Latency-Driven Distribution: Infrastructure Needs of Participatory Entertainment Applications

Farzad Safaei, Paul Boustead, Cong Duc Nguyen, Jeremy Brun, and Mehran Dowlatshahi
Smart Internet Technology, Cooperative Research Centre, University of Wollongong

ABSTRACT

This article evaluates network and server infrastructure requirements to support real-time flows associated with networked entertainment applications. These include the state information flow to update the status of the virtual environment and immersive communication flows such as voice, video, gesture, and haptics communication. The article demonstrates that scaling these applications to large geographical spreads of participants will require distribution of computation to meet the latency constraints of the applications. This latency-driven distribution of computation will be essential even when there are no limitations on the availability of computational resources in one location. The article provides detailed results on distributed server architectures for two of these real-time flows, state information and immersive voice communication. It also identifies a generic set of requirements for the underlying network and server infrastructure to support these applications and propose a new design, called switched overlay networks, for this purpose.

INTRODUCTION

The convergence of Internet and entertainment is poised to radically alter both the nature of entertainment applications and the underlying networks supporting them. Networked entertainment is likely to lead to new forms of group interaction. The degree of interactivity may range from simple Web-style point-and-click (e.g., to get information about a performer in an iTV application) to more substantive and immersive interaction that involves virtual participation and presence in a networked virtual environment (e.g., a multiplayer game). The latter category is the subject of this article and hereafter referred to as *participatory* entertainment (PE).

Participatory entertainment applications differ from the usual Web access in fundamental ways. Many users interact with the application (and through it with each other) in real time and across a large geographical span. Unlike the Web, which is primarily about retrieval of *precomputed* content, the visual and audio content associated with PE is dependent on actions of participants, *created in real time,* and highly dynamic.

PE applications often require a number of conceptually distinct but interrelated traffic flows. *State information* flows represent the actions of participants (moving, rotating, shooting, etc.) and are used to update the status of the virtual environment. As the level of immersion increases, other real-time flows, such as voice, video, gestures, and haptics, may be needed to create a better communication environment.

As discussed later, these flows have distinct requirements in terms of network and server designs. Nevertheless, they are real-time flows, and the key issue for their proper distribution and computation is managing latency. As the geographical spread of participants increases, the unavoidable rise in propagation delay will make controlling latency of paramount importance.

RESOURCE- AND LATENCY-DRIVEN DISTRIBUTIONS

Let us refer to distribution of processing load in response to scarcity of resources as *resource-driven distribution* (RDD). A common example is grid computing. To make the obvious point, RDD only makes sense if the available resources in the current or default location are insufficient.

There is another reason for distribution of computation: latency constraints. When controlling latency is critical, running the application in a single location — even with unlimited resources — may lead to unsatisfactory performance. We refer to this situation as *latency-driven distribution* (LDD). Note that LDD becomes critical only when the size of a network (in terms of propagation delay) exceeds the acceptable latency constraints of the application.

In this article we demonstrate that LDD will be essential for large-scale deployment of PE applications. We consider two different types of PE computation: that associated with state information, and voice and video communication. Analytical

0163-6804/05/$20.00 © 2005 IEEE

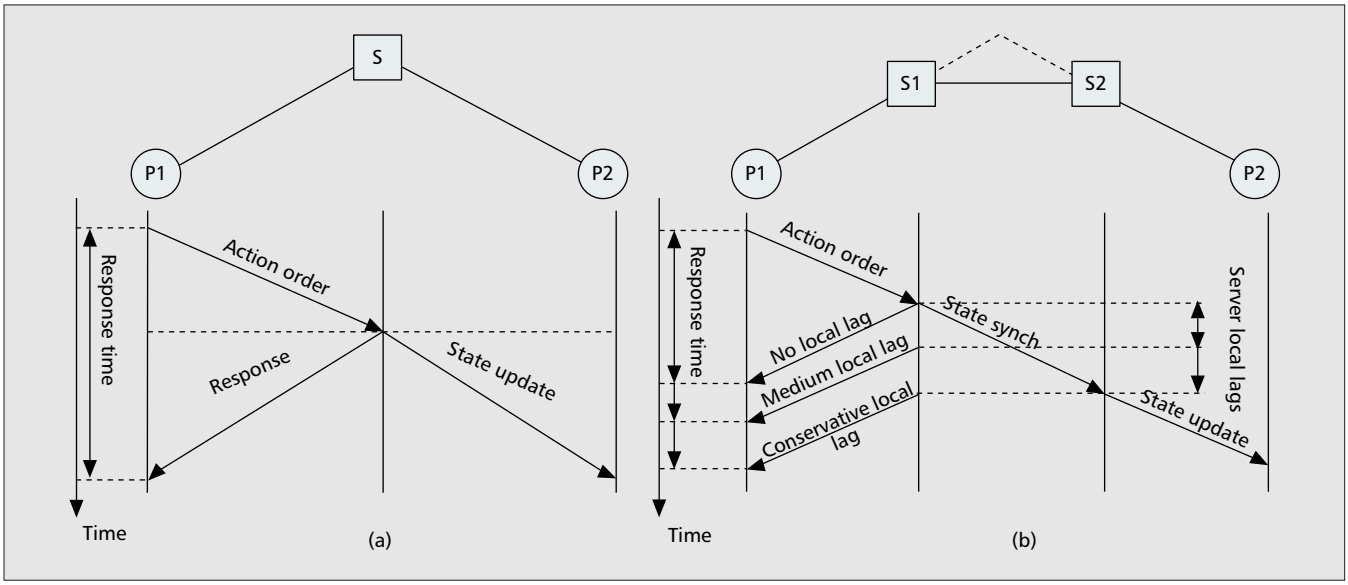

Figure 1. *a) Central and b) distributed servers for state information processing and the associated time diagrams.*

and simulation results are presented to show that the performance of a single server architecture will not be satisfactory even under ideal conditions (abundance of bandwidth and processing capacity, no packet loss or jitter). We then develop suitable models for LDD of servers in each section and show that under a reasonably broad set of assumptions, the improvement in performance is quite significant. We identify common ingredients for LDD and propose a new infrastructure that can simplify the task of writing PE applications by providing generic functions that support LDD. We believe that in conjunction with RDD technologies, such as task dispatching and load balancing over a sever cluster, the proposed LDD-enabled infrastructure can provide a powerful platform for supporting the next generation of entertainment applications over the Internet.

Responsiveness and Consistency

The current instances of PE applications, such as multiplayer games, often use a central server for processing state information. Large geographical participant spread, however, may deteriorate application responsiveness to user inputs with a single server. For this reason, most subscription-based commercial multiplayer games deploy independent virtual worlds in different locations to reach their clients.

Computation of a new state in response to the invocation of actions by participants involves modification of certain variables associated with the virtual environment. Consider the example of Fig. 1a with two participants, P1 and P2, and a central server, S. When P1 performs an action, an action order is emitted toward S, which makes a decision about this action and then sends an answer back to P1. Let us define the *response time* as the participant's waiting time for the server's reply. It can easily be shown that for this simple example the response time is bounded by the network delays from P1 and P2 to S.

LDD of state servers can improve response time. Figure 1b shows a distributed version of the previous example where two servers, S1 and S2, are located closer to the participants. Both servers are authoritative and synchronize their states by exchanging state synchronization packets. When P1 invokes an action, the action order is processed by S1. Assume, for the time being, that S1 responds to P1 immediately and then sends an update to S2. Clearly, the response time from the perspective of P1 is improved. However, the states of S1 and S2 remain inconsistent while the synchronization message is on the way. If, during this period of inconsistency, S2 makes a decision that is incompatible with S1's decision, the virtual environment may enter a *paradoxical* state. For the virtual environment to make sense, paradoxes must be either healed or avoided, as described below [1]. Both strategies require servers S1 and S2 to be time synchronized and to timestamp every decision they make.

Rollback (or timewarp): In this case, servers S1 and S2 save their virtual environment states at regular intervals called *checkpoints*. After making a "bad" decision because of a short-term inconsistency, the server detects its error once it receives the timestamped state synchronization message. It rolls back to the latest compatible saved checkpoint and recomputes the correct game state. Note that although paradoxes are fixed, they still occur and may be perceptible.

Server local lag: Before sending their response to participants, servers wait a certain time interval, called a *server local lag* (shown in Fig. 1b). In perfect network conditions (no packet loss or jitter) a server local lag equal to the network delay between S1 and S2 would avoid any paradoxes. However, local lag increases the response time and to some degree negates the impact of server distribution.

Latency-Driven Distribution of State Servers

Our investigations show that effective distribution of state information processing can lead to improvements in response time. To this end, we

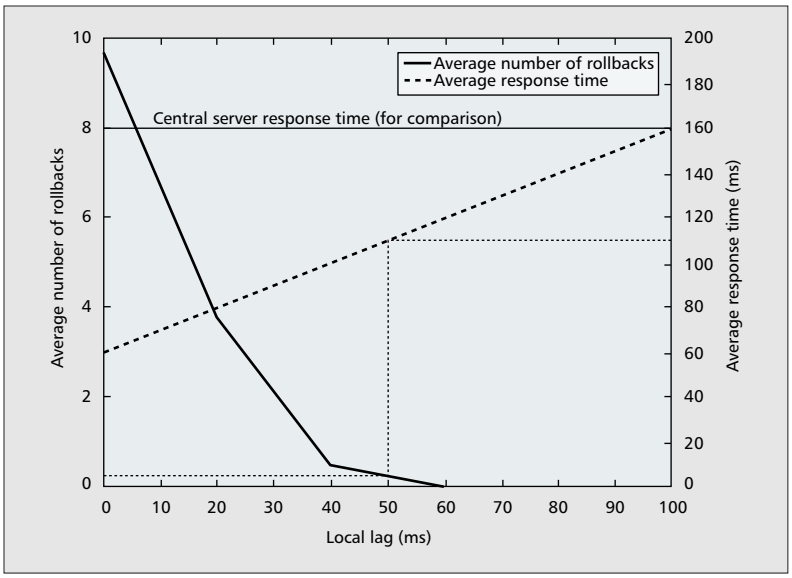

Figure 2. *Tuning local lag–response time and rollback.*

have selectively used the above techniques in conjunction with the following observations. All the forthcoming results were computed using a distributed game simulator developed by us. The same network topology (4 servers, 12 players) was used in every run simulating an average of 120 s of game. Figures were drawn based on the average of 100 simulations with identical local lag conditions and random initial seeds [2].

Tuning the Local Lag — Conservative high values for the server local lag will result in complete paradox avoidance, which might be overkill. The aim should be to find the right balance between *probability* of paradox appearance and its *impact* on the perceived utility of the application. For a given genre of PE applications and particular network conditions, it might be possible to tune the local lag between the extremes of Fig. 1b to achieve optimum performance.

Figure 2 shows the variations of the average number of rollbacks forced by the appearance of paradoxes, and the average response time when the local lag increases. For this experiment, the conservative value for local lag based on maximum network delay between servers is 100 ms. However, for this scenario a lower value such as 50 ms would result in a negligible number of rollbacks but significant improvement in average response time (110–160 ms). For comparison, distributed servers provide a better response time than an optimally placed central server under most conditions.

Unbinding State Parameters — In current distributed applications, all virtual world state parameters are synchronized using the same scheme. However, different parameters of the virtual world may represent totally different concepts with dissimilar requirements in terms of response time and paradox avoidance. In other words, not all actions are equal. For example, it may be beneficial to provide good response time to avatar movements with little or no local lag, knowing that inconsistency of the location parameter rarely results in a paradox. On the other hand, an avatar's life state parameter, if not treated consistently, could often lead to undesirable outcomes. It is therefore important to tailor the balance between responsiveness and consistency for each category of action *independently*.

Figure 3 presents the results of simulations where the local lags for moving and shooting actions have been independently increased on vertical and horizontal axes, respectively. The contours in Fig. 3a represent the number of rollbacks, those in Fig. 3b the response time. It is clear that decreasing the local lag for moving actions has a minimal positive influence on rollbacks and a huge negative impact on response time. On the other hand, decreasing the local lag for shooting actions provides sizeable reduction of rollbacks with little impact on the response time. Therefore, in this case we might apply small values of local lag to moving and larger values to shooting actions. For example, if the moving and shooting local lags are set at 20 and 50 ms, respectively, we achieve much

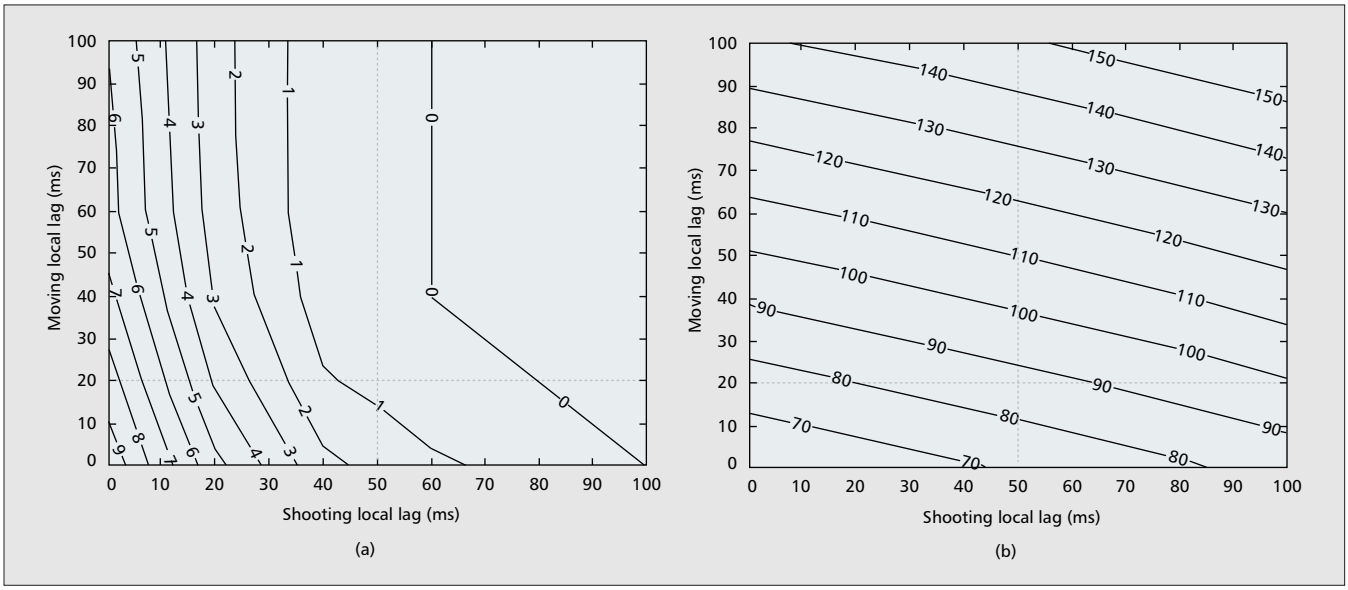

Figure 3. *a) Rollbacks and b) response time due to shooting and moving local lags.*

better response time (85 ms compared to 110 ms) with no appreciable increase in number of rollbacks, compared to a situation when both local lags are set at 50 ms (Figs. 2 and 3).

Controlling Network Delay between Servers — The network delay between the distributed servers can be controlled by the application provider. In Figs. 1a and 1b the application provider may have little to no control over the path between participants and servers, but it may be able to provide a better-quality route for interserver communications.

The above observations enable us to envisage a new model for LDD of state parameter processing for PE applications:

- Developers may be able to categorize different actions/parameters of the virtual environment based on required response time, consistency, and paradox impact.
- The PE application provider has access to a set of distributed servers around the Internet. These servers may be wholly owned by the application provider or hired from other server providers. The (virtual) servers can communicate with each other either using best effort Internet routed paths or, in some cases, through provisioned network paths such as a virtual private network with quality of service (QoS) guarantees.
- With support from the LDD "middleware" (to be described later), the application will be able to distribute authoritative state processing modules for each category of action with tailoring of local lag parameters based on the action category and the measured network delay between servers. Not all modules need to be distributed in the same way. Some that require good responsiveness may be distributed widely, while others that demand strict paradox avoidance may be more centralized.

Multimedia Group Communication

In the previous section we explored the ramifications of geographical scaling of PE applications with respect to state information processing and demonstrated that LDD could play an important role in enhancing participants' interactions. In this section we consider the creation of real-time multimedia content for natural group communication within a PE application. For brevity we focus on voice, but similar arguments would also apply to video and gesture communications.

Several research groups have investigated suitable models for voice communication in networked virtual environments (e.g., [3]). Our research team has developed an immersive voice communication environment for this purpose called the Dense Immersive Communication Environment (DICE). DICE can scale to crowded virtual spaces even when the access bandwidth of each client is limited.

DICE is immersive because participants can hear a realistic and personalized mix of others' voices in their "hearing range" in perfect harmony with the visual representation of speakers (spatial placement with respect to the listener). This requires real-time delivery of all the relevant voice streams within the hearing range and spatially placing the source of each voice at a point that perceptually matches the virtual location of the speakers with respect to the listener. All of this is highly dynamic as avatars move in the virtual environment. The audio content for each participant, therefore, has to be *created in real time from a dynamic set of dispersed sources*.

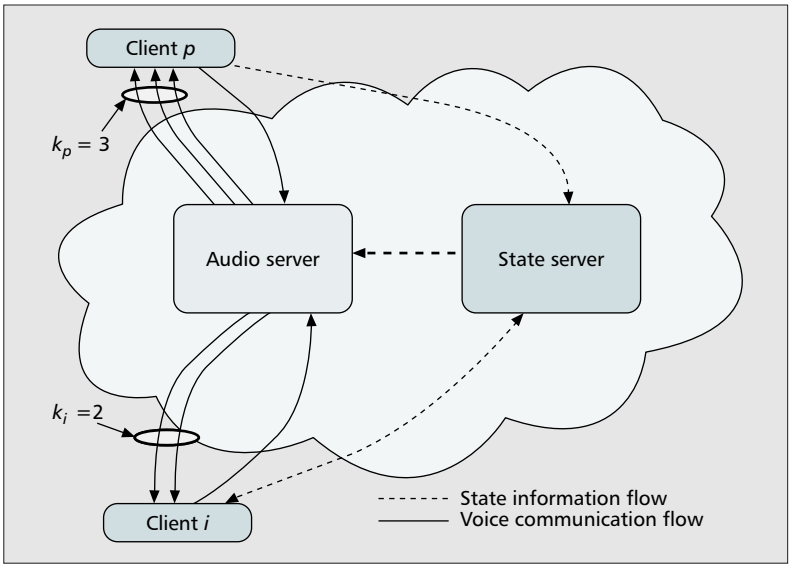

Figure 4. *DICE with a central audio server.*

Central Audio Server

Assuming that DICE is desirable for some PE applications, various models can be envisaged for its Internet-wide delivery [4]. For several reasons, including access bandwidth constraints, scalability, and privacy, we have focused on a server-based model, the simplest of which is to use a central audio server. Many different roles can be envisaged for this server. One possible approach that has been developed by us is shown in Fig. 4. The voice of each participant is captured and sent to the central audio server as a mono stream. The audio server also obtains information pertaining to location and status of avatars from the state information server (separation of these servers is conceptual, and in some cases it might be suitable to run both functions on the same hardware).

We assume that the audio server is aware of the access bandwidth limitations of each client. Typically, the state information exchange consumes a large portion of this bandwidth, so we can only support a limited number (k_i for participant i) of voice streams in the downstream direction. For example, only a maximum of three voice streams can be sent to client p in this figure (k_p = 3). When the number of avatars in p's hearing range is greater than three, the audio server groups these into three separate clusters and performs a partial audio mixing operation for each cluster. The server also calculates the *center of activity* of each cluster. This is the location of an imaginary audio source from which the cluster mix should emanate. This information is then sent to the client, and the client renders the audio scene by spatially placing each mixed audio at its center of activity. By using intelligent methods for

clustering and center of activity calculation, giving higher weight to nearby avatars, it is possible to create perceptually accurate audio scenes even when k_i is rather small.

The central audio server is obviously a processing bottleneck and a single point of failure. RDD techniques could be used to tackle these issues. But the server also introduces additional delay due to sending and receiving voice streams to/from the audio server, which becomes significant over a large-scale infrastructure.

With a single server, the network delay is dependent on the relative position of the central server with respect to the locations of participants. For example, if we choose to minimize the total average delay, the best location for the audio server would be the nearest possible server site to the *center of mass* (in terms of network delay) of the participants whose avatars are not isolated.

An application provider could choose a server site according to this criterion based on the best knowledge on distribution of potential participants. However, these estimates are seldom accurate, and may render the selected location ineffective. Changes in the physical distribution of participants can also happen due to time zone differences on a daily basis or on a longer timescale due to unpredictability in the uptake of service.

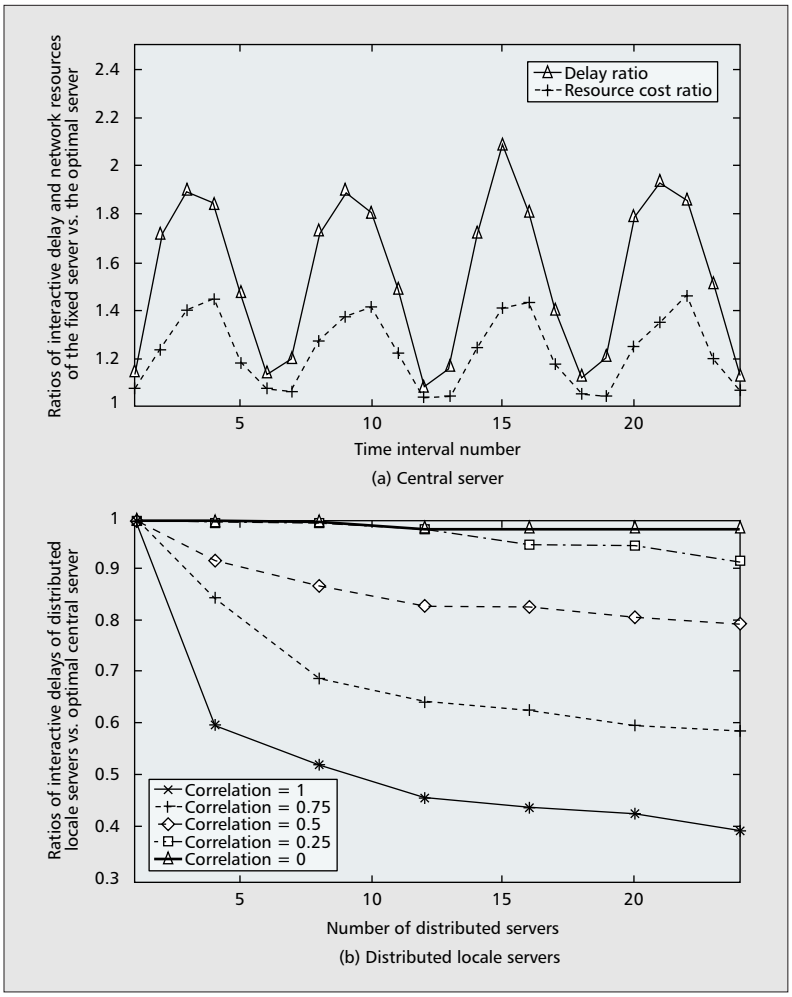

Figure 5. *a) The effect of changes in participant distribution on interactive delay and network resource usage; b) the effect of number of servers and correlation on latency.*

We have developed a simulation environment that creates both the physical and virtual worlds. The network is a transit-stub graph of 600 nodes, comprising three transit domains corresponding to North America, Europe, and Asia. Potential audio processing servers are chosen and participants distributed around this network. Uniform or cluster distributions are used to populate the virtual world with avatars.

Figure 5a shows the simulation results for total group communication delay based on a given distribution of participants in North America, Europe, and Asia that changes on a 4-hourly cycle [5]. As can be seen, the total group delay can increase by as much as 100 percent even when the initial server location is optimal. The network resource usage also increases as both the upstream and downstream audio streams have to be shipped over longer distances to/from the server.

In other words, an optimal audio server location can have a significant impact on latency reduction. However, in practice this location is not fixed. If the application provider has access to a set of potential sites, the location of audio processing may be altered in response to variations in distribution of participants [5]. When the spread of participants is over a large geographical span, however, it is best to use LDD of the audio server.

Latency-Driven Distribution of the Audio Server

Again, we assume that the PE application provider will be able to hire a distributed virtual server infrastructure with sufficient resources in key locations associated with their customer base. To improve the delay performance of DICE, we have investigated two possible distributed architectures for the audio server: distributed locale servers and distributed proxies.

Distributed Locale Servers — In this approach the virtual world is partitioned into small chunks called *locales*. One of the servers among the distributed set will be assigned to this locale to perform the audio server functions described earlier. To minimize latency, the physical location of this *locale server* has to be optimal with respect to those participants whose avatars are currently in this locale. If two adjacent locales are assigned to two different servers, the servers may need to exchange some voice streams belonging to those avatars in the boundary that are in the hearing range of each other.

We have developed optimal models for partitioning of the virtual world into locales and then assigning a server to each locale with the aim of minimizing the total interactive communication delay. We have not assumed any resource limitation on each of these servers. Hence, it is possible to assign one server to more than one locale or indeed to the whole virtual environment if no appreciable improvement is gained by LDD. We have observed that under a very broad set of conditions, LDD is beneficial [6].

The interactive delay improvement in this architecture is especially significant when people from a particular geographic region aggregate as a group in the same locale in the virtual world. In [4] a *correlation* parameter has been defined to model the relationship between the distribu-

tion of participants in the real world and that of their corresponding avatars in the virtual world. If the correlation parameter is high (close to one), people who are geographically close also tend to gather together in the virtual world (e.g., due to language or cultural reasons). If the correlation parameter is low (close to zero), members of a virtual crowd are distributed more or less uniformly across geographical regions.

Figure 5b shows that the improvement in interactive delay using a distributed locale server architecture, if any, depends on two factors:

- The number and spread of available locale servers
- The correlation parameter described above

This is because:

- The application provider must have access to a sufficient number of potential locale servers in different regions for LDD to be effective.
- When the correlation is high, choosing a server geographically close to participants of that locale can improve the delay significantly.

The figure shows that reduction in group communication delay can be as high as 60 and 20 percent *compared to an optimally located central server* for correlation values of 1 and 0.5, respectively.

The location of the locale server will be optimally selected for the current composition of participants in that locale. As time passes, however, movements of avatars between locales may make the current assignment suboptimal. A key challenge for the distributed locale server architecture, therefore, is to be able to reassign the locale servers in response to avatar movements (and reroute the associated audio flows after reassignment). This is one of the capabilities needed from an LDD-enabled infrastructure.

Distributed Proxies — The second approach for LDD of the audio server assigns a group of participants in a given geographical region to a nearby server called a *proxy*. Each proxy will perform the role of an audio server for its assigned participants. The proxy must also multicast the captured voice from its assigned participants to all other proxies that need this audio for their clients.

In some sense, distributed proxies are the "dual" of distributed locale servers, one based on partitioning the physical world (into geographical regions) and the other based on partitioning the virtual world (into locales) and then assigning a suitable server to each partition. This duality also applies to network- and server-level complexity. With distributed locale servers, the server reassignment in response to movement of avatars is complex. In most other situations, however, audio streams are stable unicast flows between clients and servers or among the locale servers. In contrast, the distributed proxy architecture has no need for reassignment of proxies. However, movement of avatars in the virtual world is translated to reconfiguration of multicast flows between the proxies on very short time scales. For example, based on a random waypoint model for motion of avatars, it is shown that for 90 percent of proxies, more than 60 percent of their multicast trees must undergo changes after avatars move up to 3 percent of the size of the virtual world from their previous locations [7]. For details on latency improvements due to this architecture see [8, 9].

Infrastructure Design

As discussed in previous sections, LDD of applications requires knowledge and control of the *spatial location* of processing. This is in contrast to the more familiar RDD, where the actual location of processing is irrelevant and often hidden from the application. There are many RDD techniques and architectures. For example, scalable Web servers use Web switches to intercept and perform pattern-matching operations on packets arriving at a server cluster for balancing the load over the servers within the cluster.

Controlling the spatial location of processing for LDD applications dictates the need to have servers (or server clusters) distributed around the Internet available for use by the application. To avoid implementing a new infrastructure for each application, we are developing mechanisms for many LDD applications to share the same server and network infrastructure. This infrastructure borrows from existing concepts (e.g., grid computing and content switching) to scale applications at each individual location. However, we introduce the concept of a new switch to control distribution of processing between server clusters based on latency considerations of each application. Since the application itself is the only entity that can decide if the delay of a particular flow of packets is important, we need a switch that can be controlled by applications. As shown earlier, each application will have different requirements for geographical distribution of processing. Even so, we have identified a set of generic and elementary functions that would be required by many LDD models.

Figure 6 shows our proposed infrastructure design, referred to as a switched overlay network (SWON). Our description here will be brief; detailed information can be obtained from [10]. The LDD application provider will hire virtual server resources in suitable locations based on the predicted customer base. The servers themselves might be owned and administered by different entities, and of course each server in this figure may be a large cluster for CPU scaling purposes.

Within the virtual servers, the application resides on top of LDD middleware. Each server is connected to the Internet through a hardware switching element referred to as a *SWON switch* (Fig. 6). The SWON switch is essentially a specifically designed label switch that can switch packets based on two types of labels, forwarding and application labels. Naturally, most hardware functions can conceptually be implemented in software too. But by separating the SWON switch from the middleware, it is possible to relieve the servers of performing mundane packet-level functions.

The SWON switch provides intercluster switching resources to each application provider. It also performs other basic packet-level functions, such as duplication and deletion. Note that each SWON switch is shared by all the applications running on the server but allocates different label spaces to each and interacts with the LDD middleware using a control module.

From the perspective of LDD middleware (and by implication the applications), the collec-

The location of the locale server will be optimally selected for the current composition of participants in that locale. As time passes, however, movements of avatars between locales may make the current assignment suboptimal.

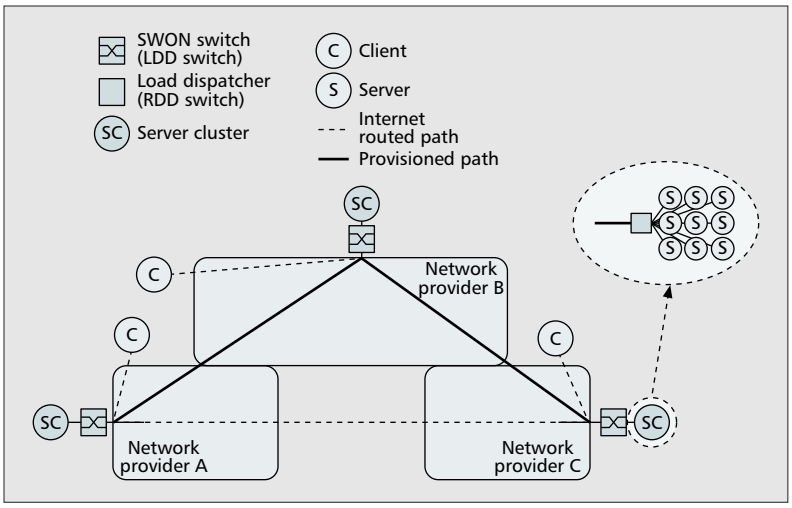

Figure 6. *The switched overlay network.*

tion of SWON switches is the network. The application can control and set up unicast and multicast flows between the distributed servers using the set of SWON switches that are collocated with the server clusters. The use of SWON switches is important because it is neither realistic nor prudent to allow applications to speak directly to Internet routers. This combination enables more rapid deployment of services without requiring advanced functions within the underlying network infrastructures. It also mitigates concerns on security and stability of the Internet through a clean segregation of functions between the network and server infrastructure.

CONCLUSIONS

Latency-driven distribution is likely to play a crucial role in scaling PE applications to a large geographical spread of participants. The computations associated with real-time flows of PE applications, such as state information and immersive voice communication, will be distributed based on specific requirements of each flow. The distribution is also required to be responsive to dynamics of the application, such as changes in the physical and virtual distribution of participants and variability of latency constraints on different actions or flows.

We have identified the generic requirements of an LDD-enabled network and server infrastructure for these applications. Our model enables the application provider to use the shared resources provisioned by various network and server providers, and deploy customized LDD designs based on the application requirements. We have developed example PE applications that are currently under customer trial and are building prototypes of our SWON architecture for their widescale deployment.

REFERENCES

[1] M. Mauve *et al.*, "Local-Lag and Timewarp: Providing Consistency for Replicated Continuous Applications," *IEEE Trans. Multimedia*, vol. 6, Feb. 2004, pp. 47–57.
[2] J. Brun, F. Safaei, and P. Boustead, "Tailoring Local Lag for Improved Playability in Wide Distributed Network Games," *Proc. Australian Telecommun. Networks and Apps. Conf.*, Sydney, Australia, 8–10 Dec. 2004, pp. 519–25.
[3] M. Radenkovic, C. Greenhalgh, and S. Benford, "Deployment Issues for Multi-User Audio Support in CVEs ," *Proc. ACM Symp. Virtual Reality Software and Tech. 2002*, Hong Kong, China, Nov. 11–13, 2002.
[4] P. Boustead and F. Safaei, "Comparison of Delivery Architectures for Immersive Audio in Crowded Networked Games," *Proc. 14th ACM Int'l. Wksp. Network and Op. Sys. Support for Digital Audio and Video*, Kinsale, Ireland, June 16-18, 2004, pp. 22–27.
[5] C. D. Nguyen, F. Safaei, and D. Platt, "On the Provision of Immersive Audio Communication to Massively Multiplayer Online Games," *Proc. 9th IEEE Symp. Comp. and Commun.*, Alexandria, Egypt, June 2004, pp. 1000–05.
[6] C. D. Nguyen, F. Safaei, and P. Boustead, "A Distributed Server Architecture for Providing Immersive Audio Communication to Massively Multiplayer Online Games," *Proc. IEEE Int'l. Conf. Networks*, Singapore, Nov. 2004.
[7] M. Dowlatshahi and F. Safaei, "A Recursive Overlay Multicast Algorithm for Distribution of Audio Streams in Networked Games," *Proc. IEEE Int'l .Conf. Networks*, Singapore, Nov. 2004.
[8] C. D. Nguyen, F. Safaei, and P. Boustead, "Performance Evaluation of a Proxy System for Providing Immersive Audio Communication to Massively Multiplayer Games," *Proc. 1st IEEE Int'l. Wksp. Networking Issues in Multimedia Entertainment*, *GLOBECOM 2004*, Dallas, TX, Nov. 2004.
[9] C. D. Nguyen, F. Safaei, and P. Boustead, "Comparison of Distributed Server Architectures in Providing Immersive Audio Communications to Massively Multiplayer Online Games," *Proc. Australian Telecommun. Network and Apps. Conf.*, Sydney, Australia, 8–10 Dec. 2004, pp. 499–505.
[10] P. Boustead, F. Safaei, and V. Nguyen, "Switched Overlay Networks (SWON): Switching Support for a Global Network of Virtual Servers ," Smart Internet Tech. CRC tech. rep., http://www.titr.uow.edu.au/swon/swon_architecture.pdf

BIOGRAPHIES

FARZAD SAFAEI (farzad@uow.edu.au) graduated from the University of Western Australia and obtained his Ph.D. in telecommunications engineering from Monash University, Melbourne, Australia. He has more than 15 years of experience in conducting and managing advanced research in the field of data communications and networks. Currently, he is professor of telecommunications engineering and director of the Centre for Emerging Networks and Applications at the University of Wollongong. He is also program manager of the Smart Internet Technology Cooperative Research Centre (CRC).

PAUL BOUSTEAD is a senior research fellow at the Telecommunications and IT Research Institute at the University of Wollongong. He is currently leading projects within the Smart Internet Technology CRC. He completed a Ph.D. at the University of Wollongong in 2000 in the area of label switching protocols for high-speed networks. His current research interests include network and server support for the delivery of distributed services over the Internet, network games, and content distribution networks.

CONG DUC NGUYEN received a B.Eng. degree in telecommunications (first class honours) from the University of Wollongong in 2000. Since 2001 he has been a Ph.D. candidate at the Centre for Emerging Networks and Applications, University of Wollongong. His current research interest is in the investigation of server and network architectures for the provision of immersive voice communications to massively multiplayer online games.

JEREMY BRUN is currently completing his Ph.D. at the Telecommunication and Information Research Institute at the University of Wollongong. He received his Master's in engineering studies from the University of Wollongong and his engineering degree from Supelec, France, in 2001. His reseearch interests include telecommunication network architecture with an emphasis on real-time network applications and game server distribution.

MEHRAN DOWLATSHAHI has been working as a research fellow at the University of Wollongong Smart Internet Technology CRC since 2003. He has received B.Sc., M.Sc., and Ph.D. degrees, all in electrical (telecommunication) engineering. His research interests are in overlay service architectures, IP telephony, performance analysis of communication networks, and peer-to-peer services and architectures.