A Survey of Application-Layer Networking Techniques for Real-time Distributed Groupware

Jeff Dyck

University of Saskatchewan

Ph.D Comprehensive Exam Survey

January 12, 2006

ABSTRACT

Real-time distributed groupware (RTDG) applications enable users to collaborate or compete in real time and over distance. RTDG includes network games, computer supported cooperative work (CSCW) applications, and collaborative virtual environments (CVE). These applications must send information over networks in real time in order to share information among group members. During transmission, the information is subjected to network loss, jitter, latency, and limited bandwidth, which can cause severe usability problems in these types of systems. One approach to improving performance is to design applications to make more efficient use of network resources – by applying application-layer networking techniques. This survey presents application-layer networking techniques that have been applied to RTDG, as well as comparisons with other applied areas to learn what needs to be done next in order to improve RTDG performance.

1. INTRODUCTION

Real-time distributed groupware (RTDG) is software that allows people to collaborate from remote locations in real-time using computers. Some examples of RTDG applications include shared whiteboards, shared desktop applications, and network games. Real-time distributed features are also being added to distance education systems (e.g., WebCT) and virtual office systems (e.g., Groove). The popularity and diversity of RTDG are increasing as new applications for collaboration are developed and collaborative features are added to existing applications.

RTDG works well on a LAN, where network resources are so plentiful that the network performance is not an issue. But the vision for RTDG is that it should be possible to use it from everywhere and by everyone, and to do so requires groupware to work over the Internet. The transition from LANs to the Internet introduces network problems that reduce usability and deter people from collaborating (Gutwin, 2001). The main network problems that affect usability are limited bandwidth, packet loss, and latency. Users refer to the effects of the network problems collectively as *lag*.

Lag appears to users as slow or delayed motion, large and unnatural jumps in graphical objects in the workspace, and inconsistencies in shared information. When lag is present, it takes longer to complete collaborative tasks and users begin working more independently (Gutwin, 2001; Pantel and Wolf, 2002; Quax et al, 2004). In studies of what users discuss while playing network games, the topic of lag and overall performance problems arises frequently (Wright et al, 2002).

One long term method to improve the lag problem is to improve the networks used by groupware applications. If all networks performed like LANs, there would be no network performance problems for RTDG applications. However, improvements to worldwide networks take a long time to make, due to the amount of infrastructure that needs to be upgraded, and the transition to better performing networks is happening slowly. Also, we can expect that new network technologies like wireless networks, satellite networks, and power line networks will continue to introduce new sources of lag. Although network improvements may one day remedy the lag problem, this will not be a reality for a long time to come. Until then, we must find ways of coping with the limitations of networks, and the only way to do this in the short term is by making better use of existing networks.

Application-layer networking techniques are techniques that can be implemented within applications and toolsets to make better use of network resources. These techniques help to make better choices for how information should be sent in order to meet the quality of service (QoS) requirements of the information. As a simple example, application programmers can decide whether to send information using TCP or UDP. TCP guarantees in-order delivery, but needs to acknowledge the receipt of information and retransmit information that does not arrive. UDP sends information unreliably, but does not have the overhead of retransmitting and acknowledging that TCP does, and therefore less latency and more throughput are possible at the expense of reliability. The best decision of which protocol to use depends on the QoS requirements of the information and the current conditions in the network, and choosing one over another is an application-layer networking decision. Additionally, we can apply networking techniques that take advantage of characteristics of the information being sent in order to improve performance. For example, when sending telepointer (remote mouse pointer) messages, which are a stream of [X, Y] coordinates, we can encode redundancy into the packets in order to increase reliability due to the small message size, or aggregate multiple telepointer messages into a single packet (Dyck et al, 2004). There are many application-layer networking techniques that can provide significant network performance benefits to RTDG applications; many have been applied to RTDG, but these techniques are scattered across various literature, and no comprehensive summary of all of these techniques exists.

This survey presents a set of application-layer networking techniques for RTDG. It follows a model for groupware networking (see Figure 1) that includes QoS (section 5), a set of application-layer networking techniques that are well-suited for delivering better performance to RTDG (section 6), and network monitoring (section 7). In this model, QoS is used to specify the requirements of the shared information generated by the application. The state of the network is continuously monitored so that timely network performance information is available. The QoS parameters and network state are then both considered to determine the most appropriate set of application-layer networking techniques to apply to deliver the QoS that is specified. Since testing is also critical to developing high performance application-layer networking for RTDG, a brief section on testing (section 8) is included.

The networking techniques presented in section 6 are of four types: encoding, routing, reliability, and scheduling. *Encoding* techniques affect the information generated by the application directly – by compressing it to reduce bandwidth requirements or by encrypting it to meet security requirements. *Routing* techniques determine the network path that is used to send information among users, which is affected by both the distribution architecture chosen for the system as well as the routing policy that is being used. *Reliability* techniques affect the probability that information will arrive uncorrupted and in the correct sequence, and network protocols and error correction techniques are the main methods for promoting reliability presented here. *Scheduling* techniques determine what information is sent, in what order, how it is assembled into packets, and at what rate, and the main techniques included in this survey are rate control, aggregation, scheduling, and filtering. The techniques described in the survey include techniques developed specifically for groupware, as well as techniques that are from other networking domains that are likely to be beneficial to groupware. Since there are techniques for compensating for delays that work at the UI and model layers of applications, a section on providing networking information to upper layers was also included.

The survey concludes with a commentary on the state of the art of application-layer networking for RTDG (section 9). This section describes potential future areas of work that would be particularly beneficial to this area.

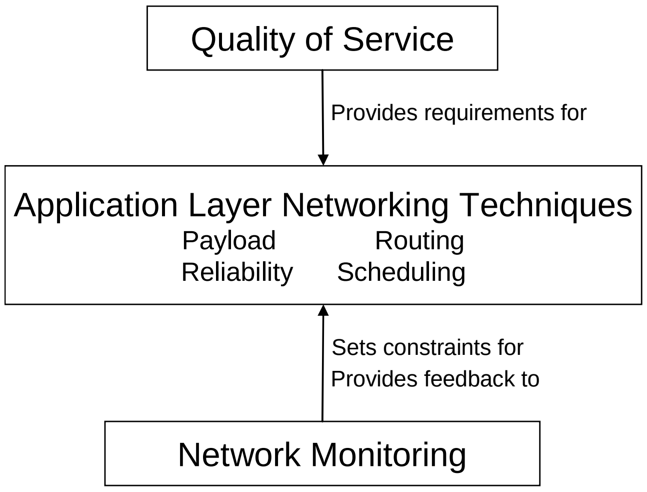

Figure 1: A groupware networking model that communicates application-layer requirements in the form of QoS, applies application-layer networking techniques to meet those requirements, and monitors the state of the network as observed at the application layer to fine tune the application-layer networking techniques.

2. RELATED SURVEYS

There are several surveys that are relevant to groupware network performance. The most closely related survey is Smed et al's survey (2002) of networking techniques used in computer games, which summarizes published work from military simulations, networked virtual environments, and networked games. This survey presents compression, aggregation, multicasting, interest management, and dead reckoning as the main techniques for improving network performance. It also presents an overview of resource allocation and discusses the tradeoff between responsiveness and consistency. Most of the techniques described in this survey are not included in Smed's survey.

Other related surveys are also relevant to subsections of this survey. Mills (1999) presents a general overview of groupware. Phillips (1999) conducted a thorough examination of groupware distribution architectures. There are several surveys of concurrency control methods, one of which is specifically for groupware (Greenberg and Marwood, 1994), and others are either general or specific to related areas (Kohler, 1981; Briot et al, 1998). Iren et al (1999) present a tutorial and broad survey of network protocols. Finally, Perkins et al (1998) surveyed packet loss recovery techniques for streaming audio.

3. RTDG OVERVIEW

Real-time distributed groupware (RTDG) allows remotely located people to collaborate using computers in real-time. This is distinct from asynchronous groupware, where group actions may not be seen until long after they occur. It is also distinct from single display groupware, where groups are located together and collaborate using a single screen. When compared with other types of groupware, the unique challenge to RTDG is the need to exchange time-sensitive information with remotely located group members.

3.1 Sub-domains of RTDG

RTDG includes three main areas which have their own origins and mostly separate communities; computer supported cooperative work (CSCW), real-time network gaming, and collaborative virtual environments (CVE). Groupware and CVEs are primarily research areas with relatively few commercial applications, while network games have been very successful, with thousands of commercial software releases; however relatively little academic research work is done that specifically addresses network gaming. Since games are generally proprietary and the networking techniques used in games are generally developed by game companies and not published, most of the techniques from games presented in this survey were found in documentation from game network libraries, books, magazine articles, web articles, and interviews, rather than from academic sources. CSCW, CVEs, and games share a common challenge to support real-time group activities within a distributed group, and it follows logically to consider their network performance challenges collectively.

3.2 Types of activity in RTDG

Greenberg and Gutwin provide a partial taxonomy of collaborative activities, which divides group work into two types: *taskwork*, which is the execution of tasks; and *teamwork*, which is the work of working together (Gutwin and Greenberg, 2000; Gutwin and Greenberg, 1999).

Taskwork affects the elements contained in the workspace that are shared by the group. For example, adding a shape to a shared whiteboard or destroying a guard tower in a simulation game are taskwork activities. Some of the important performance issues in supporting taskwork include consistency management and model layer distribution architectures, both of which have attracted considerable attention from the CSCW community (e.g., Phillips, 1999; Yang and Li, 2004). Other taskwork performance related areas, such as network transmission and security, have been addressed by the network gaming community (e.g., Dyck and Gutwin, 2006; Pritchard, 2001; Yan and Choi, 2001).

Teamwork includes the activities that are required to work together as a group, and are defined as the social and affective elements that make up group dynamics and by the *mechanics of collaboration* (Gutwin and Greenberg, 2000). The social and affective elements include things such as emotions and relationships among group members, which have an impact on the way that people collaborate (Grudin, 1988). The mechanics of collaboration include explicit communication, implicit communication, awareness, coordination of action, planning, monitoring, assistance, and protection (Gutwin and Greenberg, 2000). Supporting teamwork in groupware has attracted considerably less attention, and as a result, the performance issues remain relatively unexplored compared with the issues affecting taskwork.

The diverse and interactive nature of RTDG makes delivering high performance a much more complex problem than it is in other related domains (Gutwin and Greenberg, 1999; Gutwin and Greenberg, 2000). RTDG makes use of a wide variety of techniques to support group awareness, to enable coordination of tightly coupled activities, and to support both direct and implicit communication. These techniques have varying network performance requirements, and as a result, developing effective networking techniques for RTDG is a hard problem.

3.3 Related areas

The multimedia and networking communities have developed many techniques that can lead to better performance in RTDG systems. Many of these have not been applied to RTDG, but have been useful in related applied areas such as live broadcasts, video on demand (VoD), voice over IP (VoIP), real-time distributed systems, and conferencing.

The main difference between RTDG and other real-time distributed domains is that RTDG must support group interactions. Two of the more closely related distributed real-time domains include voice over IP (VoIP) and video on demand (VoD), which distribute time-sensitive information to a group of geographically separated people. Though groupware applications often contain video and voice (e.g., Baecker et al, 2004), they are distinctly different, since exchanging video and voice streams does not require support for group interactions. Another closely related area is real-time transaction processing, which shares some similar challenges to RTDG, but does not deal with supporting group activities either.

4. THE RTDG NETWORK PERFORMANCE PROBLEM

RTDG is frequently cited for its poor performance when used over real-world wide-area networks (e.g., Bhola et al, 1998; Sun, 2002; Vaghi et al, 1999). The performance problems in RTDG are primarily due to effects introduced by the network as information travels between users. The most critical problems are *latency*, which is the time required for information to travel between locations, *jitter*, which is variance in latency, *loss*, which results from network packets not arriving at their destination, and insufficient *bandwidth*. These problems are common in today's wide-area networks, and although networking advances are aiming to reduce these problems, the problems will be present for some time to come. In the meantime, RTDG applications must cope with latency, jitter, loss, and bandwidth limitations at the end points.

Poor performance in RTDG has been shown to have serious effects on users, including causing difficulties in coordinating collaborative actions (Vaghi et al, 1999; Gutwin, 2001; Park and Kenyon, 1999) and causing reduced collaboration (Gutwin, 2001). It has been shown that coordinating actions in real time becomes difficult when feedback is delayed, making tasks take longer to execute and resulting in more errors (Gutwin, 2001). Also, jitter has been shown to make gestural communication harder to interpret (Gutwin et al, 2003). The overall effect is that collaboration breaks down – groups tend to decrease collaboration and work more independently, and sometimes remove members with poor performance from the group or discontinue the collaborative session entirely (Gutwin, 2001). In network games, it has also been shown that one of the most commonly discussed topics in game chat is the performance of the application, which demonstrates that users are aware of and affected by performance problems (Wright et al, 2002), and several studies have confirmed these effects on gamers (e.g., Pantel and Wolf, 2002; Quax et al, 2004).

Delivering network performance in RTDG is a difficult task due to the diverse performance needs of techniques used to support interactions and situational factors that affect performance requirements (Gracanin et al, 2004). First, there are many different genres of RTDG, including games, whiteboards, conferencing systems, virtual offices, screen sharing systems, virtual classrooms, and collaborative virtual environments. Second, each application can have multiple interaction techniques that have diverse QoS requirements, such as telepointers, drawing tools, collaborative buttons and menus, and radar views. The applications also need to keep model layer data synchronized and these requirements vary based on the needs of the application. Third, performance requirements are affected by situational factors, such as proximity to other users, how tightly coupled the interactions with particular users are, and the level of awareness that a user wishes to maintain with their other collaborators. These complexities have often been insufficiently addressed when developing RTDG applications.

As a result, there is no single performance goal that will cover all cases, and no single application-layer networking technique will be suitable for the needs of RTDG. Instead, a diverse set of networking techniques is required, where appropriate techniques can be used situationally in order to meet the wide range of QoS requirements that RTDG demands.

5. QUALITY OF SERVICE

Quality of service (QoS) is used to define the performance requirements for the system in order to make it usable, and to guide techniques that are used to deliver the performance requirements. Many RTDG papers mention QoS and its importance (e.g., Marsic, 2000; Mathur and Prakash, 1995; Greenhalgh and Benford, 1999), but there are few that discuss how to model or deliver QoS. This may be because QoS in RTDG is complex compared to other classes of applications due to the wide variety and unconstrained nature of the tasks and interaction techniques that appear in RTDG, combined with the subjective and qualitative user perceptions and the intricacies of group interactions (Greenhalgh and Benford, 1999; Gracanin et al, 2004). As a result, QoS problems in RTDG are far from solved. This section summarizes the RTDG work that has been done to develop QoS models and requirements, as well as the mechanisms used to communicate, negotiate, and manage QoS.

QoS models include sets of *QoS characteristics* that are appropriate for a particular application type, as well as their associated *QoS parameters*. The most complete QoS model designed specifically for RTDG is described by Mathur and Prakash (1995). Their model includes four QoS characteristics: latency, jitter, packet loss, and asynchrony. They also define parameters associated with each of these characteristics. This QoS model captures some of the most critical QoS characteristics and parameters that are most frequently discussed in RTDG literature (e.g., Marsic, 2000; Gutwin, 2001). However, this does not represent a comprehensive QoS model, as there are many other characteristics that RTDG must consider, such as security, frequency, resolution, accuracy, precedence, and authenticity, which are described in the generic ISO/IEC QoS model and could be specialized for the purposes of RTDG.

The values of QoS parameters determine the performance goals of the system, and appropriate values are ones that deliver a good user experience without wasting resources. For example, for real-time voice conversations, users become annoyed by delays when they exceed 150ms (ITU-T, 2000), and this information has been used to guide QoS in VoIP applications. Due to the complexities of RTDG, it is less feasible to study and recommend as specific of values for QoS parameters. A limited amount of work has been done to determine the effects of different amounts of delays on users for specific tasks (e.g., Gutwin, 2001), but appropriate QoS parameter values for most RTDG situations remain unverified by user trials. Recently, several studies have been done to learn the effects of delays on players of networked games (Pantel and Wolf, 2002; Quax et al, 2004; Nichols and Claypool, 2004; Beigbeder et al, 2004).

One QoS challenge faced by RTDG is that of mapping user needs to network QoS requirements. Gracanin et al (2004) discuss this challenge and present an approach to mapping QoS among the various layers of abstraction in order to deliver QoS in a way that benefits the user experience while not wasting resources.

QoS managers coordinate the mechanisms involved in the delivery of QoS, including establishment, monitoring, alerts, maintenance, control, and enquiry (ISO/IEC, 1998). Most QoS managers that are included in RTDG systems are simple and are controlled by a central process (e.g., Greenhalgh and Benford, 1999). McCanne (1996) suggests that QoS management from the receiver side allows more flexibility for handling QoS requirements, and one example of a RTDG system that does this is GCSVA, which applies a receiver-driven approach to QoS management to a conferencing system (Beier and Koenig, 1998). Greenhalgh and Benford (1999) argue that RTDG applications have a more complex resource negotiation problem than video and voice applications, and that a central group QoS manager is better-suited for cooperative decision making.

The QoS management approaches for RTDG are different than ones for related areas because they have to adjust to the needs of different applications, scenarios, techniques, user preferences, and group dynamics. Greenhalgh and Benford (1999) describe a QoS manager for a CVE system that allocates resources to suit the application and situation based on application-specific settings and by using a spatial awareness model to determine level of interest among participants. This approach also allows quotas to be added to users and to applications or media streams, and manages the resources used by each. Content is delivered using a layered multicast model (McCanne et al, 1996), where each layer adds fidelity to the information, and layers are added or removed based on resources, quotas, applications, and the spatial awareness model.

6. APPLICATION-LAYER NETWORKING TECHNIQUES FOR RTDG

This section presents a set of application-layer networking techniques that can lead to better RTDG network performance. The descriptions of the techniques are brief, and are intended to highlight the purpose and uses of the techniques. The networking techniques are broken into four main categories: encoding, routing, scheduling, and reliability (see Figure 2).

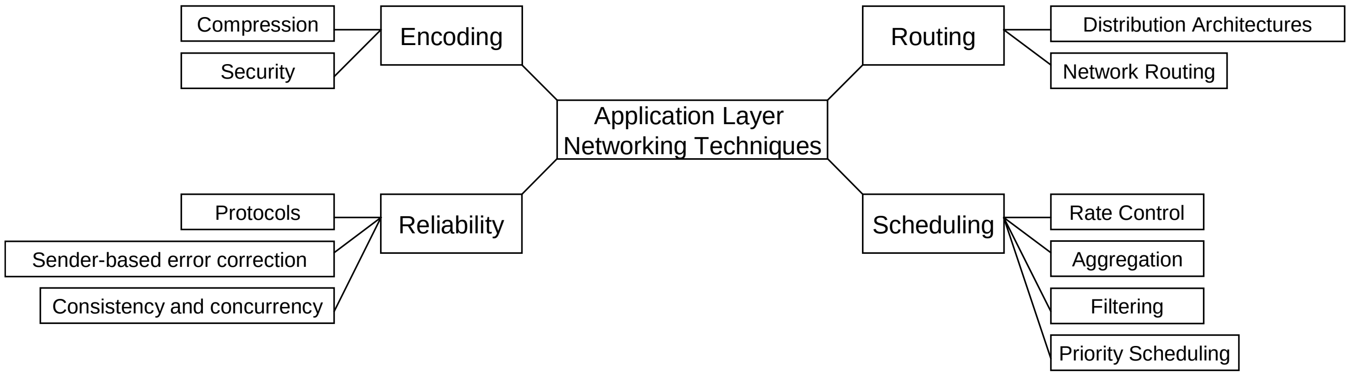

Figure 2: A taxonomy of application-layer networking techniques for RTDG.

6.1 Encoding

Encoding techniques for RTDG are applied to change the representation of the information being sent in order to increase security or decrease the payload size. Encoding techniques are broken into two subcategories, compression and encryption.

6.1.1 Compression

Much of the information exchanged by RTDG applications can be highly compressed to reduce the amount data sent over the network. *Compression* is not widely used in CSCW, but it has had limited use in CVE applications, and almost all network games use some form of compression to reduce network traffic. *Lossless compression* techniques are generally the most useful for RTDG (excluding video and voice data), as the type of data sent by RTDG applications generally does not tolerate partial loss in the way that images or sound can. CVEs have made use of MPEG-4 compression and its BIFS extension (Hosseini and Georganas, 2001), while the most common compression techniques used in games are delta and contextual (GarageGames, 2005). *Delta compression* sends updates to known absolute values rather than entire absolute values to save space. This technique has been used extensively in games, where only changes to the game state are sent in each packet. *Huffman compression* is a technique that is used frequently in games and works by replacing characters with bit sequences that have a specific code word length, assigning shorter bit sequences to represent more common sequences. *Contextual compression* techniques use characteristics of the information being sent to compress the information. This can result in improvements over generic compression techniques, but requires previous knowledge of information and additional effort to develop customized compression techniques. One example of contextual compression scheme used in CSCW appears in Dyck et al (2004), where a customized delta compression technique was used to encode telepointer messages, resulting in a 2:1 compression ratio over an optimal uncompressed representation, and as much as a 20:1 improvement over telepointer messages seen in other systems.

There are many other techniques that may be useful for compressing RTDG information, and more work is required in this area to determine which compression techniques should be applied to what types of information and under which conditions. In particular, different compression algorithms require different amounts of processing power, and produce different compression ratios. Making several compression algorithms available for different situations may be well-suited for delivering a variety of different QoS requirements, and adaptive compression techniques could be useful for optimizing the processor vs. bandwidth tradeoff under a variety of conditions.

6.1.2 Security

Security of information transmitted over the Internet is a concern that all networked applications face, and many solutions for ensuring security have been introduced. Some of the main security issues in RTDG networking include privacy, authenticity, tamperproofing, and anonymity. *Privacy* is enforced by some RTDG applications by encrypting data, but encryption can both increase bandwidth usage and processor load. Therefore, there is a tradeoff between security and performance. This tradeoff should be carefully managed by coupling information with security enhancing techniques that are appropriate. In RTDG, this can mean encrypting chat messages, which should be private and are not particularly latency sensitive, and leaving telepointer positions unencrypted because security is less important and they are latency sensitive. More work is required to develop QoS-aware encryption techniques that can meet a variety of privacy and performance requirements, and work is required to address the other issues of privacy for RTDG.

Games have been particularly motivated to address security issues, as some game players are determined to find ways to cheat and harass other gamers. Games have mainly had to prevent packets from being read, used, and generated by external applications, so encryption and authenticity techniques are common. However, they tend to only work as a deterrent, as game players can be sophisticated cheaters and it is common for them to reverse engineer a game to extract encryption techniques directly from the binary. This has created a race between game developers and cheaters, and games have had much experience with security as a result. Webb (2004) presents a survey of the reasons and methods for cheating in games, Pritchard (2001) describes six categories of online cheating, and a description of cheating techniques and the challenges of preventing cheating is provided by Yan and Choi (2001). Few of the techniques used by games for improving security have been published.

Although many of the security problems that games face are not a factor for other types of RTDG (e.g., people are not going to cheat in a shared whiteboard), the techniques used by games for preventing cheating can be used as a basis for designing techniques for other types of RTDG. Additionally, since games are latency sensitive and designed for high performance, it is likely that games have developed techniques that deliver security efficiently. However, there are no academic surveys or reliable explanations of how security is enforced in games, and work should be done to determine security techniques used in games that are applicable to other types of RTDG.

6.2 Routing

Routing techniques determine the path that information takes as it travels from its source to its destination. The users of an RTDG system form a network in itself, and so there are many possible ways of routing information through the group. RTDG routing is divided into two subcategories, distribution architectures and network routing.

6.2.1 Distribution architectures

Distribution architectures has been an area that has attracted ongoing debate from the CSCW community, as there are many arguments for each architectural style (Greenberg and Roseman, 1999). Phillips (1999) presents a survey of CSCW work on distribution architectures.

Distribution architectures determine where the pixels to be displayed on the screen are generated (display architectures), where the information to be displayed is generated and stored (view architectures), where information is processed and data is stored (model architectures), and how information is routed (network architectures). The distribution of display, view, and model components is described as being centralized (located on the server) or replicated (located on the client). Network architectures are more commonly referred to as being client-server (clients communicate via a server) or peer-to-peer (clients communicate with each other directly). Although display, view, and model layer distribution do not directly impact networking, these architectural decisions affect the type of network traffic and will need to be sent, so they are each described here briefly.

Display architectures are normally fully replicated, with the exception of screen sharing systems like VNC and Citrix, which send the actual pixels to display. Typically, in this situation, view and model architectures are also centralized and the network architecture is client-server. Therefore, when display architectures are centralized, the networking problem is reduced to one of encoding and forwarding pixels to the other clients.

View architectures are generally fully replicated in all but the earliest groupware systems. The reason is that it is generally inefficient to generate multiple views at a central location and then propagate the pixels to remote users. Therefore, RTDG systems generally generate the views at the clients. The most common exception to this is for collaboration transparent groupware systems, which convert single-user applications into groupware at runtime. One example of a collaboration transparent system that uses a centralized view layer is Netmeeting (Summers, 1998). However, it is also possible to build collaboration transparent systems using a replicated view (e.g. Begole et al, 1999). When views are centralized, the model is also usually centralized and networks usually use a client-server architecture.

Model architectures determine where data is stored in the system. A *centralized model architecture* stores all data on a central server, and then propagates data to clients for the purpose of displaying it. Updates to the data are made by sending the request to the central server, then propagating the updated information back to the client for display purposes. The advantage of this approach is that concurrency management is simple, since a central server can manage all data updates and mediate any conflicts that may occur based on the order in which information arrives. The main disadvantage is that local actions must propagate through a central server before being displayed locally, which adds latency to local actions and can make groupware applications seem sluggish compared with single user applications. The Rendezvous system (Hill et al, 1994) is an early example of a groupware system that uses a replicated view layer and centralized model and network layers.

Replicated model architectures do not have a central server that stores data, but instead keep synchronized copies of the same data on each client. The advantages are that there is no dependency on a single node, making the system more robust, and this also can result in lower latency due to eliminating the need to make updates to a centralized model layer before seeing them propagate locally. The main disadvantage is that it makes concurrency a more complicated problem to solve (Greenberg and Marwood, 1994).

Network architectures control the path that information travels through the groupware system. A *client-server network architecture* (see Figure 3A) routes all information through a central server, then propagates it to the clients in the system. The main advantage is that it minimizes the upload bandwidth required for each client in unicast networks since information only needs to be sent to the central node. Another advantage is that it offers IP address privacy since no information is sent directly to other clients in the system. The main disadvantages are that there is a single point of failure and that scalability poses a challenge since additional clients require increasingly more bandwidth at the server. Note that most systems that use a centralized model layer architecture also use a centralized network architecture, although this is not a requirement, as the machine hosting the central model layer could be treated as a peer from the network architecture perspective. Most network games use a client-server network architecture since IP address privacy is important, it makes cheating prevention easier, and it is more feasible to supply more bandwidth to a game server than to require all clients to have enough upload bandwidth to deliver unicast messages to all other participants.

Peer-to-peer network architectures (see Figure 3B) do not have a central node, but instead send information directly to each client in the network. The main advantages to this approach are that latency is reduced due to sending directly to other clients and that there is no single point of failure in the system. The main disadvantage is that scalability is more challenging when used in unicast networks, since it requires all clients to have sufficient upload bandwidth to send unicast messages to all other clients in the system (although this requirement can be eliminated using routing techniques described below). Scalability is particularly challenging due to the upload bandwidth limits imposed by most cable and Digital Subscriber Line (DSL) service providers – many home users have much less upload bandwidth than they have download bandwidth.

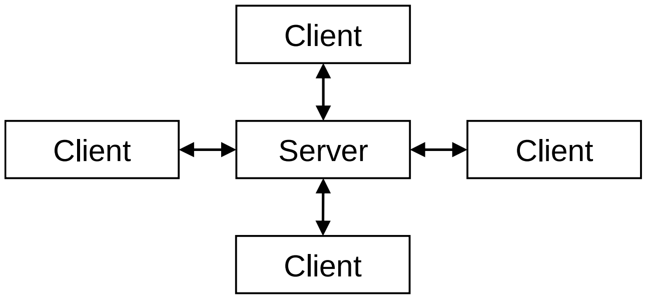

Figure 3A: A client-server architecture.

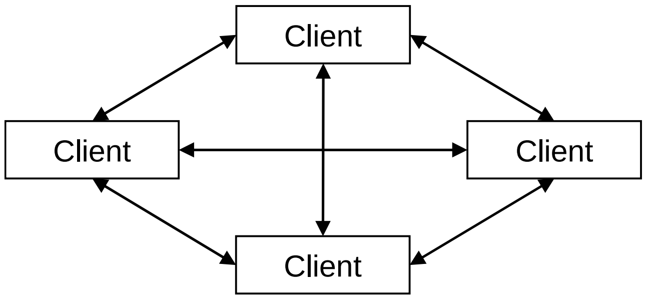

Figure 3B: A peer-to-peer architecture.

Recent high performance groupware systems have moved away from purely centralized and purely replicated architectures toward *hybrid architectures* that employ some components of both centralized and replicated architectures (Phillips, 1999). Most modern groupware toolkits use some form of a hybrid distribution architecture. The two common categories of hybrid distribution schemes are centrally-coordinated and semi-replicated.

Centrally-coordinated architectures are similar to replicated architectures, except that concurrency is controlled by a centralized concurrency manager. The concurrency manager acts as a referee when there are concurrency issues. This architecture combines the simplicity of centralized concurrency with the benefits of replicated architectures, although it is still more complex to implement than a centralized model architecture. One example of a system that uses a centrally-coordinated model layer architecture is Java Applets Made Multi-user (JAMM) (Begole et al, 1999).

Semi-replicated architectures combine both centralized and replicated model layer architectures into a single system, which couples the characteristics of centralized and replicated architectures with well-suited situations. The way that components are distributed varies among the systems that use this type of architecture. One common approach maintains all private components locally while all shared components are centralized. The advantages and disadvantages are a combination of those of centralized and replicated systems, but the costs and benefits can be tailored for subcomponents of the system. Variations of semi-replicated architectures are found in Clock (Urnes and Graham, 1999; Graham et al, 1996) and the Java Shared Data Toolkit (Burridge, 2004).

Some groupware toolkits use *flexible architectures* that allow developers to select a distribution scheme that is best for a particular application. For instance, a developer of a system that will have minimal users and is difficult to synchronize would be able to decide to select a centralized distribution scheme (O'Grady, 1996). Clock is an example of a groupware toolkit that allows application programmers to choose a distribution architecture. Clock's abstract architecture allows application programmers to choose between a centralized or semi-replicated architecture (Urnes and Graham, 1999). Programmers are able to switch between these two distribution architectures easily, so experimentation with both architectural styles is feasible (Greenberg and Roseman, 1999).

One upcoming trend may be toward *dynamic architectures*, where the architecture is selected by the systems based on network and processing performance, as well as the requirements of the application (Greenberg and Roseman, 1999; Phillips, 1999). Greenberg (1999) states, “Perhaps what is required is a dynamic and reactive groupware architecture, where the decision of what parts of the architecture should be replicated or centralized can be adjusted at run time to fit the needs of particular applications and site configurations.” This idea has been suggested by several CSCW researchers (e.g., Greenberg and Roseman, 1999; Phillips, 1999; Urnes and Graham, 1999). Some early systems, such as Enhanced XTV (Dewan and Choudhary, 1992) and Visual Obliq (Bharat and Cardelli, 1995) experimented with limited support for dynamic distribution. Enhanced XTV uses a centralized architecture that allows the server to move from client to client according to which client "has the floor" or in response to a change in the server machine state, such as a server shutdown, using a playback mechanism that plays all of the events of the groupware session back to the new server (Dewan and Choudhary, 1992). Visual Obliq uses a semi-replicated architecture that allows the client processes to be created at the server and then migrated to the clients to facilitate concurrency (Bharat and Cardelli, 1995). Litiu and Prakash (2000) describe an approach for dynamically changing between different architectural styles at runtime in the DACIA system. DACIA allows individual components called PROCs (Processing and ROuting Components) to move between different locations at runtime based on processor load and client availability using a message-based approach. Chung and Dewan (2004) present an alternate mechanism where the distribution architecture of interface and system components changes dynamically during runtime in order to better distribute processing duties among clients. Chung and Dewan's system uses a logging scheme to enable the dynamic architectural changes.

6.2.2 Network routing

Most RTDG systems communicate via *unicast* (see Figure 4), which sends a message intended for a set of recipients in separate packets addressed to each recipient. Replicated network architectures send a separate packet to each of the recipients directly from the clients, which consumes an increasing amount of upload bandwidth as the number of users in the system increases. For example, a message sent to six recipients results in six packets being sent from the client. This is particularly problematic since Internet service providers commonly provide far less upload bandwidth than download bandwidth, which creates a performance bottleneck as the number of users increases. Centralized network architectures route all messages via a central server, which results in only one message being uploaded to the server from the clients and eliminates the client upload bottleneck, but the server is then required to not only receive every message that is sent, but also to send a copy of it to each of the recipients. This causes a scalability problem for the server since it must have enough bandwidth to send each unicast message to every client. Therefore, a replicated network architecture scales better when the central server has limited bandwidth, and a centralized network architecture scales better when the server has abundant upload bandwidth. This is one of the reasons why massively multiplayer games use centralized network architectures – the game company can provide a dedicated server with sufficient bandwidth.

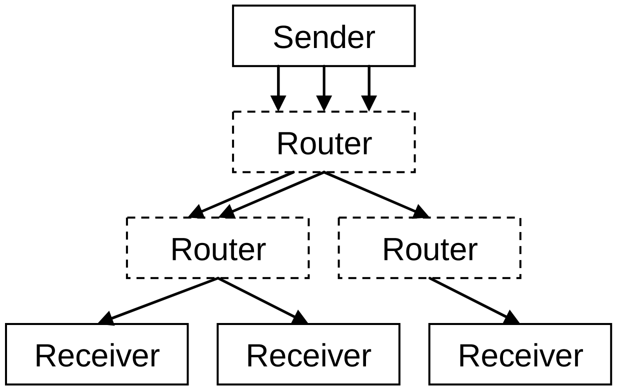

Figure 4: Unicast produces redundancy when sending identical payloads along similar routes.

When sending via unicast, packets with identical data are sent redundantly over portions of network routes. For example, sending six identical messages to six recipients is guaranteed to be redundant at least over the first network hop from the client to its ISP, and quite possibly on subsequent hops. *Multicast* is a method for eliminating some of this network redundancy. It works by sending a single packet that is addressed to a group, and copies of this packet are made at points where the routes must split to arrive at different users. This reduces overall network traffic, but comes at the cost of coordinating routes. There are two distinct types of multicast, network layer multicast and application-layer multicast.

Network layer multicast (see Figure 5) uses network routers to participate in copying and routing packets (Deering and Cheriton, 1990). This approach has been shown to provide large network performance increases in streaming video when there are large numbers of recipients. Although this is not an application-layer technique, deciding to use network layer multicast is an application-layer decision. Large portions of the Internet are not yet capable of native multicast, so it cannot be used reliably at this time (Hosseini and Georganas, 2004), but it is possible to detect whether the network supports native multicast and to use it when it is supported (e.g., Frecon et al, 1999).

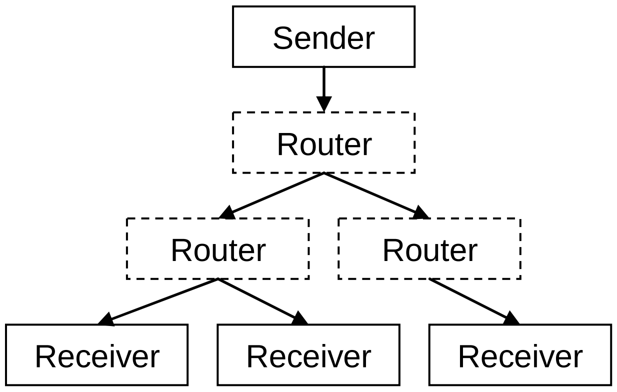

Figure 5: Network layer multicast eliminates redundancy by copying packets when routes split.

Many CSCW and CVE projects have implemented groupware systems using network layer multicast. DiveBone (Frecon and Stenius, 1998) is a hybrid unicast / multicast system that makes use of the MBone when it is available, but can unicast to clients that do not have MBone access. Massive-2 is another example of a native multicast enabled groupware system (Greenhalgh, 1996).

Matta et al (1998) observe that CSCW applications have a unique set of QoS requirements and that existing multicast path construction algorithms deliver different QoS characteristics, but that none of them can adapt to meet the variety of needs of CSCW applications. Their design allows a CSCW system to adaptively switch between several different multicast trees depending on which is best suited to delivering the QoS required. A similar design was offered by Chockler et al (1996), where four different multicast group communication system variants were used, each having different QoS qualities, and adaptively applied in order to better meet the varying QoS needs of CSCW. Each of the GCSs was used to send the messages that their inherent QoS was designed to deliver.

There are few performance evaluations that demonstrate that multicast is useful for RTDG. Given that RTDG systems combine unreliable and reliable data streams, and often have a small number of users, there are undoubtedly situations where unicast is better than multicast and vice versa. A basic model comparing bandwidth usage between multicast and unicast implementations is presented by Frecon and Stenius (1999), which shows that multicast uses far less bandwidth than unicast as the number of users increases, although this model does not account for the effects of network loss, variable QoS requirements of RTDG, or for the overhead required for multicast. Another RTDG specific performance study (Kanxin et al, 2000) shows that multicast is superior to unicast when both loss rates are low and the number of users is low. Unicast performed better in situations where both loss rates were high and there were a large number of users. More work is required to learn when multicast is suitable for use in RTDG and when it is not. Additionally, there are several multicast service delivery models available, and it is not clear which are best-suited for use with RTDG.

One of the problems with network-layer multicast is that large parts of the Internet are still incapable of delivering native multicast. *Application-layer multicast* (see Figure 6) does not rely on underlying networks, but rather builds multicast routes using the end points (Banerjee et al, 2002; Chu et al, 2000; Francis and Yoid, 1999; Chawathe and Scattercast, 2000; Janotti et al, 2000; Pendarakis et al, 2001; Zhuang et al, 2001; Ratnasamy et al, 2001). In a RTDG system, a message would be forwarded to one user, which forwards a copy of it to the next, and so on until the recipient list is fulfilled. This can eliminate the bandwidth bottlenecks that occur when sending via unicast to all users, but it can add latency due to the need to forward messages via intermediate points, as well as consume resources in building the multicast routes. One example of an application-layer multicast implementation in RTDG is described by Hosseini and Georganas (2004), where application-layer multicast was used in a 3D videoconferencing application designed for use by 4 to 10 users. They show that application-layer multicast has potential for use in this situation, but more work is required before the approach can be used. Further work is also required to determine more precisely when and how to use application-layer multicast for RTDG in general.

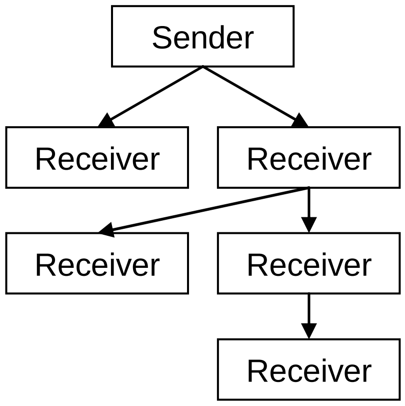

Figure 6: Application-layer multicast uses end points to route information to a group.

6.3 Reliability

Reliability techniques are used to counteract ordering and loss problems that occur while sending information over the network. The reliability techniques for RTDG are divided into four subcategories: protocols, sender-based error correction, concealing and revealing errors, and concurrency / consistency management techniques.

6.3.1 Network Protocols

RTDG typically makes use of existing network protocols designed for other purposes, but has also experimented with developing custom protocols that better suit the needs of RTDG. This section describes the protocols that have been applied to RTDG and describes the results of doing so, and describes work done to develop protocols that are specifically developed for the needs of RTDG.

Most academic groupware systems use *Transmission Control Protocol* (TCP) for sending groupware information (e.g., Greenberg and Roseman, 1999; Begole et al, 1999; Marsic, 2000). This is a simple approach that takes very little effort to implement since it guarantees in-order delivery and controls transmission rate, and therefore does not require application programmers to deal with lost information, out-of-order messages, corrupted data, or rate control issues. The main problem with TCP is that it is not well-suited for sending real-time information due to its reliability mechanism and the way it controls rate. In particular, guaranteeing in-order delivery requires considerable resources to acknowledge and retransmit lost information; second, guaranteed in-order delivery is not required for many types of information that are transmitted in real-time groupware. As a result, some RTDG applications that are implemented using TCP experience performance problems, particularly when used over networks with high loss rates and low bandwidth (Dyck et al, 2004; Dyck and Gutwin, 2006).

The other network protocol used for RTDG development is *User Datagram Protocol* (UDP). Since UDP does not guarantee delivery, order, or integrity, it is typically used in conjunction with TCP for sending unreliable data in the system or with other higher level custom protocols that increase reliability (a.k.a. reliable UDP (e.g. GarageGames, 2005)).

Almost all game networking is UDP based, with *customized protocols* that are implemented on top of UDP to deliver various reliability and ordering requirements. Most game networking libraries support several discrete reliability and ordering QoS levels (Dyck and Gutwin, 2006). For example, the Torque Network Library, which was derived from the games Starsiege: Tribes and Tribes II, allows messages to be sent with five QoS levels: 1. guaranteed ordered; 2. guaranteed, which ensures information will be delivered but does not check ordering; 3. unguaranteed, which does not offer any additional reliability to UDP; 4. current state data, which discards information queued for sending if newer information is available; and 5. quickest delivery data, which includes the piece of information in every outgoing packet until it is acknowledged as received (GarageGames, 2005).

Real-time Transport Protocol (RTP) is an application-layer packet protocol that is used primarily for sending real-time voice and video transmissions, and some have applied RTP to RTDG applications (e.g., Cheok and Li, 1996; Geyer and Effelsberg, 1998; Frecon and Stenius, 1999). There are RTP payload formats for common RTDG message types, including text messages (Hellstrom, 2000), telepointers (Civanlar and Cash, 2000), and for shared multicast virtual worlds (Robinson and Stewart, 1999). There are no studies that demonstrate how well RTP performs when used for RTDG applications. For example, Cheok and Li (1996) wrote a shared whiteboard which used RTP, and although the implementation using RTP is carefully described, this system was never performance tested against other protocols. However, examining the features of RTP has shown that it is not well suited to delivering the wide range of QoS that RTDG requires, and that the RTP header includes information that is not required for groupware traffic that is not voice or video (Mauve et al, 1999; Mauve et al, 2001; Perkins and Crowcroft, 2000).

RTP Interactive (RTP/I) is a protocol that is based on the lessons learned from the RTP protocol, and targets the specific needs of RTDG (Mauve et al, 1999; Mauve et al, 2001). Some of the key features it adds include application-specific naming of messages, synchronization with RTP media streams, flexible reliability mechanisms, and the ability to record interactive media applications. Perkins and Crowcroft (2000) criticize the original approach of RTP/I, which was to extend RTP, arguing that it preserves some of the poorly-suited features of RTP, resulting in a large header size and inappropriate timestamps, sequence numbers, receiver report formats, and unnecessary sender reports. They suggest that RTP should serve as a useful design to consider when developing a protocol used for RTDG, but that it should not be extended. They also suggest that Reliable Multicast Framing Protocol (RMFP) (Crowcroft et al, 1998) or Reliable Multicast Framework (RMF) (DeCleene et al, 1997) may be better-suited for the needs of RTDG. RTP/I was later modified according to these recommendations (Mauve et al, 2001) and submitted as an IETF draft (Mauve et al, 2000), but has not been modified since then, and has not been adopted by RTDG application developers.

One approach to meeting the diverse QoS requirements of RTDG is to employ multiple protocols that support a variety of QoS characteristics (e.g., Barrett et al, 1998, Birman et al, 1997, Chockler et al, 1996). However, doing so can cause the protocols to compete for resources, which can lead to poor performance. To avoid this problem, multiple protocols can be combined into a multi-channel protocol stack that shares some common layers with one another (Miranda et al, 1999), as shown in Figure 7.

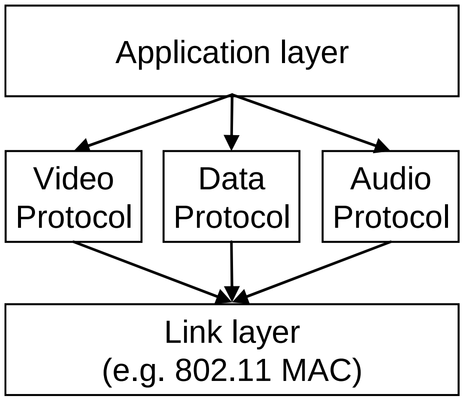

Figure 7: A multi-channel protocol stack combines common layers from different protocols, reducing redundancy and competition (adapted from Miranda et al, 1999).

There are several infrequently used protocols that capture some of the needs of RTDG but have not been applied to RTDG, such as XTP (Atwood and Zhang, 1995) and SCTP (Caro et al, 2001). *eXtendible Transport Protocol* (XTP) can send information unreliably, reliably, or partially reliably and optionally via multicast, which is appropriate for RTDG since it would move some of the techniques used to deliver QoS away from the application without having to use multiple protocols and it would simplify QoS management. *Stream Control Transmission Protocol* (SCTP) is a unicast protocol that delivers TCP-like reliability and congestion control, but adds several features that are useful to RTDG. SCTP includes a lifetime for data to be transmitted, which drops data that is no longer useful from the transmission queue. It supports multi-streaming, which partitions data into several independent streams and improves performance by allowing controlled and limited out of order delivery. It also supports multi-homing, allowing a single client to have multiple IP addresses, promoting survivability of the session in the event of network outages. The main barrier to applying these protocols to RTDG is that there are few implementations of them and they are not widely used or known. XTP, SCTP, and other protocols should be investigated to better determine their applicability to RTDG.

There is a trend in network games to steer away from protocols that deliver reliability or ordering since this is most effectively done at the message level instead of at the packet level (Dyck and Gutwin, 2006). By employing a packet-level protocol, QoS can no longer be controlled on a per-message basis since many messages with different QoS requirements may be aggregated into a single packet by the application. For example, if a packet is lost and it contained 5 messages, 1 of which required retransmission, the entire packet would be sent again using a packet-level protocol, while only one of the messages actually needed to be retransmitted. As another example, consider two types of groupware messages that required in-order delivery, but that are independent of one another. Using packet-level order guarantees, receiving one of the streams out of order would require the other stream to stall until packets were reordered, while if done at the message level, a late arrival in one stream would not affect the other. A message-level approach is used in game network libraries such as OpenTNL

(GarageGames, 2005), Raknet (Rakkarsoft, 2004), and Enet (Enet, 2003). Therefore, a very lightweight packet protocol that can take advantage of end-to-end features like native multicast and generalizes some of the information in the message protocol to save space would be best suited to the needs of RTDG. Existing work on protocols should be used to guide the design of a message-level protocol for RTDG. A lightweight packet protocol and a separate message protocol would be better-suited than packet-level protocol that does all.

6.3.2 Sender-based error correction

Sender-based error correction techniques improve reliability problems that are primarily due to loss and ordering errors. These techniques retransmit information by repeating it in subsequent packets or along multiple network routes. These techniques are critical to RTDG performance since reliability generally comes at the expense of network performance and added latency, and making use of appropriate error correction techniques can deliver the reliability that is required while using a minimum amount of network resources. There are many types of error correction techniques that can be implemented at the application layer and that are well-suited for use in RTDG.

Retransmission-based techniques resend data that is known to be lost. There are two main ways to do this – by acknowledging all information when it arrives (ACK-based) or by requesting information when loss is detected (NACK-based) (Perkins et al 1998). *ACK-based* error correction techniques are the most commonly used error correction methods seen in RTDG, but it is resource intensive since it requires all received packets to be acknowledged. *NACK-based* techniques detect loss at the receiver using sequence numbers and request a retransmission in the event of detected loss. A requirement of NACK-based protocols is to have a continuous flow of information since loss can only be detected after subsequent packets have arrived. One workaround for this requirement is to send reliable termination packets, which indicate the end of a transmission. Therefore, NACK-based protocols may be better-suited when traffic is in the form of continuous streams or bursts of information, such as with telepointers, while ACK-based protocols may be better suited when traffic comes in very short bursts or as sporadic individual messages, such as in an instant messaging system.

Forward error correction (FEC) improves reliability by including information that was sent in previous packets again in subsequent packets (Bolot et al, 1999). In the event that a packet is lost, the information can be recovered when subsequent packets arrive without retransmission (see Figure 8). This technique can be used to recover from burst losses where a number of packets in a row are lost, as long as the number of times the information is repeated exceeds the burst length. However, when burst length exceeds the number of times information is repeated in subsequent packets, losses occur, so this technique is only partially reliable. This partial reliability comes at the expense of data rate – additional data is sent redundantly, increasing the bandwidth requirements when compared to unreliable protocols. In RTDG, packet sizes are typically quite small, and individual messages can be tiny, so repeating messages in subsequent packets comes at a small cost compared with VoIP or streaming multimedia, where this technique is frequently used. FEC can be implemented to work with any kind of data (media independent FEC), and efficiencies can be gained by developing encoding techniques that take advantage of the characteristics of the data being sent (media specific FEC). *Adaptive FEC* (AFEC) (Bolot et al, 1999) adjusts the amount of repeated information dynamically in order to meet the QoS requirements of the information being sent while minimizing the network cost of doing so.

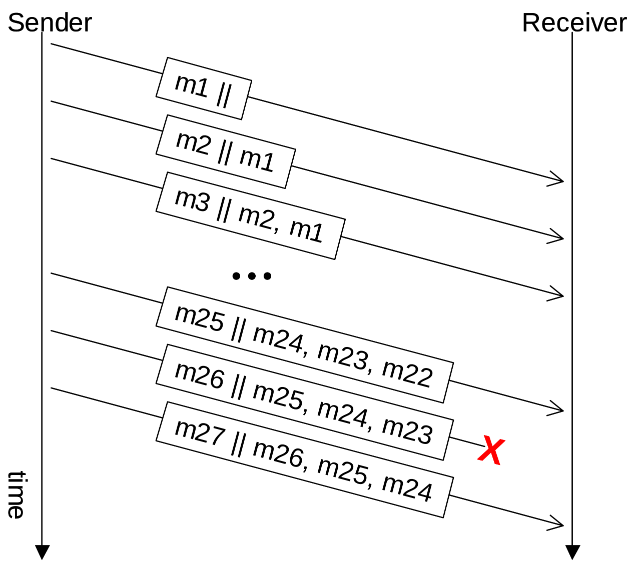

Figure 8: This FEC example includes three previous messages in subsequent packets, allowing lost information to be recovered without retransmission.

Interleaving (see Figure 9) disperses contiguous information among several packets by reordering portions of the messages, and then puts it back in order at the receiver (Perkins et al, 1998). For example, a telepointer stream that sends 20 messages per second at a packet rate of 5 packets per second (4 messages per packet) could be interleaved by sending messages 1, 5, 9, and 13 in packet 1, messages 2, 6, 10, and 14 in packet 2, and so on. In the event of a single packet loss, less contiguous information would be lost, which would result in smoother telepointer motion at the receiver. However, this comes at the cost of added latency, since the information cannot be reordered and played out at the receiver until all interleaved information is received. Therefore, this technique is only suitable when this type of reliability gain is useful and the added latency does not outweigh the gains. It is also useful for large messages, where FEC is a less feasible approach. Large messages that tolerate discontiguous loss better than contiguous loss, and that are not particularly latency tolerant, are not common in RTDG applications, and so interleaving probably has limited use in RTDG.

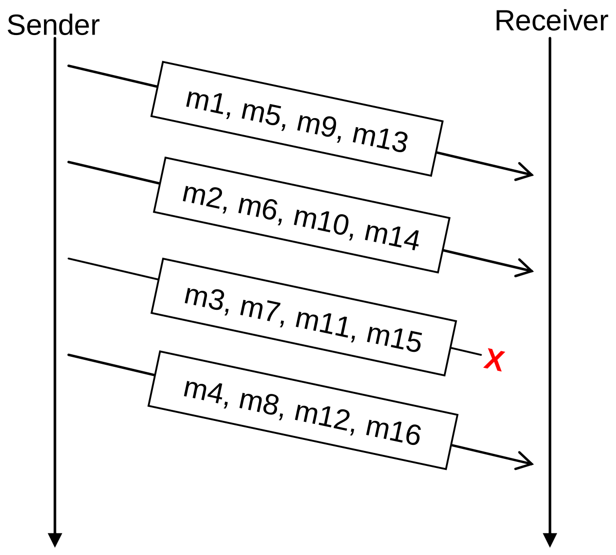

Figure 9: This interleaving example for groupware shows how messages can be dispersed through packets, resulting in less contiguous losses of messages from a single packet loss, but higher latency.

Packet path diversity sends information redundantly over two separate network routes in order to improve reliability (Liang et al, 2001). This works because different network routes have uncorrelated loss patterns (Savage et al, 1999). This technique has been shown to improve reliability more effectively than AFEC at the same data rates, and has been applied to several areas related to RTDG, such as VoIP (Liang et al, 2001) and real-time streaming video (Liang et al, 2002). The problem with this technique is that today’s networks generally do not support the ability to define a network route and specialty networks are required to implement it. However, it may be possible to do this at the application layer by using the participants in the group to build partially different network routes. Since network routes would not be totally independent, this technique will be less effective when used at the application layer, but it still may prove to be a useful way to improve performance.

6.3.3 Consistency and concurrency techniques

The CSCW community has made many contributions to improving performance of RTDG using *concurrency and consistency control* techniques. Although concurrency and consistency techniques are not application-layer networking techniques, they are application-layer techniques that do have an effect on network performance and can sometimes compensate for network problems, so they are relevant to this survey. Like distribution architectures, this area has also attracted attention within the CSCW community. Concurrency control in RTDG is a difficult problem since RTDG users rely partly on social protocols for managing concurrency – people become aware of conflicts and resolve them through negotiation with one another (Greenberg and Marwood, 1994). When we implement additional concurrency techniques, it must be done carefully so as to assist the way that users work and not interfere with the way in which social protocols function.

A large number of consistency maintenance mechanisms have been developed for use in RTDG. A taxonomy of concurrency techniques is offered in (Greenberg and Marwood, 1994) and extended in (Yang and Li, 2004), which organizes techniques into three top level categories: locking, serialization, and operational transformation.

Locking techniques award a lock to one user and prevent other users from affecting the locked data until the lock is released. Locking techniques are subdivided into pessimistic, semi-optimistic, and optimistic policies, which differ in the level of optimism in the way that the locks are requested and released (Yang and Li, 2004). *Pessimistic locking* policies block user actions after a lock is requested until it is granted, and also block actions after a lock is released. Therefore, this creates wait times while locks are being granted and released, and applications may appear unresponsive. The advantage of pessimistic locking policies is that consistency is guaranteed and there will be no rollbacks in the system. *Semi-optimistic locking* policies only block actions when a lock is released. This allows the user to continue performing actions while locks are requested and not yet granted, which increases the responsiveness of applications. However, this presents a risk of a user performing updates which will then not propagate due to the lock being denied, and so there is a chance of a rollback for the user performing the updates, but not for other users in the system since further updates are not possible until the lock is released. *Optimistic locking* policies do not block user actions, so this policy never requires the user to wait for a lock to be granted or released, and provides the highest amount of responsiveness. However, conflicts can occur as a result and rollbacks are necessary to correct conflicts.

Serialization maintains consistency by ordering all operations on the same piece of data (Yang and Li, 2004). This is also subdivided into two different categories, optimistic and pessimistic. *Pessimistic serialization* guarantees that operations are in order before executing them, while *optimistic serialization* will execute operations as they arrive, but will undo and redo them if they are found to be out of order.

Operational transformation (Sun and Ellis, 1998) is an optimistic policy that does not undo previous operations, but rather merges late arriving operations with ones that were executed out of order. This avoids the costs of undo, but requires the programmer to specify how each operation should be merged.

The best concurrency technique depends on the task and situation at hand. For example, operational transformations are well-suited for group editing tasks since they provide fast response times and concurrency, although they are not suitable when groups are large and doing loosely-coordinated work (Edwards, 1996). Therefore, *flexible* (Graham et al, 1996) and *adaptable* (Yang and Li, 2004) concurrency schemes have been offered, which allow the programmer to select an appropriate concurrency policy.

6.4 Scheduling

Scheduling techniques decide when information is sent and how it is packaged. Scheduling is broken into four subcategories: rate control, aggregation, priority scheduling, and filtering.

6.4.1 Rate control

Many groupware systems are built using *event-driven sending policies*, sending messages whenever events happen at the user interface. Although this is fine for low traffic systems, such as a chat application, this can result in poor performance when events are triggered at a high rate. For example, GroupKit sends telepointer messages at each mouse interrupt, which depends on the speed of the system it is running on, but is usually a much higher rate than is necessary for delivering high quality telepointers (Dyck et al, 2003). To improve performance, network transmission of messages should be decoupled from the system's event model and transmission rates should be carefully controlled (Dyck et al, 2004).

Static rate control uses a fixed send rate to send messages. It is usually set at a rate that works for even poor connections. For example, the game Quake sends packets at a fixed rate of 10 packets / second. Fixing the send rate at a low value comes at the expense of smoothness (or accuracy if using prediction to compensate) and adds a small amount of latency. The smoothness problem can be improved using concealment techniques (described below), but this comes at the expense of accuracy. A higher fixed send rate would improve smoothness, but would also consume more bandwidth, limiting its use. Some games allow the user to adjust the send rate manually. For example, the multiplayer strategy game Starcraft allowed the user to select between three different settings, which adjusted the transmission rate as well as the amount of gain on concealment techniques. However, it is doubtful that the user can effectively manage rate control, and this decision is likely better made by the system.

Adaptive rate control adjusts the send rate of information to best suit the QoS requirements of the information and the current network conditions. When bandwidth is plentiful, it sends at a higher rate, thus delivering better smoothness and accuracy, and when bandwidth is sparse, it slows its send rate at a cost of smoothness and / or accuracy. Dyck et al (2004) presents one example of a RTDG system that does this, where adaptive rate control was applied to improve the performance of telepointers. This technique is used in game networking libraries to manage the amount of bandwidth used (Dyck and Gutwin, 2006).

One of the problems faced by rate control algorithms is that they only work if they all play by the same rules (Floyd and Kevin, 1997). For example, TCP rate control reduces the send rate when losses start occurring, in order to decrease congestion. However, other rate control algorithms, like the static approach used in Quake, do not reduce their send rates similarly, and so TCP traffic slows while other traffic continues to push through at higher rates, creating further congestion and making TCP perform poorly at the expense of other traffic. For this reason, *TCP-friendly rate control* has been proposed as a standard that all network traffic should observe (Handley et al, 2001). Several TCP-friendly algorithms for rate control have been proposed for use with voice and video (e.g., Sisalem and Schulzrinne, 1998; Vicisano et al, 1998; Turletti et al, 1997), and these can be used to guide the design of TCP-friendly rate control for RTDG applications.

6.4.2 Aggregation

Aggregation is a bandwidth-reducing technique that combines multiple pieces of information into a single packet. It works by adding outgoing information to a queue, using an aggregator to build packets from the messages in the queue. This technique is used in most open source game networking libraries (Dyck and Gutwin, 2006), and it has been applied in CVEs such as Artery (Chiueh et al, 1997) and MASSIVE-2 (Greenhalgh et al, 1999). When sending frequent, small messages, this technique is particularly effective, as it greatly reduces the number of packets at the expense of a small amount of added latency. Aggregation can be used to limit packet size, and can be used in combination with rate control to effectively limit overall data rate effectively. Aggregation can also be used in combination with priority scheduling to better meet QoS requirements of information being sent, as well as to help applications degrade gracefully under low bandwidth conditions. More work is required to develop effective, QoS-aware aggregation policies for RTDG.

6.4.3 Priority Scheduling

Priority scheduling for network packets is a mature area, but little work has been done to develop scheduling algorithms for RTDG traffic, even though RTDG traffic has widely variable QoS requirements and could benefit from priority scheduling. Priority scheduling is used in some game networking libraries in order to help them to degrade gracefully when bandwidth is low, and to reduce latency for latency-sensitive information (Dyck and Gutwin, 2006). More work is required in this area to develop appropriate priority scheduling algorithms for RTDG traffic, and in particular, scheduling algorithms for aggregation policies should be considered.

6.4.4 Filtering

Traffic filtering reduces the amount of traffic by only sending information to other users who are interested in the updates. This is done using a degree of interest (DoI) function to determine which users should receive a given piece of information. The most common DoI parameter is whether the user is in view or not, as sending updates to users who cannot see the updates is often useless and wastes resources. DoI filtering is commonly used in CVEs, (e.g., RING (Funkhouser, 1996), MASSIVE (Greenhalgh, 1996), Community Place (Lea et al, 1997)), and games (e.g., Artery (Chiueh et al, 1997)).

6.5 Providing information to higher layers

Applications can compensate for network problems in a variety of ways beyond what application-layer networking can do. As a result, some of the information gathered about the quality of the network must be made accessible to higher level network layers. This section describes two classes of techniques that are used by higher application layers that make use of network data: adaptive UI techniques and concealing and revealing errors.

6.5.1 Adaptive UI techniques

Adaptive user interface components that change their interaction techniques to accommodate network conditions can deliver a lower level of service to the user under poor network conditions. This approach was introduced by Whalen and Black (1998). For example, the system would use telepointers when there were sufficient network resources to support them, but would automatically switch to supporting a much lower network cost static pointer device that the user could use by clicking and dragging. This strategy removed a stream of media, while continuing to support some of the useful features of telepointers, such as simple gestures and the ability to show someone where you are working in the workspace explicitly.

Although UI adaptation is not a network layer technique, it is an approach that adapts to changes in the network and helps users and applications to cope with limited network resources. No further related work has been published in this area, and it seems worthy of further exploration, as we can imagine many interface component degradation and swapping techniques that can deliver different service levels under a variety of network conditions.

6.5.2 Concealing and revealing errors and delays

One way to improve the usability of RTDG applications is to use techniques to conceal losses or late arrivals by interpolating and predicting, or to reveal network problems to users in a useful way so that users can adapt to work with them more effectively (Gutwin et al, 2004). *Concealment* techniques fall into several categories, including insertion, interpolation, regeneration, extrapolation, and prediction. Concealment approaches are used in many network games and some CVEs, but are seldom used in CSCW applications (one example appears in Dyck et al, 2004). *Revealing* network problems to users is done in games mainly by providing a small visualization of ping times, latency, and jitter; in CVEs and CSCW applications, interface decorators that reveal delay to users have been developed (Gutwin et al, 2004).

7. NETWORK MONITORING

RTDG systems are used by people with widely varying amounts of network resources, where the applications often have bursty traffic patterns, and where the number of users can change drastically. This implies that RTDG application must adapt their network performance dynamically. Part of being able to do this effectively is monitoring network resources at run time. Additionally, information needs to be known about the connections between each of the clients in the groupware session in order to build application-layer multicast routes. This section describes two approaches: application layer network resource monitoring, and cross layer signaling.

7.1 Monitoring resources

Remos (DeWitt et al, 1997) is an early *network monitoring* system that was dedicated to tracking network conditions. Most earlier distributed systems monitoring work was aimed at monitoring processor loads for cluster computing. The Remos paper identifies many of the challenges involved in monitoring network resources. The main aspects that make it hard to monitor resources are that the networks differ greatly in the amount, accuracy, and depth of information that can be queried directly, as well as where and how the information is stored. Another challenge is that network resource sharing is not well-considered in operating system designs, and to provide information to an application about the resources available to it is not feasible since it is unclear what the requirements of other applications will be. The most reliable method for determining resource availability is to generate traffic to test the resources, but this consumes resources and is less useful if frequent status updates are required.

Monitoring issues for RTDG have not been considered directly, but the general resource monitoring problem has been studied. Although RTDG undoubtedly has its own monitoring requirements, we can use ideas from other monitoring work as a basis for designing an approach for RTDG. The best approach to monitoring the network in groupware may be to monitor the application traffic directly and periodically report statistics for loss, jitter, latency, and bandwidth to other nodes, which is similar to the approach used in RTCP. This is the only method that works on all networks, and it is simpler than trying to query the network directly for this information. It may also be useful to query a network monitoring tool like Remos for some information, although it is unclear if the necessary information for RTDG can be obtained more effectively from application traffic directly.

7.2 Cross layer signaling

The high level information that the application layer receives about the state of the network limits the ways in which we can cope at the application layer. For example, when the application layer observes a loss occurring, it does not receive information about why the loss occurred. This information is filtered out at the MAC layer, and does not propagate up to the application layer. The MAC layer may be a wireless card and may know that the loss occurred because of interference, and this information about why the loss occurred could be useful to the application layer for making adaptive decisions. The OSI network model does not allow this information to be accessed at the application layer. *Cross layer signaling* (see Figure 10) argues that although sharing information between network layers adds complexity and violates the OSI model, it can provide performance benefits to application-layer techniques that can use the information for making better decisions about how to compensate for network problems.

This is demonstrated in a study by Haratcherev et al (2005) that used cross layer signaling to considerably improve visual quality of a wireless mobile real-time video stream. In this study, the authors noted that the link adaptation technique in 802.11 wireless used at the MAC layer is better suited to downloading files than it is to streaming video, and that they could compensate for this using an adaptive video encoder and rate controller if it had information about the signal to noise ratio, packet loss statistics, and rate control information from the MAC layer. They used this MAC layer information to couple their video encoder's adaptive rate control technique with the MAC layer's rate control, and they adjusted their video encoder to match the generated video data rate to the MAC layer's rate.

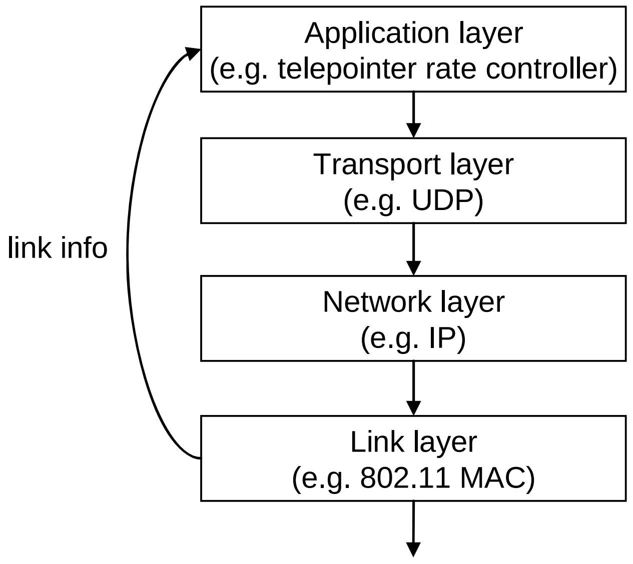

Figure 10: Cross layer signaling uses information from lower network layers to help make better adaptive decisions at higher network layers.

This approach has not been used for RTDG yet, but it is likely that it could similarly lead to performance improvements by improving application-layer adaptation.

8. TESTING

Testing RTDG networking techniques or implementations is a difficult task, as the traffic generated depends on the applications used, the tasks being performed, and on the interactions of the users. One way to test these applications is by running real user tests using simulated or real networks. This approach is used extensively by games – developers test performance early during development by using simulated networks, and large scale beta tests are done with real users prior to releasing a game. This is very costly and time consuming, and is not a feasible approach for small budget projects, especially those without a motivated, free workforce of beta testers like games have.

There are two main low-cost approaches to testing RTDG – characterizing an application then simulating its characteristics, and using simulators to generate semi-realistic traffic.

The first approach to *simulating* an RTDG application is to *characterize* the network traffic generated by the application, model the traffic statistically, then turn that model into a simulation. An excellent example of this approach is seen in a study of two games, Starcraft and Counter-strike (Claypool et al, 2003). This study gathered traffic data for both games with different numbers of players, and presented many findings about traffic characteristics. The traffic was then modeled statistically, and the models for each game were then built as modules for the network simulator NS2. This paper presents an effective approach to studying and simulating network traffic for a groupware system, and should be considered for future RTDG characterization and simulation studies. However, this approach can only be used for existing systems. A similar study of game traffic for two games, Quake World and Unreal Tournament, used a similar approach (Joyce, 2000).

Another approach is to use a *simulator* to generate hypothetical groupware traffic. One example of a CSCW traffic simulator is Bargh et al's CoCoWare (2000), which simulates groupware application traffic using group size, sender group size, and message pattern as the main parameters. The message pattern was modeled as bursts of messages with intervals in between the bursts and use burst length, message size, in-burst interval, and between-burst interval as parameters to model this. The Zereal simulator (Engum et al, 2002) is designed to simulate massively multiplayer game traffic, and works by modeling player actions rather than by simulating traffic characteristics. The traffic generated is the result of the simulated player actions. This is a much more complex and less reliable approach than the characterizing and modeling approach described above, but one that is useful when modeling systems or situations that are not yet implemented.

9. DISCUSSION

RTDG faces a more difficult application-layer networking challenge than any other class of distributed software. This is because of RTDG's diversity – there are many different types of applications, each application has many different interaction techniques with different QoS needs, the number of users can vary greatly (from 2 to 100,000+), traffic depends on user interaction, and the networks that RTDG is used over are heterogeneous, from low bandwidth, lossy, wide-area wireless networks to high-speed, reliable LANs. Most RTDG applications implement very few of the techniques presented in this survey, and as a result, network performance is generally poor. This is not the fault of the developers, but rather a result of the immaturity of application-layer networking for RTDG.

This survey presents a number of techniques that can be applied to improve application-layer networking performance. However, the techniques alone do not solve the performance problem. Rather, an approach that integrates these techniques and makes effective situational use of them is needed, and developing such an approach is a complex task. There are several other prerequisites that need to be fulfilled in order to put these techniques into effective use in groupware. This section presents some of the issues that must be addressed in order to make effective use of this set of techniques together.

9.1 Toward a RTDG network standard

The RTDG community has treated the application-layer networking problem as a separate problem for each application or toolkit that is developed. Application developers are implementing customized application-layer networking techniques into applications and high level toolkits. This is in spite of the fact that RTDG applications share common application-layer networking needs. This practice of re-implementing subsets of application-layer techniques needs to evolve into an integrated approach that can be incrementally improved.

Developing a generic approach to RTDG networking is a very complex task, and there have been few attempts to do so (e.g., Mauve, 1999), none of which have been adopted in any significant way. RTDG is at the very beginning of the process seen in other networking domains – techniques evolve into standards, standards into commonly used protocols and techniques, and protocols and techniques are then implemented as libraries that are used by many applications, and sometimes later into operating systems directly, and these standards can evolve into end-to-end features supported in network hardware and software. These standards evolve through controlled processes within a community of experts and new techniques are adopted and implemented into updated libraries. Applications use these libraries through standard interfaces, and updates to the networking techniques can happen independently of applications that use them. This approach abstracts the complexity away from application developers while giving them the high performance benefits of expertly developed networking techniques, and it also allows independent evolution of the techniques used. Clearly, application-layer networking for RTDG needs to follow this proven approach to networking that has worked in other domains.

However, due to the complexity of the RTDG networking problem, this evolution to an integrated approach has several prerequisites, and these prerequisites have not yet been met. For instance, a standard interface between RTDG applications and networking libraries requires careful consideration of the QoS requirements of RTDG, and so more QoS work must be done for the purpose of building this interface. As another example, RTDG network libraries must be able to deliver many different QoS levels, and a network architecture for managing this complexity needs to be developed. The most pressing work in RTDG networking is then to meet these prerequisites to enable integration.

This direction is beginning to evolve in network games, which has the most experience of any of the research communities of RTDG with delivering high performance over the Internet. Games have realized that good application-layer network code is hard to develop, and several stand-alone libraries have evolved to meet the needs of games. Some examples are Microsoft's DirectPlay (which is now defunct and being replaced by Xbox Live!), the Torque Network Library (GarageGames, 2005), and Raknet (Rakkarsoft, 2004). However, games that use these libraries have to be integrated using non-standardized interfaces, the libraries use different sets of techniques for delivering performance, and the designs and techniques used in these libraries are largely unpublished, so they are far from standardization. The approach used in these game libraries should be examined and considered when designing a generic approach for RTDG.

An additional reason to standardize RTDG networking is to facilitate integration among applications. terHofte (1998) argues that group work requires many tools, and since a group may want to use multiple tools, they should integrate with one another. Group formation tools (e.g., game lobbies and yellow pages servers) also rely on the ability to integrate information from various RTDG applications, and standardization would facilitate the development of these types of tools.

9.2 Tailored application-layer networking techniques

Many opportunities remain for developing application-layer networking techniques that are tailored for the needs of RTDG. Many of the techniques described in this survey are from related classes of software, and have yet to be applied to RTDG. Applying these techniques is not simply a matter of plugging them into RTDG applications. Rather, they need to be tailored for the needs of groupware to take advantage of the features of RTDG traffic. RTDG traffic has several characteristics that make it distinct from traffic in other classes of real-time distributed software. In particular:

- Information has diverse QoS requirements
- Individual pieces of information are usually small
- Traffic is bursty due to dependency on user interaction
- Information can be filtered by interest

It is these characteristics that will lead to novel versions of existing techniques that perform well for RTDG. These characteristics of RTDG traffic can also be used to develop new networking techniques that have not been seen in other classes of applications. For example, these characteristics led to several high performance techniques in open source game libraries (Dyck and Gutwin, 2006). As we learn more about the nature of RTDG traffic, we will have more possibilities for new methods to take advantage of them for the purpose of improving performance.

9.3 A QoS architecture for RTDG

When compared to related areas such as VoIP, real-time streaming video, and distributed systems, RTDG has taken a very simplistic approach to QoS. This is partly because defining and delivering QoS in RTDG is a more complex problem than it is in these related areas, and its complexity has been avoided. In particular, RTDG needs a QoS model that considers all of the QoS characteristics that are relevant to RTDG, and a set of mechanisms that are well-suited to implement application-layer QoS in RTDG applications. RTDG also needs more studies to determine the QoS requirements needed to support various collaborative activities by considering the quality of experience of the user and mapping it to QoS parameters. More advanced treatment of QoS in RTDG is an essential step that will lead to development of new high performance techniques and will enable abstractions between applications and networking libraries.

9.4 Degrading gracefully

One of the most severe causes of performance problems in RTDG is that the applications do not degrade gracefully under constrained network conditions. Instead, RTDG applications tend to maintain an unsustainable data rate, resulting in large amounts of lag or a complete halt to distributed interaction. To solve this problem, RTDG applications need to monitor the network resources that are available and make the most effective use of them, sacrificing quality where it is least important and diverting resources to the most critical functions. Some of the techniques described here can be applied to help RTDG applications degrade gracefully under constrained network conditions, but more work is required in this area, as the inability of RTDG to degrade remains an important performance issue. In particular, guidelines about what to degrade and methods for how to degrade are necessary.

9.5 Situational and combined use of techniques

This survey includes a large number of techniques, but knowing when to use each of them and how to use them together remain unsolved problems. Many of these techniques are well-suited for particular situations, and poorly suited for others. For example, FEC is a good technique for when pieces of information are small, frequent, latency sensitive, and loss tolerant, and when bandwidth is not constrained. Therefore, it is a good technique for sending telepointer positions, but not a good technique for sending chat messages. What is needed is a set of canonical RTDG situations and a set of techniques that is particularly useful for delivering each of them. Dyck et al (2004) present an example of this kind of work for telepointers, but more general examples are required that can guide the design of a variety of interaction techniques that can be used in different kinds of RTDG applications.

A second challenge is that many of these techniques affect each other and they cannot simply be pieced together. For example, interest-based traffic filtering reduces the recipient list for information, making it more cumbersome to build multicast routes that suit each of the recipient lists. As another more common example, FEC increases bandwidth usage in order to improve reliability, but the increased bandwidth also affects the rate at which information can be sent, so FEC and rate control create a tradeoff between reliability and latency or frequency. Therefore, designs that integrate multiple methods need to be carefully considered and described, we need sets of techniques that are effective in combination, and these techniques need to be coupled with situations in which they are useful.

9.6 Summary: Top issues in application layer networking for RTDG

1. Work toward RTDG network standards. This will enable integration among applications, abstraction for application programmers, and faster and more productive progress toward better performance.

2. Study and model QoS. The complexity needs to be better understood and modeled. Doing so will set goals for networking development, define constraints for application developers, and provide an effective means for communicating and negotiating the needs for RTDG information.

3. Study games. Games have the significant real-world experience with RTDG networking, and many of the techniques they use are not well-known or published by academic communities.

4. Develop more techniques. Very few application layer networking techniques have been tailored and tested for use with RTDG. More work is required to tailor existing techniques and develop new techniques that make use of the characteristics of RTDG traffic.

5, Adapt to network conditions. RTDG applications need to degrade gracefully when network resources are constrained, and more attention, experience, and methods are required to accomplish this effectively.

10. CONCLUSION

This survey presents an overview of application-layer networking techniques that have been or could be applied to RTDG to improve performance. This guide is intended to help application developers to design and build RTDG systems that perform well over the Internet, as well as serve as a reference tool for researchers. This survey also identifies several areas for future work in RTDG networking, and presents some of the most critical issues. An additional contribution is the organizational taxonomy of application-layer networking techniques for RTDG, shown in Figure 2. This survey will lead to better RTDG application performance, which will increase the possibilities for interaction techniques and improve usability of RTDG applications when networking resources are constrained.

REFERENCES

[1] Atwood, J., Zhang, Y. A definition of the XTP service and its formal specification. Proc. of the 20th Annual IEEE Conference on Local Computer Networks. 1995.

[2] Baecker, R.M., Wolf, P. and Rankin, K. The ePresence Interactive Webcasting System: Technology Overview and Current Research Issues. Proc. Elearn 2004.

[3] Banerjee, S., Bhattacharjee, B., Kommareddy, C. Scalable Application Layer Multicast. Technical report, UMIACS TR-2002. Available at: http://citeseer.ist.psu.edu/banerjee02scalable.html.

[4] Bargh, M., Ter Hofte, H. Analysis and simulation of the performance of real-time groupware systems : Definition of performance measures and indicators, performance study approach and simulation parameters. (Rep. No. TI/RS/2001/008) Enschede: Telematica Instituut, 2000. Retrieved from http://extranet.telin.nl/dscgi/ds.py/ViewProps/File-10788.

[5] Barrett, S., Tangney, B., Cahill, V. Constructing distributed groupware systems: a walk on the Wilde side. Proc. SIGOPS European workshop on Support for composing distributed applications table of contents. 1998.

[6] Begole, J., Rosson, M., Shaffer, C. Flexible collaboration transparency: supporting worker independence in replicated application-sharing systems. ACM Trans. Comput.-Hum. Interact. 6, 2 (Jun. 1999), Pages 95 - 132. 1999.

[7] Beier, I., Koenig, H. GCSVA - A Multiparty Videoconferencing System with Distributed Group and QoS Management. Proc. IC3N 1998.

[8] Beigbeder, T., Coughlan, R., Lusher, C., Plunkett, J. Agu, E., Claypool, M. The effects of loss and latency on user performance in unreal tournament 2003®, Proc. ACM SIGCOMM 2004 workshops on NetGames '04: Network and system support for games. 2004.

[9] Bharat, K., Cardelli, L. Migratory applications. Proc UIST 1995.

[10] Bhola, S., Banavar, G., Ahamad, M. Responsiveness and consistency tradeoffs in interactive groupware. Proc. ACM symposium on Principles of distributed computing 1998.

[11] Birman, K., Friedman, R., Hayden, M. The Maestro Group Manager: A Structuring Tool For Applications With Multiple Quality of Service Requirements. Cornell University Technical Report. February 1997.

[12] Bolot, J.-C., Fosse-Parisis, S. Towsley, D. Adaptive FEC-Based Error Control for Internet Telephony, Proc. of INFOCOM '99. 1999.

[13] Briot, J-P., Guerraoui, R., Lohr, K-P. Concurrency and distribution in object-oriented programming. ACM Computing Surveys (CSUR), Volume 30 Issue 3, September 1998.

[14] Burridge, R. Java Shared Data Toolkit User Guide. Sun Microsystems, JavaSoft Division. Available from https://jsdt.dev.java.net. 2004.

[15] Caro, A., Amer, P., Conrad, P., Heinz, G. Improving Multimedia Performance Over Lossy Networks via SCTP. Proc. ATIRP 2001.

[16] Chawathe, Y. Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service. Ph.D. Thesis, University of California, Berkeley, Dec. 2000.

[17] Cheok, E., Li, C. Shared Whiteboard for collaborative computing using Java Multicast and RTP. Available at: http://www.cs.columbia.edu/~hgs/teaching/ais/1996/projects/sw/ . 1996.

[18] Chiueh, T., Ballman, A., Pradhan, P. Distributed system support for network-based multi-user interactive applications. In Proceedings of 1st Distributed Simulation Symposium, 1997.

[19] Chockler, G., Huleihel, N., Keidar, I., and Dolev, D. Multimedia Multicast Transport Service for Groupware. TINA Conference on the Convergence of Telecommunications and Distributed Computing Technologies, 1996.

[20] Chockler, G., Huleihel, N., Keidar, I., Dolev, D. Supporting Multiple Quality of Service Options with High Perfomance Groupware. Technical Report CS96-3, Institute of Computer Science, The Hebrew University of Jerusalem, March 1996.

[21] Chu, Y.-H., Rao, S. G., Zhang, H. A Case for End System Multicast. Proc. ACM SIGMETRICS 2000.

[22] Chung, G., Dewan, P. Dynamic architectures: Towards dynamic collaboration architectures. Proc. CSCW 2004.

[23] Civanlar, M., Cash, G. RTP Payload Format for Real-Time Pointers. RFC 2862. June 2000.

[24] Claypool, M., LaPoint, D., Winslow, J. Network Analysis of Counter-strike and Starcraft. Proc. IPCCC 2003.

[25] Crowcroft, J., Vicisano, L., Wang, Z., Ghosh, A., Fuchs, M., Diot, C., Turletti, T. RMFP: A Reliable Multicast Framing Protocol. Internet Draft: draft-crowcroft-rmfp-02.txt, March 1998.

[26] DeCleene, B., Bhattacharaya, S., Friedman, T., Keaton, M., Kurose, J., Rubenstein, D., Towsley, D. Reliable multicast framework (RMF): A white paper, March 1997.

[27] Deering, S., Cheriton, D. Multicast Routing in Datagram Internetworks and Extended LANs. In ACM Transactions on Computer Systems, May 1990.

[28] Dewan, P., Choudhary, R. A high-level and flexible framework for implementing multiuser user interfaces. ACM Trans. Inf. Syst. 10, 4 (Oct. 1992), Pages 345 - 380.

[29] Dewitt, T., Gross, T., Lowekamp, B., Miller, N., Steenkiste, P., Subhlok, J., Sutherland, D. ReMoS: A resource monitoring system for network aware applications. Tech. Rep. CMU-CS-97-194, School of Computer Science, Carnegie Mellon University, December 1997.

[30] Dyck, J., Gutwin, C., and Makaroff, D. Using Behaviour Characteristics to Improve Groupware Performance. University of Saskatchewan Technical Report HCI-TR-2003-01.

[31] Dyck, J., Gutwin, C., Subramanian, S., and Fedak, C. High-Performance Telepointers. Proc. ACM CSCW 2004.

[32] Dyck, J., Gutwin, C. Beyond the LAN: Techniques for Improving Groupware Performance from Networked Games. In development. 2006.

[33] Edwards, W.K. Flexible conflict detection and management in collaborative applications. Proc. UIST 1997.

[34] Enet. Enet Features and Architecture. Available at: http://enet.cubik.org/Features.html. 2003.

[35] Engum, H., Iversen, J., Rein, O. Zereal: A semi-realistic simulator of Massively Multiplayer Online Games. Technical Report available at http://abiody.com/gamemining/publications/2002/ZerealSimulator.pdf. 2002.

[36] Floyd, S., Kevin, F. Router mechanisms to support end-to-end congestion control. Technical report, Feb. 1997.

[37] Francis, P. Yoid: Extending the Multicast Internet Architecture, 1999. White paper http://www.aciri.org/yoid/.

[38] Frecon, E., Greenhalgh, C., Stenius, M. The DiveBone - an application-level network architecture for Internet-based CVEs. Proc. ACM symposium on Virtual reality software and technology. 1999.

[39] Frecon, E., Stenius, M. DIVE: A Scalable network architecture for distributed virtual environments. Distributed Systems Engineering Journal, vol. 5, no. 3, pp. 91-100, 1998.

[40] Funkhouser, T. Network topologies for scalable multi-user virtual environments. Proc. of the Virtual Reality Annual International Symposium, pages 222–8. 1996.

[41] GarageGames. Torque Network Library Design Fundamentals. TNL 1.5.0, 23 Feb 2005. Available at: http://opentnl.sourceforge.net/doxydocs/fundamentals.html

[42] Geyer, W., Effelsberg, W. The Digital Lecture Board – A Teaching and Learning Tool for Remote Instruction in Higher Education. Proc. of the 10th World Conference on Educational Multimedia (ED-MEDIA) `98. 1998.

[43] Gracanin, D., Zhou, Y., DaSilva, L. Quality of Service for Networked Virtual Environments. IEEE Communications Magazine April 2004.

[44] Graham, T.C.N., Urnes, T., Nejabi, R. Efficient Distributed Implementation of Semi-Replicated Synchronous Groupware. Proc. UIST 1996.

[45] Greenberg, S. and Marwood, D. Real time groupware as a distributed system: Concurrency control and its effect on the interface. Proc. CSCW 1994.

[46] Greenberg, S. and Roseman, M. Groupware Toolkits for Synchronous Work. In M. Beaudouin-Lafon, editor, Computer-Supported Cooperative Work (Trends in Software 7), Chapter 6, p135-168, John Wiley & Sons Ltd, ISBN 0471 96736 X. 258pp. 1999.

[47] Greenhalgh, C. Dynamic, embodied multicast groups in MASSIVE-2. Technical Report Department of Computer Science, The University of Nottingham NOTTCS-TR-96-8. 1996.

[48] Greenhalgh, C., Benford, S., Craven, M. Patterns of network and user activity in an inhabited television event. Proc. ACM symposium on Virtual reality software and technology 1999.

[49] Groove. Available at http://www.groove.net.

[50] Grudin, J. Why CSCW applications fail: problems in the design and evaluation of organization of organizational interfaces. Proc. CSCW 1998.

[51] Gutwin, C. Effects of Network Delay on Group Work in Shared Workspaces. Proc. ECSCW 2001.

[52] Gutwin, C., and Greenberg, S. The Effects of Workspace Awareness Support on the Usability of Real-Time Distributed Groupware. ACM Transactions on CHI 1999.

[53] Gutwin, C., Greenberg, S. The Mechanics of Collaboration: Developing Low Cost Usability Evaluation Methods for Shared Workspaces. IEEE WETICE 2000.

[54] Gutwin, C., Dyck, J., Burkitt, J. Using cursor prediction to smooth telepointer jitter. Proc. GROUP 2003.

[55] Gutwin, C., Benford, S., Dyck, J., Fraser, M., Vaghi, I., and Greenhalgh, C. Revealing Delay in Collaborative Environments. Proc. CHI 2004.

[56] Handley, M., Pahdye, J., Floyd, S., Widmer, J. TCP Friendly Rate Control (TFRC): Protocol Specification. Internet draft draft-ietf-tsvwg-tfrc-02.txt, work in progress, May 2001.

[57] Hellstrom, G. RTP Payload for Text Conversation. RFC 2793. May 2000.

[58] Hill, R.D., Brinck, T., Rohall, S.L., Patterson, J.F., Wilner, W. The Rendezvous architecture and language for constructing multi-user applications. ACM Transactions on Computer-Human Interaction, 1(2), June, 1994.

[59] Hosseini, M., Georganas, N. Suitability of MPEG4's BIFS for Development of Collaborative Virtual Environments. Proc. 10th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises. 2001.

[60] Hosseini, M., Georganas, N.D. End System Multicast Protocol for Collaborative Virtual Environments. J. PRESENCE: Teleoperators and Virtual Environments, (Special Issue on Advances in Collaborative Virtual Environments), MIT Press, Vol. 13, Issue. 3, June 2004.

[61] Iren, S. Network-Conscious Image Compression PhD Dissertation, Univ of Delaware, 1999.

[62] ISO/IEC. Quality of Service: Framework. ISO/IEC 13236:1998. Available from http://www.iso.org. 1998.

[63] ITU-T. Guidance on one-way delay for Voice over IP. In ITU-T g.114. Available from http://www.itu.int. 2000.

[64] Joyce, S. Traffic on the Internet --- Report. Available at citeseer.ist.psu.edu/joyce00traffic.html. 2000.

[65] Kangxin, Z., Jianhua, L. Hongwen, Z. Sender Delay Comparison of IP-Based Reliable Synchronous Collaboration Communication. Proc. ICCT 2000.

[66] Kohler, W. A Survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems. ACM Comput. Surv. 13(2): 149-183 (1981).

[67] Lea, R., Honda, Y., Matsuda, K., Matsuda, S. Community place: Architecture and performance. In Proceedings of the 2nd Symposium on Virtual Reality Modeling Language, pages 41–50. 1997.

[68] Lee, K. Adaptive network support for multimedia. Proc. ACM MOBICOM, 1995.

[69] Liang, Y., Steinbach, E., Girod, B. Real-time voice communication over the internet using packet path diversity. Proc. Multimedia 2001.

[70] Liang, Y., Setton, E., Girod, B. Channel-Adaptive Video Streaming Using Packet Path Diversity and Rate-Distortion Optimized Reference Picture Selection. IEEE Workshop on Multimedia Signal Processing Dec. 2002.

[71] Litiu, R., Prakash, A. Developing adaptive groupware applications using a mobile component framework. Proc. CSCW 2000.

[72] Marsic, I. Real-Time Collaboration in Heterogeneous Computing Environments. Proc. ITCC 2000, p222-227.

[73] Mathur, A.G., Prakash, A. A Protocol Composition-Based Approach to QoS Control in Collaboration Systems. Technical Report CSE-TR-27495, U. of Michigan, Ann Arbor, MI, Dec. 1995.

[74] Matta, I., Eltoweissy, M., Lieberherr, K. From CSCW Applications to Multicast Routing: An Integrated QoS Architecture. Proc. IEEE ICC'98, Atlanta, GA, June 1998.

[75] Mauve, M., Hilt, V.Kuhmünch, C., Effelsberg, W. A General Framework and Communication Protocol for the Transmission of Interactive Media with Real-Time Characteristics. Proceedings of the IEEE International Conference on Multimedia Computing and Systems Volume II-Volume 2, p.641, June 07-11, 1999.

[76] Mauve, M., Hilt, V., Kuhmünch, C., Vogel, J., Geyer, W., Effelsberg, W. RTP/I: An Application Level Real-Time Protocol for Distributed Interactive Media. IEFT Draft. March, 2000.
Available at: http://www.informatik.uni-mannheim.de/pi4/lib/projects/rtpi/papers/rtpi.txt.

[77] Mauve, M., Hilt, V., Kuhmünch, C. and Effelsberg, W. RTP/I - Toward a Common Application Level Protocol for Distributed Interactive Media. IEEE Transactions on Multimedia, 2001.

[78] McCanne, S., Jacobson, V., Vetterli, M. Receiver-Driven Layered Multicast, Proc. ACM SIGCOMM '96, Aug. 1996.

[79] Mills, K. Introduction to the electronic symposium on computer-supported cooperative work. ACM Computing Surveys (CSUR), Volume 31 , Issue 2 (June 1999).

[80] Miranda, H., Rodrigues, L. Flexible Communication Support for CSCW Applications. String Processing and Information Retrieval Symposium. International Workshop on Groupware. September 21 - 24, 1999.

[81] Nichols, J., Claypool, M. The effects of latency on online madden NFL football, Proceedings of the 14th international workshop on Network and operating systems support for digital audio and video, June 16-18, 2004.

[82] O'Grady, T. Flexible data sharing in a groupware toolkit. Master's thesis, University of Calgary, Calgary, Alberta, Canada, November 1996.

[83] Pantel, L., Wolf, L. C. On the impact of delay on real-time multiplayer games. Proc. NOSSDAV 2002.

[84] Park, K., Kenyon, R.V. Effects of Network Characteristics on Human Performance in the Collaborative Virtual Environment. IEEE Virtual Reality '99 Conference. 1999.

[85] Pendarakis, D., Shi, S., Verma, D., Waldvogel, M. ALMI: An Application Level Multicast Infrastructure. Proc Usenix Symposium on Internet Technologies & Systems, March 2001.

[86] Perkins, C., Crowcroft, J. Notes on the use of RTP for shared workspace applications. ACM Computer Communication Review, Volume 30, Number 2, April 2000.

[87] Perkins, C., Hodson, O., Hardman, V. A Survey of Packet Loss Recovery Techniques for Streaming Audio. IEEE Network, Sept/Oct 1998.

[88] Phillips, W.G. Architectures for Synchronous Groupware. Technical Report 1999-425. Department of Computing and Information Science, Queen's University, Kingston, Ontario, Canada. 1999.

[89] Pritchard, M. How to Hurt the Hackers: The Scoop on Internet Cheating and How You Can Combat It. Information Security Bulletin, February 2001.

[90] Quax, P., Monsieurs, P., Lamotte, W., De Vleeschauwer, D., Degrande, N. Objective and subjective evaluation of the influence of small amounts of delay and jitter on a recent first person shooter game. Proc. SIGCOMM 2004 workshops on NetGames '04: Network and system support for games. 2004

[91] Rakkarsoft. Raknet Manual. Available at: http://www.rakkarsoft.com/raknet/manual/. 2004.

[92] Ratnasamy, S., Handley, M., Karp, R. Shenker, S. Application-level multicast using content-addressable networks. In Proceedings of 3rd International Workshop on Networked Group Communication. 2001.

[93] Rhee, I., Cheung, S., Hutto, P., Sunderam, V. Group communication support for distributed collaboration systems. In Proceedings of the 17th International Conference on Distributed Computing Systems. 1997.

[94] Roberts, D., Sharkey, P. Maximising Concurrency and Scalability in a Consistent, Causal, Distributed Virtual Reality System Whilst Minimising the Effect of Network Delays. Proceedings of the 6th Workshop on Enabling Technologies on Infrastructure for Collaborative Enterprises. 1997.

[95] Robinson, J., Stewart, J. RTP Payload format for Shared Multicast Virtual Worlds (SMVW). Internet Draft, June 1999. Available at http://www.cs.columbia.edu/~hgs/rtp/drafts/draft-stewart-avt-00.txt

[96] Sandkuhl, K., Messer, B. Towards Reference Architectures for Distributed Groupware Applications. 8th Euromicro Workshop on Parallel and Distributed Processing. 2000.

[97] Savage, S., Collins, A., Hoffman, E., Snell, J., Anderson, T. The end-to-end effects of Internet path selection. Computer Comm. Review, vol. 29, no. 4, pp. 289–99, Oct. 1999.

[98] Sisalem, D., Schulzrinne, H. The Loss-Delay Adjustment Algorithm: A TCP-friendly Adaptation Scheme. NOSSDAV 1998.

[99] Smed, J., Kaukoranta, K., and Hakonen, H. A Review on Networking and Multiplayer Computer Games. Technical Report 454, Turku Centre for Computer Science, 2002.

[100] Summers, B. Official Microsoft NetMeeting book. Redmond, WA: Microsoft Press, 1998.

[101] Sun, C., Ellis, C. Operational transformation in real-time group editors: issues, algorithms, and achievements. Proc. CSCW 1998.

[102] Sun, C. Optional and Responsive Fine-grain Locking in Internet-based Collaborative Systems. IEEE Transactions on Parallel and Distributed Systems, Vol. 13, No. 9, Sept. 2002.

[103] ter Hofte, H. Working Apart Together: Foundations for Component Groupware. Number 001 in Telematica Instituut Fundamental Research Series. Telematica Instituut, Enschede, the Netherlands, 1998. ISBN 90-75176-14-7. Also available from http://www.telin.nl.

[104] Turletti, T., Parisis, S., Bolot, J. Experiments with a layered transmission scheme over the Internet. Technical report RR-3296, INRIA, France. 1997.

[105] Urnes, T., Graham, N. Flexibly Mapping Synchronous Groupware Architectures to Distributed Implementations. Proc. of the 6th International Workshop on the Design, Specification and Verification of Interactive Systems (DSV-IS '99, Braga, Portugal, June 2-4). Springer-Verlag, 1999.

[106] Vaghi, I., Greenhalgh, C., Benford, S. Coping with inconsistency due to network delays in collaborative virtual environments. Proc. VRST 1999.

[107] Vera, A., Kvan, T., West, R., Lai, S. Expertise, collaboration and bandwidth. Proc. Human factors in computing systems, 1998.

[108] Vicisano, L., Rizzo L., Crowcroft, J. TCP-like Congestion Control for Layered Multicast Data Transfer. Proc. INFOCOM 98.

[109] Webb, S. A Survey of Cheating Techniques in Online Games. From CS 7001 class project at Georgia Tech in 2004. Available at http://home.cc.gatech.edu/webb/uploads/10/MiniProject3.pdf

[110] WebCT. Available at http://www.webct.com.

[111] Whalen, T., Black, J. Adaptive Groupware for Wireless Networks. Proceedings of the Second IEEE Workshop on Mobile Computer Systems and Applications. 1998.

[112] Wright, T., Boria, E., Breidenbach, P. Creative Player Actions in FPS Online Video Games: Playing Counter-Strike. In The International Journal of Computer Game Research, Volume 2, Issue 2, December 2002. Available at: http://www.gamestudies.org/0202/wright/

[113] Yan, J. J., Choi, H.-J. Security issues in online games. In Proceedings of International Conference on Application and Development of Computer Games in the 21st Century. 2001.

[114] Yang, Y., Li, D. Separating Data and Control: Support for Adaptable Consistency Protocols in Collaborative Systems. ACM CSCW'04 Conference. 2004.

[115] Zhuang, S. Q., Zhao, B. Y., Joseph, A. D., Katz, R., Kubiatowicz, J. Bayeux: An architecture for scalable and fault-tolerant widearea data dissemination. Proc. NOSSDAV 2001.

GLOSSARY OF ACRONYMS

ACK: Acknowledgement

AFEC: Adaptive Forward Error Correction

CSCW: Computer Supported Collaborative Work

DoI: Degree of Interest

CVE: Collaborative Virtual Environment

FEC: Forward Error Correction

GCS: Group Communication Specification

LAN: Local Area Network

MAC: Medium Access Control

NACK: Negative Acknowledgement

OpenTNL: Open source version of the Torque Networking Library

OSI: Open System Interconnection

QoS: Quality of Service

Raknet: A game networking library

RTDG: Real-time distributed groupware

RTP: Real-time transport protocol

RTP/I: Interactive extension of RTP

SCTP: Stream Control Transmission Protocol

TCP: Transport Control Protocol

UDP: Unreliable Datagram Protocol

VoD: Video on Demand

VoIP: Voice over Internet Protocol

XTP: eXtendible Transport Protocol