How the Internet Helps Build

Collaborative Multimedia Applications

Start with the Internet itself for freely available software technologies, then add some custom software.

FOR PROGRAMMERS, DESIGNING AND IMPLEMENTING distributed multimedia applications used to be a formidable challenge. As recently as five years ago, tools were scarce for developing and integrating multimedia content, as was system support for real-time media streaming and its control. Application pioneers had little choice but to custom-build most of the technology needed for their distributed multimedia applications. Custom building typically involved such sophisticated elements as authoring tools [6], reliable audio-video transport mechanisms [9], connection management schemes [8], and agents to coordinate the behavior of cooperating application components dispersed throughout a network [5]. Aside from the heavy labor requirement implied by this approach, another practical consequence was the difficulty of achieving application portability.

The recent Internet revolution has greatly affected construction of networked multimedia applications. New approaches to distributing application logic (such as Java applets) and coordinating distributed processing (such as CORBA) have claimed the attention of application developers [4]. And Web browsers, such as Netscape Navigator and Microsoft Internet Explorer, now offer a universally available container in which to assemble and integrate rich interactive software components. Moreover, the range of today's multimedia toolkits and component

John R. Nicol, Yechezkal S. Gutfreund, Jim Paschetto, Kimberly S. Rush, and Christopher Martin

We learned that we could go a long way toward being technology integrators (as opposed to technology inventors).

technologies has expanded significantly as a result of a new industry driven by the World-Wide Web's mass appeal. Such developments have fostered interest in creating real-time services for the Web and the applications they enable [3].

Here, we expand on some of our experience developing distributed multimedia applications the hard way, explaining how they helped motivate our migration toward Internet component technologies as a platform on which to build a new generation of prototype business applications. We focus on the design and implementation of TourGuide—the first nontrivial collaborative application we built using this approach to integrating off-the-shelf Internet component technologies. Though substantial efforts to build variations of multimedia-enhanced collaborative applications for the Internet have been reported ([7], for example), a further goal of our work has been to test the viability of building such applications using a Web browser as the principal application container. This way, applications would be highly available (most people already have a browser) and somewhat more platform-independent (popular browsers are supported on multiple platforms).

A critical step toward convincing potential end users of the value of emerging network technologies is to create what we call "concept business applications," or service prototypes spanning vertical market segments, designed to show how multimedia techniques, supported by state-of-the-art networking technology, influence sophisticated business communication. Simple multimedia presentation suites and general-purpose desktop video-conferencing applications are insufficient for this purpose, since they leave too much to users to fill in concerning how they alone can fundamentally change the nature of how their organizations do business.

Our first major step in exploring the validity of the concept business applications approach yielded the Broadband Multimedia Applications Demonstration Suite (BMADS) [9]—a custom collection of networking and middleware components and applications—spelling out the user benefits of broadband network support. On the basis of many BMADS demonstrations over the past several years, we are now convinced of the value of the concept business applications approach, despite some practical drawbacks to BMADS. For example, a notable lack of commercial software for such applications during BMADS development required us to design and implement everything we needed from scratch, including reliable mechanisms for transporting and switching real-time media streams, connection management functions, video-conferencing tools, and application-sharing widgets. This effort involved a time-consuming and iterative process as we augmented the demonstrations included in the suite and enhanced the underlying technology required for their support over a period of about five years.

Many obstacles we faced developing BMADS have become less problematical in recent years, partly due to significant cultural developments concerning the perceived value of multimedia technology, as well as significant technical advances. The spectacular growth in popularity of the Web [1] has spawned something of a cottage industry centered on developing multimedia-related technologies. So, whereas the application of multimedia techniques was recently regarded as technically impractical by many, the Internet has changed the technology options so much that multimedia is quickly being accepted as a mainstream concern. Prospects for today's multimedia application developers have also improved greatly. And more pragmatically, it is no longer feasible to develop concrete plans for multi-year investigations of application concepts and their implementation, as we did with BMADS. Six months has become a very long time in the multimedia applications business.

We decided to opt in favor of a development approach that would allow us to readily incorporate the latest multimedia technology as part of our

ongoing experiment with multimedia concept business services. We couldn't help but turn to the Internet, seeking to determine the viability of constructing prototypical distributed multimedia applications based as far as possible on integrating ready-made component technologies. We therefore began development of a distributed application we call TourGuide to support multimedia-enhanced collaboration over the Internet.

TourGuide

TourGuide provides an interactive multimedia experience in which one or more users playing the role of tour guide take a group of tourists through a preselected set of Web sites. During a TourGuide session, users can speak with one another, simultaneously view multimedia Web pages (under the tour guide's control), and give the tour guide feedback at the push of a button. The communications framework provided by TourGuide is general enough that it accommodates a range of applications, including training, customer support, and collaboration involving problem solving. We also created a TourGuide demonstration scenario to help viewers "connect" with the implications of the emerging technologies we used. That is, we describe a hypothetical business communication problem before demonstrating a technical solution to it through a carefully designed prototype service. As with BMADS, the TourGuide technology demonstrations allow us to play the roles of both tour guide and tourist.

In the TourGuide scenario, a fictitious Web-based design agency is contracted by GTE Corp. to prepare a package of Web-based content to help promote a major new marketing campaign. The agency has been told to take advantage of the latest audio-video streaming technologies in order to deliver the ad's message in a way as eye-catching as possible for a broad range of Web users. In our standard TourGuide demonstration, the agency representative (the tour guide) gets ready to present to executive clients at various locations (the tourists) the agency's preliminary Web content. The tour is scheduled, and the clients are given a URL for the design agency's site where they can join the tour. Joining the tour is accomplished through an applet-assisted login process. Once logged in, each tourist's browser loads a page containing a user interface similar to that shown in the right-hand portion of Figure 1.

The left-hand side of the tourist's browser includes a shared display area; any Web location selected by the tour guide as part of the tour causes the referenced page to be displayed in this area, and all participants in a TourGuide session then view the same contents in this portion of the interface at any given time. The right-hand side of the tourist-user interface includes several displays and controls; in the top-right area is the Roster, a Java applet that dynamically displays the names of the participants currently logged into the tour; in the bottom-right area is a control that allows each tourist to enter and leave a real-time voice conference with other tour participants, and the middle-right area of the tourist's user interface offers an array of buttons allowing feedback on the material being discussed. On pressing any of these buttons, a distinctive audio jingle is sounded at each participant's host computer, and a corresponding symbol is displayed

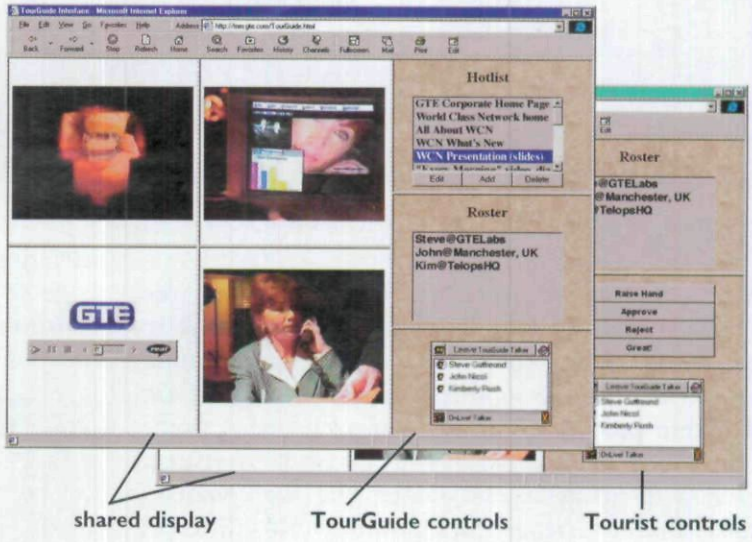

Figure 1. TourGuide and tourist user interfaces

in the Roster next to the name of the tourist providing the feedback. These stimuli give the tour guide useful feedback concerning client acceptance of the material being discussed.

At the beginning of a tour, a tour guide loads the TourGuide application into his or her own browser (left-hand side of Figure 1). Tourists can join or leave a tour at any time, resulting in a jingle being played at all remaining participant locations, along with an update on each participant's Roster display. Tourists joining the session see the Web location last selected by the tour guide load up in the shared Web-page area of their user interfaces.

The TourGuide user interface also features a HotList in the top-right area, as in Figure 1, left-

Figure 2. Collaborative page layout in TourGuide

hand side. Implemented by a Java applet, the HotList gives the tour guide a convenient way to create a list of locations to be visited during a tour, as well as a way to modify such lists during a tour. The tour guide uses the buttons associated with the HotList to create a list of sites that can be visited, each with an associated URL. The tour guide is thus able to create configurations for different tours, each of which can be loaded into the HotList for piloting the associated tour. The tour guide can take the tourists to any desired Web location by clicking on the corresponding entry from the HotList.

As part of our standard TourGuide demonstration scenario, the agency representative (the tour guide) takes tourists on a tour illustrating how key emerging Internet technologies are being implemented. Pages visited contain mostly custom multimedia content in support of GTE's proposed marketing campaign. We typically illustrate the collaborative aspect of the related communication by developing the interaction in several phases. So, in response to a client's awareness that most of its Internet customers are dial-up users, the tour guide first demonstrates how marketing messages can be delivered at modem speeds. This is accomplished by visiting pages that communicate the message through a dynamic slide show accompanied by an audio track, as in the shared-display area of Figure 1, or a low-resolution video movie that can be streamed in real time. We use feedback buttons throughout to illustrate how client-acceptance ratings can be communicated (while preserving floor control) as clients are exposed to the material prepared by the design agency. Clients then ask to see how the video would look at higher data rates, and, in response, the tour guide demonstrates higher-resolution versions of the same video content transmitted over various access technologies with different classes of connection speeds. Once again, clients can view the streamed video of the proposed marketing message at their own locations in approximate synchronization with all other tour participants. As the dialog proceeds, clients become increasingly confident about the viability of delivering a multimedia marketing message to a range of Internet customers; their attention then turns to aesthetic concerns. For example, the client

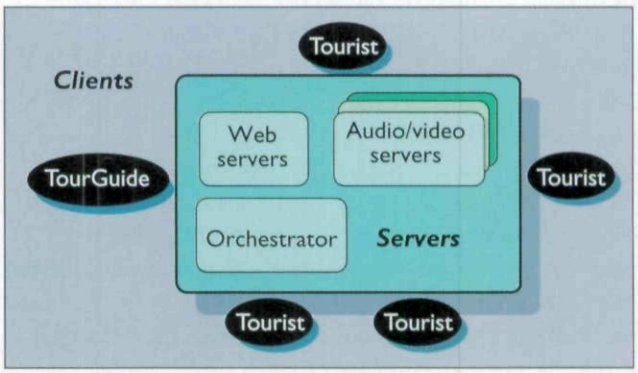

Figure 3. TourGuide application support platform

might want to discuss alternative page layout designs. The tour guide then visits a page containing some basic design widgets (see Figure 2), all of which can be manipulated in a what-you-see-is-what-I-see (WYSIWIS) manner until the blueprint for a preferred design is agreed upon, thus bringing the business meeting to a close.

Our basic approach was to create a general application support platform offering basic services pro-

Server Technology	Description and Application to TourGuide
Real Networks *RealAudio Server* *www.realnetworks.com*	RealServer, one of the first widely available technologies supporting real-time audio streaming over the Internet, can serve audio streams at 14.4kbps through ISDN rates in mono or stereo and is multistream-capable. It also supports event-driven presentations featuring playback of an audio stream with accompanying images synchronized by a common timeline. The TourGuide demo scenario uses RealServer technology to demonstrate delivery of dynamic presentations to users with modest Internet connection speeds (see Figure 1).
OnLive! Technologies *Community Server* *www.onlive.com*	OnLive! Technology's applications include the OnLive! Conferencing Server and the Talker client, which is a plug-in that can be embedded in a Web page for multipoint audio conferences supported by the Conferencing Server. The Talker plug-in was used in support of TourGuide's Internet telephony component.
VDOnet Corp. *VDOLive* *On-Demand Server* *www.vdonet.com*	The VDOLive On-Demand server is scalable and can serve multiple video streams simultaneously at the best connection rate for each client. The VDO encoder uses an efficient proprietary compression algorithm based on a wavelet-encoding scheme. The server software is specialized for streaming video accompanied by audio over low-bandwidth connections (from 14.4kbps). We used VDOLive technology to simulate the quality of video deliverable at dial-up (28.8kbps) and ISDN (128kbps) rates as part of the TourGuide demo.
Microsoft Corp. *NetShow Server* *www.microsoft.com*	The NetShow server is an open, extensible streaming platform supporting playback of streamed audio, video, and event-driven presentations. Because the software architecture is codec independent, it potentially supports any video-encoding format. We used the NetShow server as part of TourGuide to demonstrate the higher quality of video achievable by playback at ADSL rates, or multimegabits per second.

Table 1. Principal Internet servers in the TourGuide testbed

vided by a federation of network servers. TourGuide's application-specific elements were then layered on the platform through a combination of high-level application logic and scenario-specific multimedia content. A key motivation for constructing such an architecture (see Figure 3) was to explore the feasibility of building-in a high degree of generality, and hence reusability, in terms of the basic services and mechanisms available to application developers.

In accordance with our objective of testing the viability of building sophisticated Internet applications from emerging Internet software technologies, we learned we could go a long way toward being technology integrators (as opposed to technology inventors). We discovered that much of the functionality needed to support TourGuide was available through readily attainable technologies.

Table 1 lists the main Internet component technologies supporting TourGuide, along with comments on how each contributed to developing the platform. In addition to audio-video streaming servers, we used the Microsoft Information Server to serve Web content. We also created a variation of the TourGuide demonstration featuring a Microsoft Chat Server in support of real-time shared-text dialog and a Progressive Networks RealVideo server in support of live streaming the tour guide's talking head to the tourists.

Despite the considerable leverage gained from loosely integrating these component technologies, we were not entirely surprised to find functional holes that could be filled only with custom technology. The principal missing component was due to the need to coordinate events and actions between "peer" sets of applets across the network. Such coordination provides the application with the WYSIWIS sharing semantics characteristic of many collaborative applications. As an example of when such sharing is required by TourGuide, consider the actions following the tour guide's selection of a HotList item. That is, the URL of the page associated with the selected HotList item must somehow be communicated to all participating tourists. So, to coordinate Java applets, we developed a component we call the Orchestrator [5].

The Orchestrator, based loosely on the Linda programming language model [2], is a multithreaded Java application implementing a distributed shared object space (called TupleSpace) in which applets can post, read, or permanently store arbitrary Java objects. Coordination events can be posted to a TupleSpace and used to orchestrate the actions of remote applets. Applets can share information easily via TupleSpaces without needing to be aware of the underlying communication mechanisms. The Orchestrator manages TupleSpaces, handles calls from the applets, and controls the Archiver threads (see Figure 4).

A tuple has two parts—a key and a value. Though a key can be any Java object, it is often a string. To retrieve a tuple, an applet invokes the

`fetch` command on a designated TupleSpace using a specified key. A TupleSpace is a container for tuples. When an applet posts a tuple, it is added to the TupleSpace. Tuples can also be deleted by an applet. Orchestrated applets issue a `connect` command to connect to the Orchestrator. If an applet needs to post tuples to or fetch tuples from a TupleSpace, it sends an `attach` command to the Orchestrator to attach to the desired TupleSpace. When the tuples processing is complete, the applet can detach from the TupleSpace. An applet can also send commands to the Orchestrator to create and destroy TupleSpaces.

Applets can flag a TupleSpace to be cached (making it persistent). Cached TupleSpaces are checkpointed to disk at regular intervals by the Archiver.

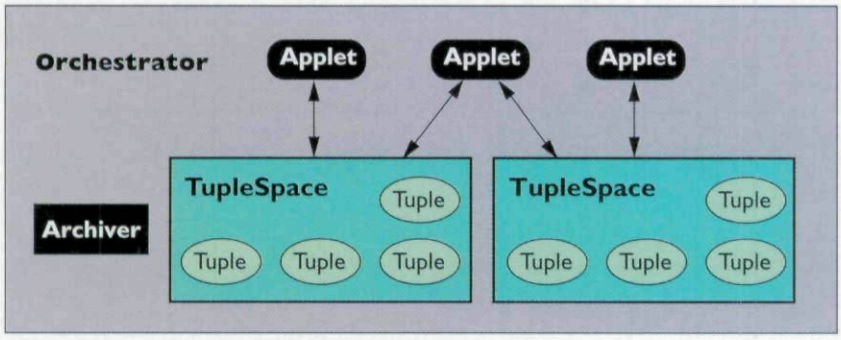

Figure 4. Orchestrator architecture

TupleSpaces are saved when the Orchestrator shuts down and restored when it starts up.

While describing the Orchestrator's programming interface in detail is beyond the scope of this article, we can offer a sense of how the interface is used to support the TourGuide's HotList feature, as in Figure 1. The HotList applet responds to a tour guide's mouse-click by extracting the tuple corresponding to the selected HotList item. The applet then uses the Orchestrator post command to send a URL tuple including the URL of the selected Web location to the appropriate TupleSpace within the Orchestrator. Arrival of this tuple causes a callback to a thread within each participating tourist or tour guide browser to respond by fetching the tuple and causing the referenced location to be loaded within the shared-display portion of the user interface.

The Orchestrator also provides TourGuide with whiteboard-style support for the spatial manipulation of shared objects in a WYSIWIS manner—a feature we used in our scenario to model the collaborative design of Web-page layouts between corporate executives and the content development agency. In this context, the action of dragging a shared object in one user interface results in the generation of the locus of events, or tuple postings, at a frequency sufficient to ensure the smooth changes of location of corresponding objects. The Orchestrator's ability to process several hundred tuples per second proved sufficient for this purpose.

Conclusions

Our earlier research strongly reinforced our appreciation of the value of application scenarios as a way to demonstrate the implications of multimedia technology on business communications [9]. Our experience with TourGuide is no exception. The TourGuide scenario was built on a platform providing general-purpose support for one style of collaborative multimedia applications but customized through scenario-specific multimedia content and appropriate custom-application functionality. The prototype implementation was an example of a solution to a real-world business communication problem many people can relate to.

TourGuide has proved useful for explaining the implications of available bandwidth on the quality of video that can be streamed in real-time to multiple users as part of a collaborative activity. It was a natural framework within which we could tangibly demonstrate the state of the art in terms of real-time media streaming using commercially available technology supporting playback at rates ranging from modem, to ISDN, to T1, to ADSL. The application also helps demonstrate various software capabilities supporting collaboration, including open audio channels, feedback mechanisms, and WYSIWIS design layout.

Beyond its pedagogical value, TourGuide has also helped demonstrate the viability of building distributed multimedia applications by synthesizing emerging Internet component technologies. Despite the mix of technologies we used to create this application, we were especially pleased with the seamless appearance of its user interfaces; avoiding applications external to Web browsers was a key contributing factor to the result. We learned that the core functionality required for collaborative applications, like TourGuide, could be supported largely through freely available Internet software technologies. We were also pleased by TourGuide's degree of platform independence.

Although we used third-party software to support most of the core TourGuide functionality, we

were not surprised to find functional holes. In particular, we knew of no third-party support for coordinating the behavior of distributed applets. Even so, the only substantial custom component we implemented for this purpose—the Orchestrator—took about three months to implement; the application support environment and TourGuide scenario demonstration required a similar development effort, from concept analysis through implementation and testing. Also worth noting, however, is that a considerable amount of effort was due to technology evaluation and testing. ■

References

1. Berners-Lee, T. WWW: Past, Present, and Future. *IEEE Comput. 29,* 10, (Oct. 1996), 69–77.
2. Carriero, N., and Gelernter, D. Coordination languages and their significance. *Commun. ACM 35,* 2 (Feb. 1992), 97–107.
3. England, P., Allen, R., and Underwood, R. RAVE: Real-time services for the Web. In *Proceedings of the 5th International World-Wide Web Conference* (Paris, France, May 6–10). O'Reilly & Assoc., Inc., Sebastopol, Calif., 1996, 1,547–1,558.
4. Evans, E., and Rogers, D. Using Java applets and CORBA for multi-user distributed applications. *IEEE Internet Comput. 1,* 3 (May–June 1997), 43–55.
5. Gutfreund, Y., and Nicol, J. The WWWinda orchestrator: A mechanism for coordinating distributed flocks of Java applets. In *Proceedings of Multimedia Computing and Networking 1997* (San Jose, Calif., Feb. 10-11). Society of Photo-Optical Instrumentation Engineers, Bellingham, Wa., 1997, pp. 295–302.
6. Hodges, M., and Sasnett, R. *Multimedia Computing: Case Studies from MIT Project Athena.* Addison-Wesley Publishers, Reading, Mass., ISBN 0-201-52029-X, 1993.
7. Maly, K., Abdel-Wahad, H., Overstreet, C., Wild, J., Gupta, A., Youssef, A., Stoica, E., and Al-Shaer. Interactive distance learning over intranets. *IEEE Internet Comput. 1,* 1 (Feb. 1997), 60–71.
8. Nicol, J., and Phuah, V. Experiences of constructing multimedia applications over ATM. In *Multimedia Services Over the Broadband Network: Current and Future Technologies,* F. Golshani, D. Minoli, and F. Groom, Ed. Adv. Bd. International Engineering Consortium, Chicago, Ill., 1996, pp. 365–374.
9. Phuah, V., Nicol, J., and Gutfreund, Y. ATM to the desktop: Impacting modern business communications with broadband technology. *Telematics and Informatics (Special Issue on Multimedia Technologies, Systems, and Applications) 14,* 1, (Feb. 1997), 5–25.

John R. Nicol (nicol@gte.com) is a senior principal member of technical staff at GTE Laboratories Inc. in Waltham, Mass.
Yechezkal S. Gutfreund (gutfreund@acm.org) is an independent consultant in Boston.
Jim Paschetto (jpaschetto@gte.com) is a senior member of technical staff at GTE Laboratories Inc. in Waltham, Mass.
Kimberly S. Rush (krush@gte.com) is a member of technical staff at GTE Laboratories Inc. in Waltham, Mass.
Christopher Martin (cmartin@gte.com) is a member of technical staff at GTE Laboratories Inc. in Waltham, Mass.