advertisements to link together those interested in a particular topic. Audio-video conferencing systems typically have a direct method with perhaps email used to alert potential attendees of an upcoming session. Commercial web-conferencing systems like WebEx [13] and Placeware [14] use this approach. In asynchronous collaboration, one typically "just" has notification mechanisms for object availability and update. However such sharing environments do not usually support the "microscopic" events as say one user edits a file; rather "major events" (check-in and check-out) are communicated between participants. Nevertheless synchronous and asynchronous collaboration can be supported by variants of the same publish-subscribe messaging technology of the type described in section 5.

Collaboration involves multiple capabilities which we now discuss in a Web Services or Grid framework.

1. Mechanism to set up members (people, devices) of a "collaborative sessions" and their properties
2. Generic shared tools such as text chat, white boards, and audio-video conferencing
3. Shared applications such as Web Pages, PowerPoint, and scientific visualization

In section 3, we describe the first item as a "Collaboration as a Web service" which is currently set using standards like H323 SIP or JXTA. The last two categories in the list above can be viewed as "just shared objects" where objects could be Web Services but typically are not at the moment. In section 4, we suppose that we can port all objects to Web Services and describe how one can build a general approach for making such Web services collaborative. This gives us "Collaborative Web Services" with one example -- audio-video conferencing -- given in section 3.

3 Collaboration as a Web Service

Here we also discuss audio-video (A/V) conferencing as a Web Service because the two best known session specification standards SIP and H323 are associated with this application. We have [15,16] defined an XML specification XGSP defining session information for both general collaborations and the A/V subsystem. This describes registration, session parameters, session membership, negotiation (of for instance common codecs) and linkage of clients to media servers.

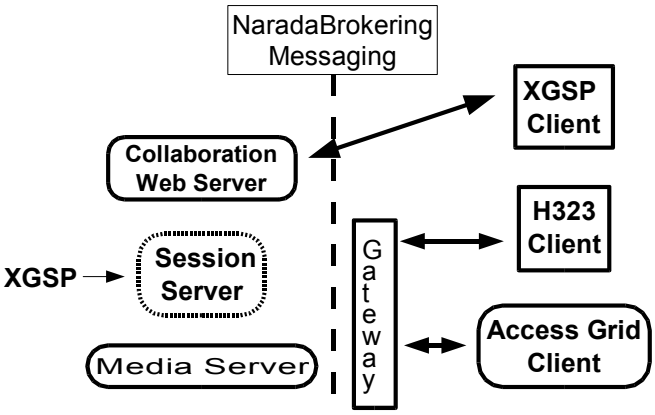

Fig. 2 Integration of multiple A/V subsystems using XGSP

Figure 2 shows our prototype with multiple clients – the Access Grid from Argonne [17], a typical H323 client (such as a commercial Polycom system [18]) and one using a native XGSP framework, interoperating. There is the specialized A/V application media server and the general session server with a web interface. We describe the messaging environment NaradaBrokering in section 5.

4 Collaborative Web Services

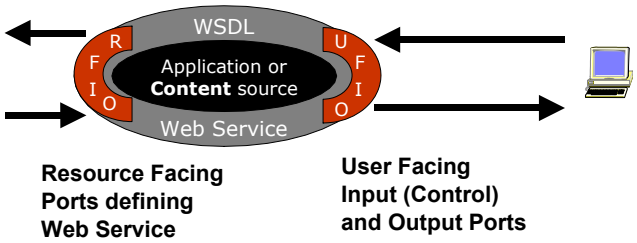

Fig. 3 User and Resource Facing Ports of a Web Service

In order to describe a general approach [19] to collaboration, we need to assume that every Web service has one or more ports in each of categories shown in fig. 3. We exploit the fact that the state and results of each Web service is entirely defined by messages on these ports. The first category are (resource-facing) ports which supply the information needed to define the state of the Web service; these may be augmented by user-facing input port(s) that allow control information to be passed by the user. The final category is formed by user-facing output ports that supply information needed to construct the user interface. Asynchronous collaboration can share the data (e.g. URL for a Web page or body of an email message) needed to define a Web service (display Web page or browse e-mail in examples).

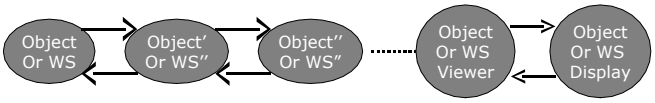

Fig. 4 Typical Pipeline between Web services

Let us now consider synchronous collaboration. If one examines an object, then there is typically some pipeline as seen in fig. 4 from the original object to the eventual displayed user interface; we here assume that each stage of the pipeline is a Web service with data flowing from one to another. Rather than a simple pipeline, one can have a complex dynamic graph linking services together but this does not change this discussion. Note that such a pipeline is natural to support not only needed transformations but also modern portal architectures [20, 21] where one stage in the pipeline corresponds to the aggregation shown in fig. 5.

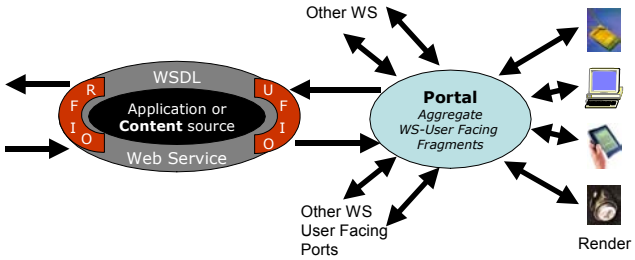

Fig. 5 Web Service User Facing Ports and Aggregation Portals

One can get different types of sharing depending on which "view" of the basic object one shares i.e. on where one intercepts the pipeline and shares the flow of information after this interception. We can identify three particularly important cases illustrated in figures 6, 7 and 8; these are shared display, shared input port and shared user-facing output port. The shared input port case is usually called "shared event" but in fact in a modern architecture with all resources communicating by messages, all collaboration modes can be implemented with a similar event mechanism described in section 5. In each collaboration mode we assume there is a single "master" client which "controls" the Web service; one can in fact have much more complex scenarios with simultaneous and interchangeable control. However in all cases, there is

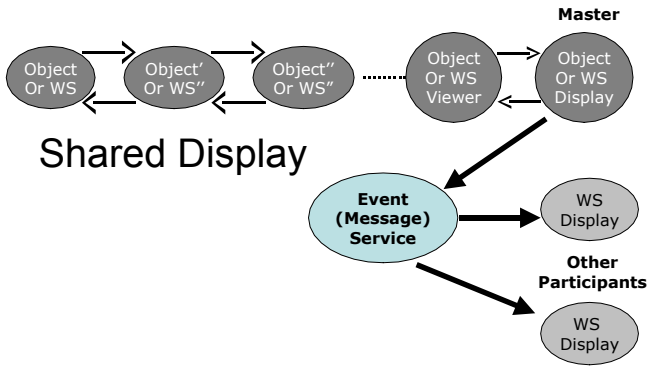

Fig. 6 Shared Display Collaboration Mode

instantaneously one "master" and one must transmit the state as seen by this system to all other participants.

In shared display model of fig. 6, one shares the bitmap (vector) display and the state is maintained between the clients by transmitting (with suitable compression) the changes in the display. As with video compression like MPEG, one uses multiple event types with some defining full display and others just giving updates. Obviously the complete display requires substantial network bandwidth but it is useful every now and then so that one can support clients joining throughout session, has more fault tolerance and can define full display update points (major events) where asynchronous clients can join a recording. Supporting heterogeneous clients requires that sophisticated shared display environments automatically change size and color resolution to suit each community member. Shared display has one key advantage – it can immediately be applied to all shared objects; it has two obvious disadvantages – it is rather difficult to customize and requires substantial network bandwidth.

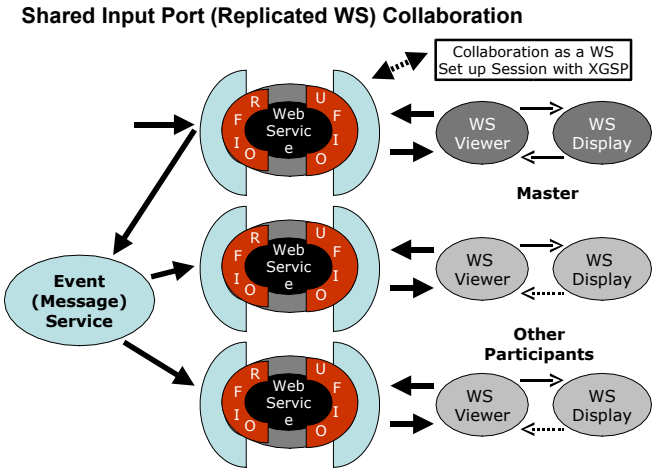

Fig. 7 Shared Input Port Model for Collaborative Web Services

In the shared input port (or input message) model of fig. 7, one replicates the Web service to be shared with one copy for each client. Then sharing is achieved by intercepting the pipeline before the master web service and directing copies of the messages on each input port of the "master" Web service to the replicated copies. Only the user-facing ports in this model are typically partially shared with data from the master transmitted to each replicated Web service but in a way that can be overridden on each client. We can illustrate this with a more familiar PowerPoint example. Here all the clients have a copy of the PowerPoint application and the presentation to be shared. On the master client, one uses some sort of COM wrapper to detect PowerPoint change events such as slide and animation changes. These "change" events are sent to all participating clients. This model isn't usually phrased as "shared input ports" but that's just because PowerPoint as currently shipped is not set up as a Web service with a messaging interface. One can build a similar shared Web browser and for some browsers (such as that for SVG from Apache) one can in fact directly implement the Web service model. [22-24] There is a variant here as one can either trap internal events (such as slide changes in PowerPoint or textareas changes in a browser) or the external mouse and keyboard events that generated them. We once developed a sophisticated shared browser using the JavaScript event model to trap user input to a browser. These events were transmitted directly to participating clients to implement such a shared input port model with the user interface playing the role of input ports. We can hope that developments [25] such as WSRP and WSIA (Web services for Remote Portals and Interactive Applications) will define user-facing message ports and the interactions of Web services with input devices so that a coherent systematic approach can be given for replicated Web services with shared input ports.

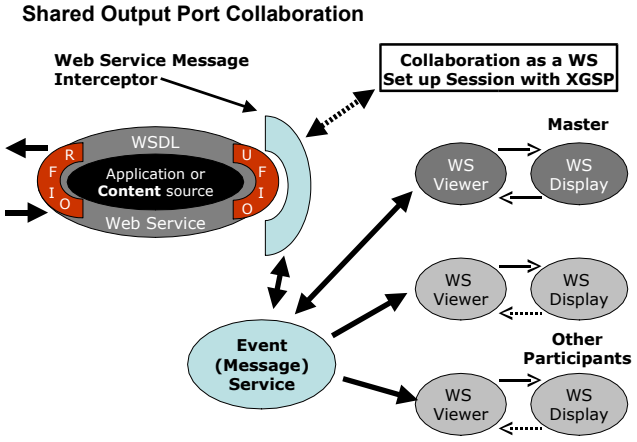

Fig. 8 Shared Output Port Model for a Collaborative Web Service

The shared output port model of fig 8 only involves a single Web service with user-facing ports giving a messaging interface to the client. As in the next section and fig. 5, a portal could manage these user-facing ports. As (by definition) the user-facing ports of a Web service define the user interface, this mechanism simply gives a collaborative version of any Web service. One simple example can be built around any content or multi-media server with multi-cast output stream(s). This method naturally gives, like shared display, an identical view for each user but with the advantage of typically less network bandwidth since the bitmap display usually is more voluminous than the data transmitted to the client to define the display. Elsewhere [19, 22, 23], we discuss user interfaces and suggest that the user facing ports should not directly define the interface but a menu allowing the client to choose the interface style. In such a case, one can obtain from the shared user-facing model, collaborative views that are customized for each user. Of course the replicated Web service model of fig. 7 offers even greater customizability as each client has the freedom to accept or reject data defining the shared Web service.

Here we have discussed how to make general applications collaborative. Figs. 7 and 8 also use the (XGSP-based) core collaboration web service introduced in section 3 [15,16] with ports allowing sessions to be defined and to interact with the event service. This collaboration web service can support both asynchronous and all modes of synchronous collaboration.

5 NaradaBrokering Message Service

We have designed and implemented a system NaradaBrokering [26-28] supporting the messaging needed in a Peer-to-Peer Grid with a dynamic collection of brokers supporting a generalized publish-subscribe mechanism.

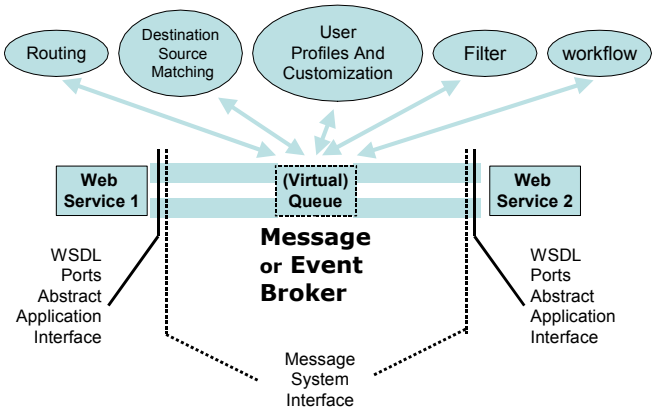

Fig. 9 Architecture of NaradaBrokering Message/Event Serice

This system "intercepts" all messages and provides capabilities that can be performed at the message level independent of the the details of the services generating the messages. The routing includes dynamic protocol choice and tunneling through firewalls to provide optimized transport subject to application-level quality of service requirements. It supports TCP,UDP, multicast, SSL and RTP. NaradaBrokering breaks a given message transmission into links (each between NaradaBrokering nodes) and attempts to use the most appropriate protocol on each link using a built-in performance monitoring service. As described elsewhere [5,6,19, 26-28] this can operate either in a client-server mode like JMS (the Java Message Service [29]) or in a completely distributed JXTA-like peer-to-peer mode. By combining these two disparate models, NaradaBrokering can allow optimized performance-functionality trade-offs for different scenarios. Note as mentioned earlier, that typical overheads for broker processing are around one millisecond. This is acceptable for real-time collaboration [16] and even audio-video conferencing where each frame takes around 30 milliseconds. We have demonstrated that such a general messaging system can be applied to real-time synchronous collaboration using the commercial Anabas infrastructure [30]. The Anabas system uses JMS to handle all collaboration modes and we have successfully used NaradaBrokering to replace JMS in this case.

Thus in collaboration, NaradaBrokering provides the appropriate (software controlled multicast) linkage between session participants and further copes with firewalls and other transport link issues. In the case of slow links such as dial-up links and handheld clients, NaradaBrokering supports filtering of the messages to change formats such as reducing resolution of an image or choosing a more compact codec for audio-video messages. We have built "all" protocol s into NaradaBrokering so that can be used in all collaboration applications including those needing UDP and TCP/IP

Acknowledgments

This publication made possible through partial support provided by DoD High Performance Computing Modernization Program (HPCMP) Programming Environment & Training (PET) activities through Mississippi State University under the terms of Agreement No. # GS04T01BFC0060. The University of Illinois also provided support through the PACI Partners program funded by NSF. The opinions expressed herein are those of the authors and do not necessarily reflect the views of the sponsors.

References

1. The Grid Forum http://www.gridforum.org
2. Globus Grid Project http://www.globus.org

3. "Peer-To-Peer: Harnessing the Benefits of a Disruptive Technology", edited by Andy Oram, O'Reilly Press March 2001.
4. Fran Berman, Geoffrey Fox and Tony Hey, 'Grid Computing: Making the Global Infrastructure a Reality', John Wiley & Sons Ltd, Chichester, 2003. See http://www.grid2002.org
5. Hasan Bulut, Geoffrey Fox, Dennis Gannon, Kangseok Kim, Sung-Hoon Ko, Sangmi Lee, Sangyoon Oh, Xi Rao, Shrideep Pallickara, Quinlin Pei, Marlon Pierce, Aleksander Slominski, Ahmet Uyar, Wenjun Wu, Choonhan Youn "An Architecture for e-Science and its Implications" presented at *2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2002)* July 17 2002. http://grids.ucs.indiana.edu/ptliupages/publications/spectsescience.pdf
6. Geoffrey Fox, Ozgur Balsoy, Shrideep Pallickara, Ahmet Uyar, Dennis Gannon, and Aleksander Slominski, "Community Grids" invited talk at *The 2002 International Conference on Computational Science*, April 21 -- 24, 2002 Amsterdam, The Netherlands. http://grids.ucs.indiana.edu/ptliupages/publications/iccs.pdf
7. Web Services Description Language(WSDL) version 1.1 http://www.w3.org/TR/wsdl
8. Collection of Resources on distance education by G. Fox http://grids.ucs.indiana.edu/ptliupages/publications/disted/.
9. Fox, G., Scavo, T., Bernholdt, D., Markowski, R., McCracken, N., Podgorny, M., Mitra, D., and Malluhi, Q., ``Synchronous Learning at a Distance: Experiences with TANGO Interative". Supercomputing 98 Conference, November 1998. http://www.old-npac.org/projects/training/Papers/sc98/
10. Geoffrey Fox, Sung-Hoon Ko, Marlon Pierce, Ozgur Balsoy, Jake Kim, Sangmi Lee, Kangseok Kim, Sangyoon Oh, Xi Rao, Mustafa Varank, Hasan Bulut, Gurhan Gunduz, Xiaohong Qiu, Shrideep Pallickara, Ahmet Uyar, Choonhan Youn, *Grid Services for Earthquake Science*; Concurrency and Computation: Practice and Experience in ACES Special Issue, 14, 371-393, 2002. http://aspen.ucs.indiana.edu/gemmauisummer2001/resources/gemandit7.doc
11. Geoffrey Fox, "Education and the Enterprise with the Grid", chapter in ref. 4.
12. Sun Microsystems JXTA Peer to Peer technology. http://www.jxta.org.
13. WebEx Collaboration Environment. http://www.webex.com
14. Placeware Collaboration Environment. http://www.placeware.com
15. Geoffrey Fox, Wenjun Wu, Ahmet Uyar, Hasan Bulut "A Web Services Framework for Collaboration and Audio/Videoconferencing"; proceedings of *2002 International Conference on Internet Computing IC'02*: Las Vegas, USA, June 24-27, 2002. http://grids.ucs.indiana.edu/ptliupages/publications/avwebserviceapril02.pdf
16. Hasan Bulut, Geoffrey Fox, Shrideep Pallickara,Ahmet Uyar and Wenjun Wu, *Integration of NaradaBrokering and Audio/Video Conferencing as a Web Service.* http://grids.ucs.indiana.edu/ptliupages/publications/AVOverNaradaBrokering.pdf
17. Access Grid Conferencing Environment from Argonne National Laboratory, http://www.accessgrid.org
18. Polycom Conferencing Environment http://www.polycom.com
19. Geoffrey Fox, Dennis Gannon, Sung-Hoon Ko, Sangmi Lee, Shrideep Pallickara, Marlon Pierce, Xiaohong Qiu, Xi Rao, Ahmet Uyar, Minjun Wang, Wenjun Wu, "Peer-to-Peer Grids", chapter in ref. 4.
20. Jetspeed Portal from Apache http://jakarta.apache.org/jetspeed/site/index.html
21. O. Balsoy, M. S. Aktas, G. Aydin, M. N. Aysan, C. Ikibas, A. Kaplan, J. Kim, M. E. Pierce, A. E. Topcu, B. Yildiz, and G. C. Fox., "The Online Knowledge Center: Building a Component Based Portal." Proceedings of the International Conference on Information and Knowledge Engineering, 2002. http://grids.ucs.indiana.edu:9000/slide/ptliu/research/gateway/Papers/OKCPaper.pdf
22. Geoffrey Fox, Sung-Hoon Ko, Kangseok Kim, Sangyoon Oh, Sangmi Lee, "Integration of Hand-Held Devices into Collaborative Environments"; proceedings of the *2002 International Conference on Internet Computing* (IC-02). June 24-27 Las Vegas. http://grids.ucs.indiana.edu/ptliupages/publications/pdagarnetv1.pdf.
23. Sangmi Lee, Geoffrey Fox, Sunghoon Ko, Minjun Wang, Xiaohong Qiu, "Ubiquitous Access for Collaborative Information System using SVG", Proceedings of SVGopen conference July 2002, Zurich, Switzerland. http://grids.ucs.indiana.edu/ptliupages/projects/carousel/papers/draft.pdf
24. Geoffrey Fox, Sangmi Lee, Sunghoon Ko, Kangseok Kim, Sangyoon Oh, "CAROUSEL Web service: Universal Accessible Web service Architecture for Collaborative Application",November 2002, http://grids.ucs.indiana.edu/ptliupages/publications/Carousel_PerCom03.doc.
25. OASIS Web Services for Remote Portals (WSRP) and Web Services for Interactive Applications (WSIA) http://www.oasis-open.org/committees/

26. Geoffrey Fox and Shrideep Pallickara "The NaradaBrokering Event Brokering System: Overview and Extensions; proceedings of the *2002 International Conference on Parallel and Distributed Processing Techniques and Applications* (PDPTA'02). http://grids.ucs.indiana.edu/ptliupages/projects/NaradaBrokering/papers/NaradaBrokeringBrokeringSystem.pdf
27. Geoffrey Fox and Shrideep Pallickara "JMS Compliance in the NaradaBrokering Event Brokering System" to appear in the proceedings of the *2002 International Conference on Internet Computing* (IC-02). http://grids.ucs.indiana.edu/ptliupages/projects/NaradaBrokering/papers/JMSSupportInNaradaBrokering.pdf
28. Geoffrey Fox, Shrideep Pallickara, and Xi Rao, "A Scaleable Event Infrastructure for Peer to Peer Grids", proceedings of 2002 Java Grande/ISCOPE Conference, Seattle, November 2002, ACM Press, ISBN 1-58113-599-8, pages 66-75. http://grids.ucs.indiana.edu/ptliupages/publications/ScaleableEventArchForP2P.doc
29. Sun Microsystems. Java Message Service .http://java.sun.com/products/jms
30. Anabas Collaboration Environment, http://www.anabas.com