Dennis G. Brown
dbrown@ait.nrl.navy.mil
Naval Research Laboratory
4555 Overlook Ave. SW
Washington, DC 20375

Simon J. Julier
julier@ait.nrl.navy.mil
ITT Advanced Engineering & Sciences
2560 Huntington Ave.
Alexandria, VA 22303

Yohan Baillot
baillot@ait.nrl.navy.mil
ITT Advanced Engineering & Sciences
2560 Huntington Ave.
Alexandria, VA 22303

Mark A. Livingston
markl@ait.nrl.navy.mil
Naval Research Laboratory
4555 Overlook Ave. SW
Washington, DC 20375

Lawrence J. Rosenblum
rosenblum@ait.nrl.navy.mil
Naval Research Laboratory
4555 Overlook Ave. SW
Washington, DC 20375

Event-Based Data Distribution for Mobile Augmented Reality and Virtual Environments

Abstract

The full power of mobile augmented and virtual reality systems is realized when these systems are connected to one another, to immersive virtual environments, and to remote information servers. Connections are usually made through wireless networks. However, wireless networks cannot guarantee connectivity and their bandwidth can be highly constrained. This paper presents a robust event-based data distribution mechanism for mobile augmented reality and virtual environments. It is based on replicated databases, pluggable networking protocols, and communication channels. The mechanism is demonstrated in the Battlefield Augmented Reality System (BARS) situation awareness system, composed of several mobile augmented reality systems, immersive and desktop-based virtual reality systems, a 2D map-based multimodal system, handheld PCs, and other sources of information such as external data servers.

1 Introduction

As computers have become smaller and more powerful, portable, even wearable, high-performance computer systems for augmented and virtual reality have become feasible. State-of-the-art laptops and small-form-factor PCs boast processor, memory, disk, and graphics hardware that rival supercomputers from only five years ago. These PCs can be built into wearable (but still bulky) computer systems for individuals that provide information through a head-worn display. The full power of these systems is realized when these systems are networked with one another and with remote information servers. For example, a user walking down a street might be able to see the locations of other users, up-to-date information about prices in shop windows, and even email (Feiner, 2002). The focus of our research on the Battlefield Augmented Reality System (BARS) (Julier, Baillot, Lanzagorta, Brown, & Rosenblum, 2000; Livingston et al., 2002) is on the problem of developing information systems able to provide users with **situation awareness,** that is, data about the environment and its contents.

The problem of distributing data to users of wearable systems is somewhat unusual. Most research into distributed virtual reality focuses on the support of common, consistent virtual worlds that replace the real world, typically maintained by a small number of users. However, the objective of BARS is to support a consistent information space that augments the real world, one that does not need to provide a simulated replacement for the real world. There-

Presence, *Vol. 13, No. 2, April 2004,* 211–221

fore, data objects tend to be less complicated, containing mainly semantic information instead of detailed geometry and textures, and updates may occur less frequently than in virtual environments. Furthermore, BARS requires a unified architecture that allows transport of general state information that can be used for many purposes, such as the obvious task of distributing augmenting information, as well as more general uses such as "remote control" of applications, in which a remote device such as a PDA controls another computer. This system must also handle the poor network connectivity that can sometimes be encountered in military operations. Given these parameters, we have developed a robust, flexible, and general event-based networking infrastructure for data distribution. The mechanism builds upon three techniques: distributed databases, pluggable transport protocols, and a high-level management technique known as *channels*.

The structure of this paper is as follows: The problem statement and a survey of existing literature are given in Section 2. Section 3 describes the event distribution system. Representative performance measurements are presented in Section 4. Conclusions and future work are discussed in Section 5.

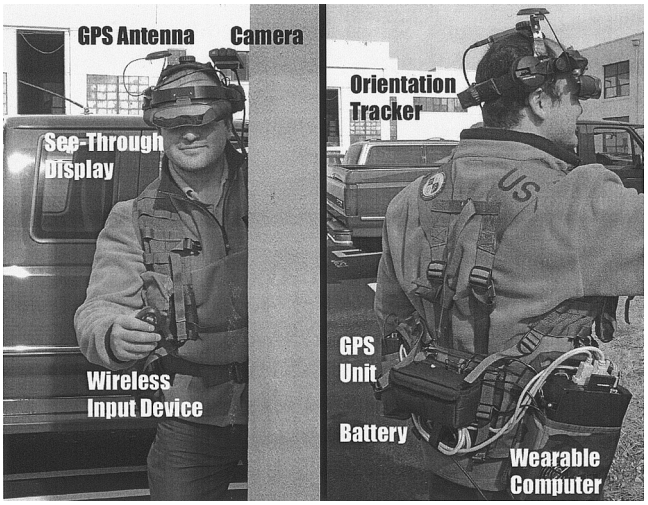

Figure 1. *The BARS Wearable.*

2 Problem Statement

The Battlefield Augmented Reality System (BARS) is a collaborative mobile augmented reality system designed to improve the situation awareness of, and the coordination among, members of a team of mobile users. Improving situation awareness means that each user obtains a better understanding of the environment through enhanced sensory perception. The types of data include the names of buildings, routes, objectives, and the locations of other users. While short-range radio communications can accomplish much of this data presentation, the passive and natural display paradigm of augmented reality makes the internalization of the information by an individual faster and easier.

The hardware of a prototype wearable system is shown in Figure 1. It consists of a wearable computer, a display, and a tracking system. The computer is responsible for generating 3D graphics and spatialized audio in real time. The generated graphics are shown on an optical see-through head-mounted display. The tracking system determines the position and orientation of the user's head, using a Global Positioning System (GPS) receiver for position and a solid-state inertial navigation system for orientation. A camera can be used for tracking and sending video reports to a base station. The user operates the system using a cordless mouse and a wrist keyboard. Wireless 802.11b networking is used for data distribution and GPS corrections. Future mobile Augmented Reality (AR) systems will communicate using a hardened military networking system (North, Bryan & Baker, 1999), but it is probable that any such system will still be vulnerable to connectivity and bandwidth complications in urban areas, and the distribution system design reflects that consideration.

The BARS mobile user sees computer graphics superimposed on or next to the real objects they are intended to augment, in addition to status information such as compass direction and messages from other users. Figure 2 shows a view using the system, in which another BARS user is augmented and is following a virtual path. This application has a number of characteristics that impact the distribution of information and events between users:

Figure 2. *A sample augmentation.*

- The objective is to provide relevant information, not a consistent virtual world. The BARS environment is populated by a set of objects that are self-contained entities and other types of discrete data. Each object can be relatively simple, representing a building type and location, an avatar to symbolize another user, the location of a hazard, and so on. It is not necessary to transmit complicated geometric objects or behaviors—only semantic information. The latency in the update of an object or an entity is a secondary consideration.
- Data distribution between users can be heterogeneous. Different users might perform different tasks and thus have different information requirements.
- The distribution system should facilitate collaboration between users. In addition to environmental data, the distribution system must support the propagation of metadata such as task assignments, objectives, and personalized messages.
- Users should have the ability to create reports and update entities in the database. For example, a user might observe that an environmental feature (such as a vehicle) is not where the database indicates it should be. The user should have the ability to move the object to its correct location.
- Network connectivity is unreliable. As a user traverses a terrain, reception strength and bandwidth may vary.

Given the potential importance of distributed systems for virtual and augmented reality, a great deal of research has been carried out by various research groups.

The Naval Postgraduate School (NPS) has been working on large-scale distributed virtual environments for over a decade, their first version of NPSNET being shown in 1991. The data distribution in the current version, NPSNET V (Capps, McGregor, Brutzman, & Zyda, 2000), uses replicated data and the model-view-controller paradigm to maintain it. Users of the system see a view of the entities (models) drawn by a rendering system. These entities are modified by the protocols (controllers) that translate network messages into state changes. New protocols can be loaded at run-time to extend the behaviors of entities. The packets are sent via multicast, and the system allows many protocols to share a single multicast socket. Network traffic is controlled with an area-of-interest manager.

The Distributed Interactive Virtual Environment (DIVE) system (Frécon & Stenuis, 1998) is designed to scale to many participants while maintaining a high level of interactivity at each site. It is based on a peer-to-peer architecture that uses reliable multicast to maintain a database that is replicated at each peer. Using a hierarchical partitioning of the database, only part of the database can be replicated at some peers. To maintain highly interactive rates, some copies of the database may be updated faster than others, but there are mechanisms to ensure equality over time.

MASSIVE-2 (Greenhalgh, 1996) uses the concept of a local object, or artifact, and remote views of that artifact. These artifacts are paged in and out of remote applications. A spatial model of interaction is used, in which a remote application has a focus that defines a subgroup of the artifacts to be paged in and out, instead of copying the entire database. Artifact state changes are distributed using reliable multicast.

Not all work has been solely for immersive virtual environments. Some collaborative AR systems have been built, typically for small groups of colocated users. One example is the Shared Space system (Billinghurst, Baldis, Miller, & Weghorst, 1997) for computer-supported collaborative work using AR.

Typically, users operate closely with one another and latency requirements are extremely high. Another system designed for augmented and virtual reality is COTERIE (MacIntyre & Feiner, 1996), which provides useful semantics through its notions of Distributed Shared Memory and Callback Objects. Distributed Shared Memory allows many applications to share the same data objects, while Callback Objects allow applications to be notified of changes to those data objects—an event-like mechanism. The Studierstube system (Reitmayr & Schmalstieg, 2001) extends the concept through its use of an extensible scenegraph of application objects that is distributed to all users.

3 The BARS Event Distribution System

First, some terms will be defined as they are used in this discussion. A *session* consists of one or more *applications,* or program instances, which may exist in any number on one or more machines on a network. Each application uses a core set of libraries to maintain a local database of objects and to communicate over a network. Applications may also include modules to read data from sensors, draw the augmented display, and perform other tasks, depending on the purpose of the application. The local database is a copy of a master database that is shared between all applications on the network. The distribution system is responsible for selectively replicating the master database in all applications.

The distribution system is based entirely on the concept of *events.* Events are used to instantiate objects (in effect, to transmit a view of a database between systems), to update existing objects, and to provide other non-database status information such as protocol management data. The event distribution system is based on three components: replicated object repositories, event transporters, and communication channels. These components will be described below, as will bridge applications, which communicate with outside information systems, and some other uses of the event distribution system.

3.1 Replicated Object Repositories

All of the data for a scenario is stored in an object repository. The data consists of the mostly static models of the physical surroundings (buildings, streets, points of interest, etc.), dynamic avatars that represent users and other entities, and objects created to communicate ideas, such as reports of enemy locations, routes for users to follow, and digital ink. This repository is replicated in whole or in part for each application.

When an application starts, it loads an initial set of objects from a number of sources, including saved databases, other applications already running on the network, and files specified on the command line. The initial set of objects typically consists of street labels, landmarks, building information, and other terrain-like information, as well as an initial set of objectives, routes, and phase markers for the current task. Since the user is given a database to start, and everything else in the wearable system is self-contained, the user will have a working AR system even if all network connectivity is lost during an operation.

Although network limitations may hamper wireless communications for the mobile users, there are few limitations on the base users. Base users are those who use stationary systems and are not mobile, such as users at fixed command centers. Their applications run on stationary Virtual Reality (VR) systems such as a desktop computers, 3D workbenches, and immersive VR rooms. Using the same distribution system, they can have high levels of detail and interaction by taking advantage of the increased bandwidth for replicating more objects and seeing change events at a higher frequency.

3.2 Event Transportation

The heart of the event transportation system is the Object and Event Manager. The Object and Event Manager is responsible for dispatching events within an application and distributing those events to remote applications. When the Object and Event Manager receives an event, it places that event on an asynchronous event queue. An event-dispatching thread delivers the event to all the listeners that are subscribed to receive the specified event type. The event-dispatching mecha-

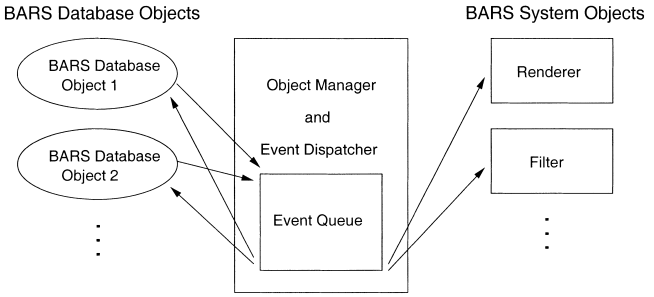

Figure 3. *Event distribution within an application: arrows show event movement.*

nism maintains two sets of data—the set of valid event types, and the set of listeners registered for each event type. Because the event system is based on the Java Abstract Window Toolkit event model (Sun Microsystems, 2003) the Reflection Application Programming Interface is leveraged to achieve these steps. Each event type is implemented in its own class. For each event type, a listener interface is defined, to be implemented by interested objects. When an object is registered with the Object and Event Manager, its instance type is queried and it is registered to receive all event types for which it has implemented listener methods. It is possible to dynamically extend the set of events and listeners handled by the dispatcher at run-time.

The following is an example of the life of an event within an application that tracks a user's position. The user's position is updated by calling a method in the user object to set its pose based on data gathered from tracking devices. In turn, this method creates an event that encapsulates the change in pose. The event is enqueued at the event dispatcher. The dispatcher later sends the event to all listeners, including the initial object itself, as well as other system components (such as the graphics system, which updates the viewpoint). Note that the object's pose isn't set until it receives the event back from the dispatcher (the alternative is to set the position at the same time the event is sent)—this way, the order of events is preserved. Figure 3 shows the flow of events within an application.

The propagation of events within a single application instance has been described above. This event mechanism was extended to allow many separate applications to trade events by creating Event Transporters. Event Transporters allow Object and Event Managers in different application instances to send and receive events over Internet Protocol (IP). Figure 4 shows the flow of events between applications. If an event is tagged as distributed, an Event Transporter serializes the event and broadcasts it to other applications. The Event Transporters in remote applications synthesize the event object and dispatch it on those applications' event queues. The system uses several types of transporters based on IP multicast, the Lightweight Reliable Multicast Protocol (LRMP) from INRIA (Liao 1998), and a combination protocol called the Selectively Unreliable Multicast Protocol (SUMP) that combines IP multicast and LRMP. Typically, application instances use SUMP on the local network. To communicate outside of the local network (where multicast is typically filtered out) TCP/IP transporter and bridge are used (described later in Section 3.4). Because of the connectionless nature of IP multicast, the distribution is robust—the network connection can be unreliable and the user application will still function, although without network updates at some times.

As events are created, they are tagged "reliable" or "unreliable" designating how they should be transported. Object creation and deletion events are always sent reliably. Object changes are sent reliably or unreliably based first on whether the modification is relative to other changes or not. Relative changes have an ordering and each one is important, so those are sent reliably. Non-relative changes, such as the constant updates of a user's position, are mostly sent unreliably since if one were missed, the next would overwrite it anyway. Periodically, these non-relative changes are sent reliably. This policy makes the assumption that the implementation of IP networking in a real operation may drop IP packets often, making reliable multicast expensive, and so events are not sent reliably unless they are thought to be truly necessary.

3.3 Channels

The problem with the event distribution mechanism described above is that all events for all objects

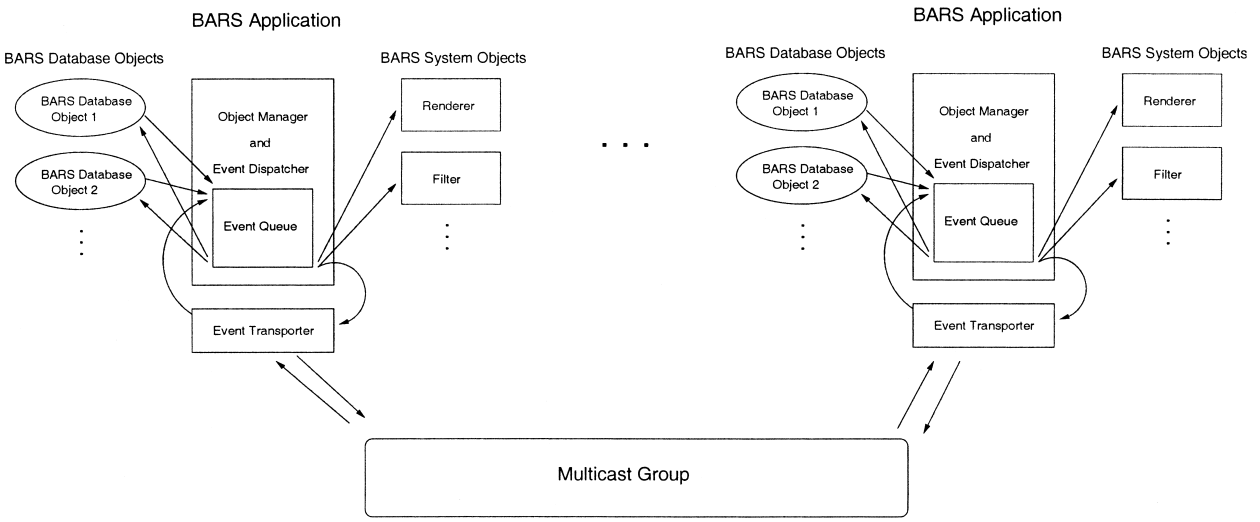

Figure 4. *Event distribution between applications: arrows show event movement.*

would be broadcast to every single application. Creating copies of every object for every user and updating those replicas would swamp the network with information that would be irrelevant for many users. To overcome this problem, the database is only partially replicated in each application instance.

In creating this replication mechanism, the uses of BARS drove the policies. One condition to consider is that a mobile user can see only so far and can deal with information only in a relatively small radius, so a spatial area-of-interest mechanism was considered. It is *not* necessarily the case that a mobile user only cares about objects that can be seen from his or her current position in the real world; for example, a mobile AR application may include an overhead map mode in which the user can zoom out to an arbitrary height to observe objects within a huge radius around the current position. However, it seems that there would be few situations in which a mobile user would request information about objects farther away, at the horizon for example, so for most situations, a simple area-of-interest mechanism is reasonable.

Another condition is the type of information that is being distributed. Even if some objects are near a mobile user, they may not be important and might only cause distraction. Alternatively, the objects may indeed be too far away to be seen, but very important, such as with possible sniper locations. For these cases, a simple area-of-interest mechanism isn't sufficient. In an earlier paper (Julier, Lanzagorta, Sestito, Rosenblum, Höllerer, & Feiner, 2000), a filtering mechanism for mobile augmented reality was described. This filtering mechanism operates on the local object database within an application instance. It does not show users objects in which they have no interest in order to reduce display clutter. In practice, it simply hides objects from the user—it does not actually control whether or not the application instance holds replicas of these objects or receives events related to these objects.

Keeping these situations in mind, *channels* have been developed. The term is overloaded in the literature, but in this system, a channel is a set of related objects. It is implemented as an instance of an event transporter and a multicast group designated for that transporter. An application can join an arbitrary number of channels and create new channels, until all available multicast groups are allocated. Figure 5 shows a single application using two channels.

One example of a channel is a set of objects in a certain spatial area. As users move from location to location, they can join and leave channels based on spatial areas. Another example is the set of hazardous objects;

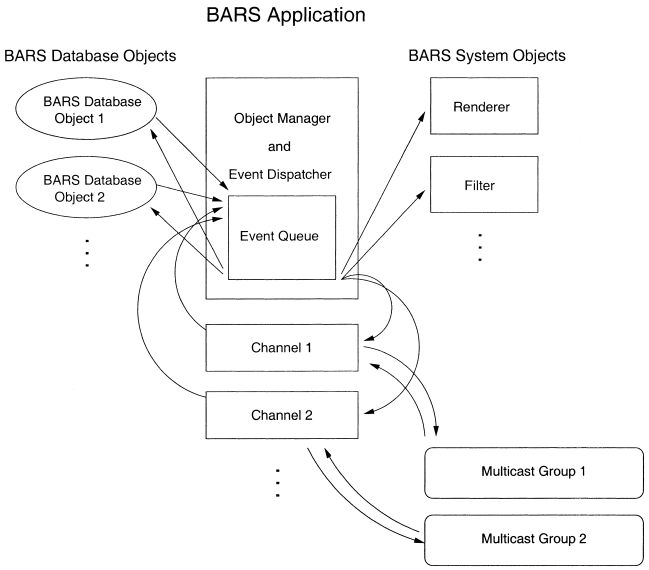

Figure 5. *Application joined to two channels: arrows show event movement.*

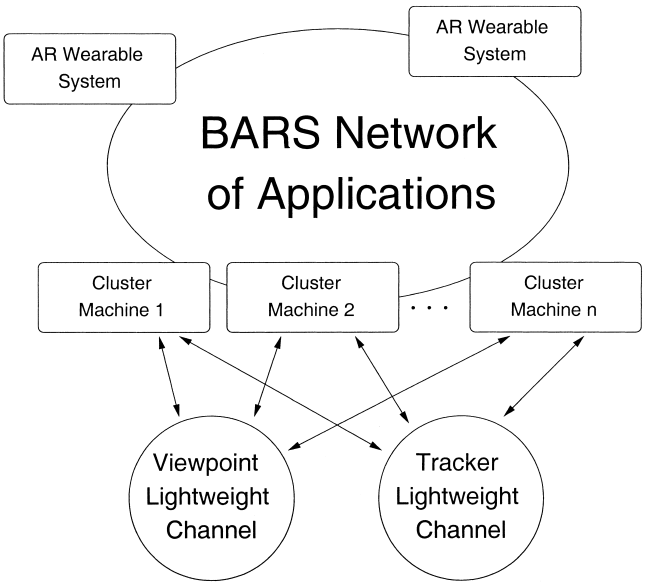

Figure 6. *Lightweight channels allow machines to share certain data, bypassing the regular distribution system.*

while in the previous example the application instance would replicate only objects nearby, the hazardous objects channel could cover a larger area, but only include those hazards. Also, BARS incorporates several interaction modules that produce subsequent objects. For example, one interaction module is responsible principally for real-time, interactive geometric construction (Baillot, Brown, & Julier, 2001). It allows users to collaboratively place points and build new objects from those points; in this case, the intermediate points would not be visible to other users because they are placed in a channel joined only by the constructing users. Other users would see only the final objects. Another interaction module lets a user draw digital ink for interpretation by a multimodal interaction system—this ink is turned into new objects or user-interface commands. In this case, the application instance of the user drawing the ink would be placed in a separate channel, joined by the application to interface with the multimodal system. The ink is placed in this channel so that other users will not see these sketches out of context.

In some cases, a set of applications may need to bypass the general event distribution system. "Lightweight channels" allow specific parts of an application to communicate directly over the network. This mechanism is used to coordinate a specific data set on many machines at interactive (30 updates per second or higher rates), such as in a cluster of eight machines driving a stereo four-wall immersive VR room. In that case, the user's viewpoint must be well coordinated between the machines, because it can be very disorienting to the user if the walls of the room show differing viewpoints even for very short amounts of time. For such applications, the viewpoint, and also the tracked device data (such as wand position), are sent over lightweight channels to only the cluster machines. At the same time, the distributed database continues to be maintained among the cluster machines and every other application in the session using the regular data distribution system as described earlier in this section. While the regular distribution system is not fast enough to maintain interactive rates when distributing the viewpoint or tracker data, it can handle database updates across the cluster machines and the rest of the session applications at reasonable rates (usually well over five updates per second). More performance data will be given in section 4. Figure 6 shows how a VR cluster works with the distribution system.

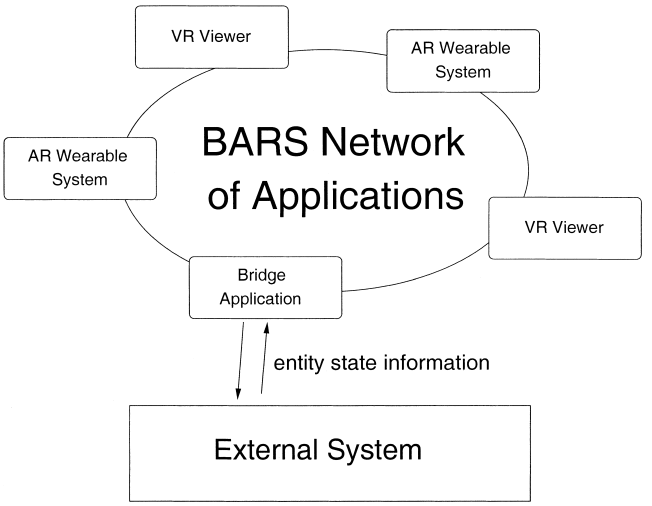

Figure 7. *A bridge application allows applications in a BARS session (two wearable systems, two VR viewers) to interact with data from an external system.*

3.4 Bridges

As alluded to in the previous section with the multimodal interaction example, some applications communicate with external information systems. These applications are called *bridges*. They join both the BARS distribution system and an outside system and translate object creation and changes between BARS and the outside system. By maintaining maps between BARS objects and these outside objects, those objects can be represented in BARS and vice-versa. Figure 7 shows how a bridge application fits into a session. Two systems with which BARS can communicate are the Columbia Mobile Augmented Reality System (Höllerer, Feiner, Terauchi, Rashid, & Hallaway, 1999) and the Oregon Graduate Institute's QuickSet multimodal interface (Pittman, Smith, Cohen, Oviatt, & Yang, 1996).

A TCP/IP transporter for this distribution system was mentioned earlier. Although the system is designed primarily for IP multicast, sometimes the need to connect to networks that block multicast arises. For this case there is a simple TCP/IP bridge application that joins local multicast groups and creates socket connections to other applications outside of the local network through TCP/IP Event Transporters. This can be a bit of a bottleneck, but one TCP/IP bridge per outside application can be created if necessary, and effective creation of channels can help limit the amount of data transferred.

3.5 Other Uses

One final aspect of this event distribution system to be discussed is its flexibility. Although the system was designed to carry information about objects in the database, at its heart it is still a distributed event system. The same base event types and transporters are used for other purposes. One set of events are "metaevents" to configure the Event Transporters; a similar set of events configures channels. Another set of events is used to distribute the user interface of an application to remote systems. For example, when a new user is trying out the mobile system and is unfamiliar with the user interface, the mobile system can be controlled from another computer (usually a handheld computer or PDA) by sending and receiving events that are handled by the user interface system. Yet another set of events is used for evaluation test studies: to trigger the tests, the test administrator uses an application configured to control the test subject's application This control happens simply by registering a new set of events and listeners and using the existing event transport system.

4 Performance Measurements

Now that the mechanisms used to control the network resource usage by the applications have been described, the performance of the system will be examined. The first issue to consider is the impact of using serialized Java events instead of writing and reading raw data into network packets through using code tailored to each event type. The serialized event sizes measured in a simple test ranged from 776 bytes (for a typical object-attitude-change event) to 5919 bytes (for a creation event for a path of twenty points). If specialized but simple algorithms encoded the data, the attitude-change event would need at least 60 bytes: 48 bytes for the six doubles representing the attitude (three doubles

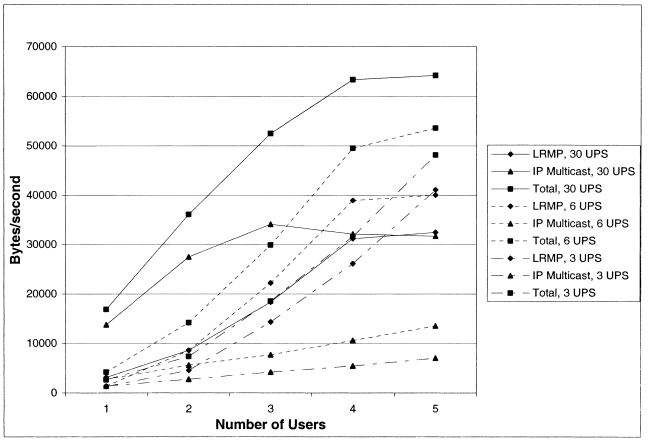

Figure 8. *Network usage at session start.*

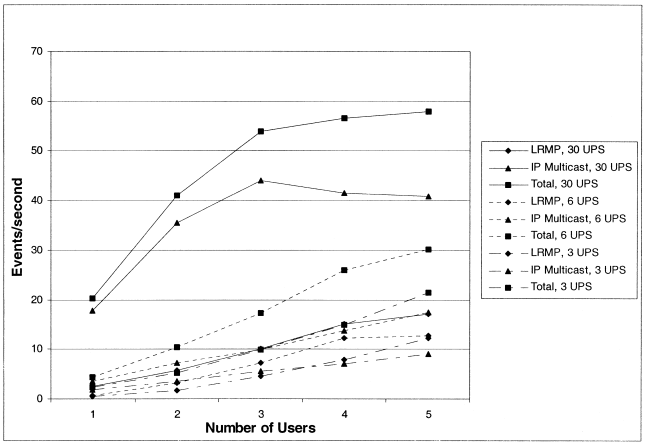

Figure 9. *Event throughput at session start.*

each for position and orientation), plus 12 bytes for three long integers specifying the source, target, and type of change. The path-creation event in this example would require at least 492 bytes: 12 bytes for three long integers to specify source, target, and type of creation, plus 480 bytes to encode twenty three-dimensional positions that make the points of the path. So, it appears that using serialized Java events inflates the event sizes by an order of magnitude over a simpler hand-coded system that would be harder to extend. For this research in augmented reality, having the flexibility to add new event types at run-time without any specialized code is important, so the penalty is accepted.

Next, distribution throughput in a set of test cases designed to simulate typical usage scenarios will be analyzed. In each test case, there is an existing session of one to five users. The session uses SUMP (the combination of LRMP and IP Multicast transporters) to transport events over the network in a single channel. Each of the users has created six notational objects (five point objects and a line of many points). A new user joins this session using 802.11b wireless networking and measures distribution system throughput in both bytes per second and events per second (since event sizes vary quite a bit) for the two transporters. These measurements are taken just after the new application starts up and after it has been running for five minutes. The test is repeated for three different user-position-update rates (how often the users send their positions to the network): 30 updates per second (UPS), 6 updates per second, and 3 updates per second. Thus, the independent variables are the number of users in the session, UPS rate, and time of measurement. The dependent variables are bytes per second and events per second measured by the new user. In reading the test summary, the reader will notice that the graphs show upper limits of data throughput rates that do not approach the theoretical upper limits of 802.11b. The measured throughput is affected not only by total network load but also by the speed of the machine collecting data, because it needs to process each event. Optimizing the implementation and using a faster machine may produce higher absolute numbers, but the scalability properties will remain the same.

As an application starts, it queries the existing applications in the session for distributed database objects. To provide redundancy in case of network outage, each application sends its set of distributed objects using reliable events (so in this case, they travel over LRMP). This technique needs to be improved, however, because it provides much duplicate data, and grows geometrically in network usage with the number of users; the LRMP data points in the graphs in Figures 8 and 9 show this effect. The applications also continuously send, at the specified rate, positional updates, primarily over the unreliable transport (IP Multicast). When the

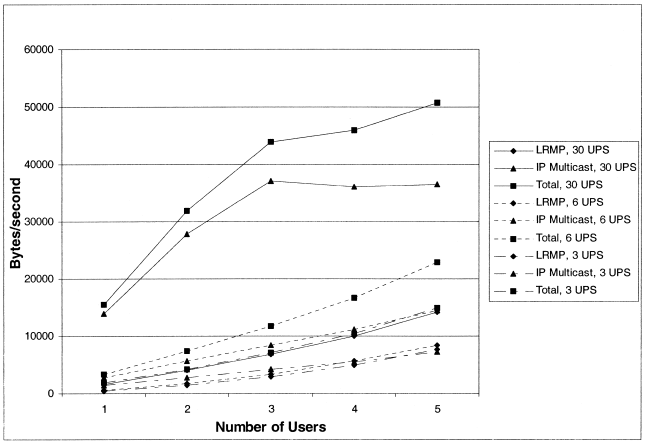

Figure 10. *Network usage over time.*

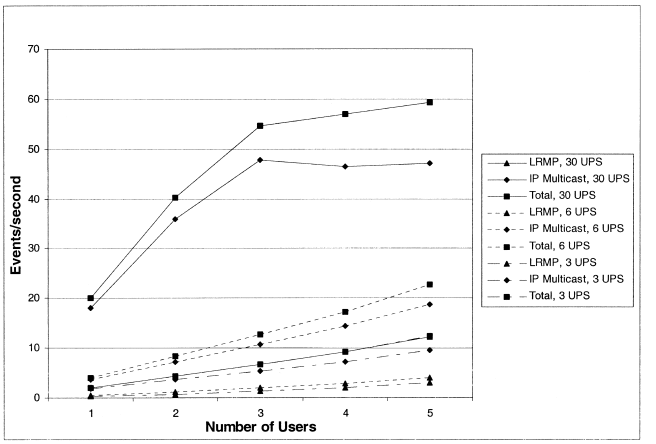

Figure 11. *Event throughput over time.*

existing users are sending thirty positional updates per second, the distribution system on the data gathering machine appears to become saturated at four users. The IP Multicast traffic grows linearly with the number of users, as expected, except where it actually decreases as the IP Multicast packets are dropped in favor of the LRMP packets. This phenomenon can be seen with as few as three users using the highest update rate in the test.

As the graphs in figures 10 and 11 show, after the session has been running for five minutes at the lower update rates, the total network usage (LRMP and IP Multicast combined) grows only slightly more than linearly with the number of users, as the small, unreliable updates begin to dominate over time. However, with high update rates for position, the network remains saturated well after the initial database duplication; remember that every tenth positional update is sent reliably, so even after the initial database duplication, there is still significant traffic using LRMP.

These measurements show that, in the test case of six updates per second, the distribution system easily kept up with five users over time. This capacity may seem insufficient, and with more optimization it can be greatly increased, as discussed earlier. However, for one driving scenario—being able to see the locations of friendly forces—it is more than enough for a small team. It is clear, however, that using serialized Java events is a heavyweight way to share data, and that a better optimized networking component using the basic architecture presented could provide much better throughput in a version of BARS constructed for actual operations.

5 Conclusions and Future Work

An event-based data distribution system implemented for BARS, a mobile augmented reality system, has been presented. It addresses some requirements encountered that developers of networked virtual environments have not addressed, such as working on unreliable network connections, handling many types of users through its channels, working well in both high-bandwidth (immersive VR) and low-bandwidth (mobile AR) applications, communicating with outside systems through its bridges, and being flexible enough for non-database data distribution.

For future work, the channel concept needs to be developed further. Intelligent algorithms to automatically create channels and join applications to those channels would be the next step. These algorithms would take into account such factors as: database object types; the user's tasks, location, and short-term interests; and network conditions. Also, bridge applications to widely-used military information systems will be created. Another area to consider is the problem of "partitioned"

data networks (Reijers et al., 2002). This condition arises when a mobile user (or set of mobile users) is out of contact for prolonged periods of time. During this time, multiple state updates have occurred and it is necessary to synchronize the systems again.

References

Baillot, Y., Brown, D., & Julier, S. (2001). Physical model authoring using mobile computers. *Proceedings of the 2001 International Symposium on Wearable Computers,* 39–46.

Billinghurst, M., Baldis, S., Miller, E., & Weghorst, S. (1997). Shared space: Collaborative information spaces. *Proceedings of HCI International 1997.*

Capps, M., McGregor, D., Brutzman, D., & Zyda, M. (2000). NPSNET-V: A new beginning for dynamically extensible virtual environments. *IEEE Computer Graphics & Applications, 20*(5), 12–15.

Feiner, S.K. (2002). Augmented reality: A new way of seeing. *Scientific American, 286*(4), 48 – 55.

Frécon, E., & Stenuis, M. (1998). DIVE: A scaleable network architecture for distributed virtual environments. *Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), 5*(3), 91–100.

Greenhalgh, C. (1996). *Dynamic, embodied multicast groups in MASSIVE-2 (Technical Report NOTTCS-TR-96-8)*. Nottingham, UK: The University of Nottingham, Department of Computer Science.

Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., & Hallaway, D. (1999). Exploring MARS: Developing indoor and outdoor user interfaces to a Mobile Augmented Reality System. *Computers and Graphics, 23*(6), 779–785.

Julier, S., Baillot, Y., Lanzagorta, M., Brown, D., & Rosenblum, L. (2000). BARS: Battlefield Augmented Reality System, Presentation made at *NATO Information Systems Technology Panel Symposium on New Information Processing Techniques for Military Systems*, Istanbul. October.

Julier, S., Lanzagorta, M., Sestito, S., Rosenblum, L., Höllerer, T., & Feiner, S. (2000). Information filtering for Mobile Augmented Reality. *Proceedings of the IEEE 2000 International Symposium on Augmented Reality*, 3–11.

Liao, T. (1998). *Light-weight Reliable Multicast Protocol Specification* (Internet-draft). Institut National de Recherche en Informatique et en Automatique (INRIA). Retrieved July 27, 2003, from http://webcanal.inria.fr/lrmp/draft-liao-lrmp-00.txt

Livingston, M. A., Rosenblum, L., Julier, S., Brown, D., Baillot, Y., Swan, J., et al. (2002). An Augmented Reality System for military operations in urban terrain. *Proceedings of the Interservice/Industry Training, Simulation and Education Conference 2002*, 868–875.

MacIntyre, B., & Feiner, S. (1996). Language-level support for exploratory programming of distributed virtual environments. *Proceedings of ACM Symposium on User Interface Software and Technology* (UIST 1996), 83–95.

North, R., Bryan, D., & Baker, D. (1999). Wireless networked radios: Comparison of military, commercial, and R&D Protocols. *Proceedings of the 2nd Annual UCSD Conference on Wireless Communications.* Retrieved February 9, 2004, from http://www.adiesa.aeema.asn.au/documents/WirelessNetworkedRadios-ComparisonofMilitaryCommercialandRDProtocols.pdf

Pittman, J., Smith, I., Cohen, P., Oviatt, S., & Yang, T. (1996). QuickSet: A multimodal interface for military simulations. *Proceedings of the 6th Conference on Computer-Generated Forces and Behavioral Representation*, 217–224.

Reijers, N., Cunningham, R., Meier, R., Hughes, B., Gaertner, G., & Cahill, V. (2002). Using group communication to support Mobile Augmented Reality Applications. *Proceedings of the 5th IEEE International Symposium on Object-oriented Real-time Distributed Computing (ISORC 2002)*, 297–306.

Reitmayr, G., & Schmalstieg, D. (2001). Mobile Collaborative Augmented Reality. *Proceedings of the IEEE 2001 International Symposium on Augmented Reality*, 114–123.

Sun Microsystems, Inc. (2003). Java API Documentation. Retrieved July 27, 2003, from http://java.sun.com/docs