

Payoff-Based Communication Adaptation based on Network Service Availability

Robin Kravets* Ken Calvert † Karsten Schwan ‡

Abstract

Interactive applications will continue to push the limits of available network bandwidths, processing speeds, and other computing resources. These applications present challenges concerning the specification of service requirements for their multiple types of data transfer and the online management of service quality. In this paper, we present a cooperative solution in which a configurable communication layer is used to adapt communication in response to both application requirements and network resource availability. Adaptation decisions are based on (1) *payoff functions* which capture the requirements of the application in a functional form, and (2) *service availability curves* which represent network resource availability. Experimentation with representative multimedia applications demonstrates the benefit of using such payoff-based adaptation.

1 Introduction

Distributed applications operating in dynamic network environments may experience unpredictable changes in resource availability. Without information about the state of the network, there is little an application can do to compensate for such changes. Similarly, without information about the requirements of the application, the communication infrastructure has limited opportunities to improve the match of application requirements with network resources. Our work proposes a cooperative solution in which the communication layer uses both application resource requirements and information about network resource availability to determine how to configure the communications.

The intent is to enable applications to operate within acceptable parameters despite potentially insufficient network resources. This research addresses interactive distributed applications able to exploit application-level semantics to specify the importance of the transfer of information between their distributed components. Specifically, these applications may require high bandwidth data transfers, but they can also utilize tradeoffs in transmission quality vs. transfer time to improve the services they offer to end users. One such application is a distributed virtual environment application requiring the transfer of large amounts of data originating from many different objects in the virtual world, utilizing audio, video and image data and concurrently performing multiple data transfers. This application may be willing to accept lower levels of quality, which correspond to reduced communication requirements, for virtual objects that are “far away” in order to realize improvements in the performance of accessing “closer” objects. Specifically, the data from an image object may be transmitted with lower quality when it is not currently important to the user looking at it.

We support such applications by providing an adaptive communication layer that configures communication protocols based on the benefit the communication layer expects to realize for the application. The key to such support is an understanding of the possible tradeoffs in communication and the fashion in which these

*College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA, robink@cc.gatech.edu. Support for this work is sponsored in part by an AT&T/Lucent Technologies PhD Fellowship.

†College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA, calvert@cc.gatech.edu

‡College of Computing, Georgia Institute of Technology, Atlanta, Georgia, USA, schwan@cc.gatech.edu

tradeoffs benefit the application. The solution approach chosen in our work is one in which the application specifies its resource requirements in terms of *payoff functions*, which capture the value of the data being transferred by the application in a functional form. The network provides detailed resource specification in the form of *service availability curves* which describe the range of available services from the network. The communication layer incorporates both types of information to determine how to adapt to changes in application requirements or network resource availability.

By permitting applications to share dynamic “service quality” requirements with the communication layer, this layer is able to make intelligent decisions about the communication configuration suitable for the current state of the application. The specific service parameters considered in this paper concern the reliability of data transfer. Using a variable reliability protocol, the communication layer is provided with the means to adapt communication based on information about the reliability required for the current data being transmitted. In addition, by monitoring dynamic resource availability from the network, the communication layer is able to make intelligent decisions about the communication configuration best suited for the currently available network resources. The specific network resource information utilized in this paper may be obtained from *Loss-Load Curves* [WC91, Wil96] or via network flow metering [BMR97]. The configuration decisions derived from application service specification and network resource monitoring concern the appropriate choice of the reliability level of data transfer.

The novel techniques proposed in this paper address a large class of distributed multimedia applications. These applications share the characteristic that they are able to continue working in less than ideal situations, and can do so by reducing certain service requirements in order to improve others. This class of applications includes video conferencing systems [AMZ95], web applications [BHD⁺96], distributed interactive simulations [HW96] and distributed virtual environments [OSF⁺97].

The remainder of this paper addresses the issues involved in communication adaptation, focusing on incorporating expanded network resource availability specification into our communication layer. Section 2 describes the problems facing distributed applications that operate in dynamic network environments. Section 3 describes our communication layer and the specific application used in this paper. This section also defines *payoff functions*, a mechanism for specifying value-based application service requirements, and *service availability curves*, a technique for specifying dynamic network resource availability. We then present our techniques for payoff-based adaptation of communications. Section 4 describes the results of experiments using these techniques. Section 5 concludes the paper and describes future research.

2 Background

Present distributed applications are not typically designed to adapt themselves to changes in network conditions. Instead, such applications tend to rely on network adaptation at the transport level, such as TCP’s congestion control mechanism, which adapts transmission rate in response to perceived congestion in order to minimize packet loss. These adaptations are based on the assumption that the application is willing to accept additional overhead in transmission time in order to receive the transmitted data at one hundred percent reliability. However, they do not consider the requirements of specific applications and therefore, take actions that may be contradictory to the needs of the application. This is because these adaptations only use knowledge about the state of the network obtained from observing the current packet stream. In comparison, better decisions would be made by a communication layer sensitive to specific application needs with knowledge about the general behavior of network resources.

A simple example of using appropriate communication quality is the transfer of an image object in a distributed virtual environment. In this context, quality may refer to the reliability of the data being transmitted, the lossiness of a compression scheme, or the timeliness of data transmissions. The application is willing to accept less than one hundred percent transmission reliability up to some threshold. Such compromises in reliability may benefit the application when the available networking resources fluctuate since a fully reliable protocol will impose extra delay when the packet loss rate increases. In comparison, a protocol

that understands that the data need not arrive reliably will only ask for retransmissions if the acceptable quality threshold is crossed. Therefore, by allowing the application to place values on both reliability and transmission times, the communication layer can consider the effects of increased or decreased loss rates on the transmission time and choose appropriate reliability levels.

Concerning the network, it is beneficial to consider the effects of current communication characteristics, like loss rate and available bandwidth, on the service provided to the application. For example, in a congested network, the loss rate typically rises with increases in offered load. An application that has no knowledge of the presence of congestion in the network will not know when to reduce its requested transmission speed. Consider the image object in the distributed virtual environments application, where the application will try to transmit the image data as it is produced. In the presence of congestion, the application may experience losses similar to those shown in the upper curve in Figure 1. The delays caused by the recovery of lost messages at such loss rates may not be acceptable. Alternatively, if lost messages are not recovered, the amount of loss may also not be acceptable to the application. This implies that the use of information about current network state may enable adjustments in transmission rates that result in acceptable loss and delay, as indicated by the lower curve in Figure 1.

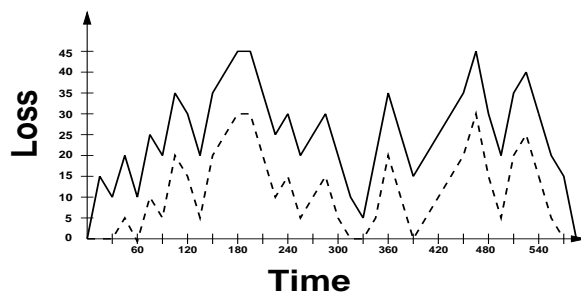


Figure 1: Example Network Loss Over Time

3 Problem Domain and Description

Our work addresses applications with dynamic service requirements that operate in environments with unpredictable resource behavior. Toward this end, we have developed techniques for quality of service specification which can be used as the basis for dynamic communication adaptation. Using these techniques, we show how configurable communication protocols can be used to adapt communication to changes in application requirements and network resource availability. This section first introduces our communication layer and one of the target applications for which it is intended. Next, we present *payoff functions*, a service specification technique which enables application valuation of communication parameters. These valuations are used in conjunction with dynamic resource information, in the form of *service availability curves*, to explore the effects of tradeoffs in competing communication parameters. The results are used as a basis for runtime reconfiguration of protocol stacks.

3.1 Communication Layer

We consider the design and implementation of an end-to-end *communication layer*, which mediates between an application and a network service, as depicted in Figure 2. The purpose of the communication layer is to map between the services available from the network and the services needed by the application. The application submits data units to the end-to-end communication layer for transmission; the communication layer processes them and then interacts with the network service to transfer the information through the network. The communication layer may then process the transferred information further on the receiving

side before ultimately passing data units up to the application. Thus, the communication layer implements a channel using the given network service, and makes it available to the application. The characteristics of this channel —as quantified by measures such as throughput, delay, delay jitter, loss rate, and cost, and by semantic characteristics such as order preservation— determine its *value* to the application.

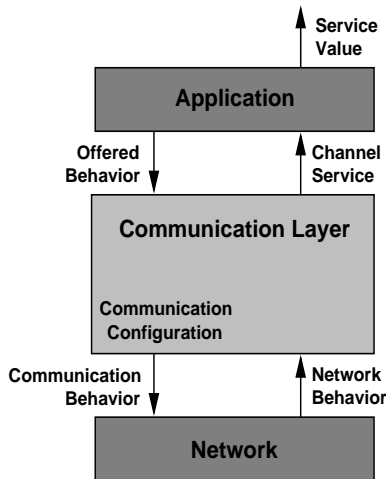


Figure 2: Communication Architecture

Our goal is to design the communication layer to adapt its behavior to maximize service value in the face of changing application needs and network characteristics. We can view this as an optimization problem, the structure of which is depicted in Figure 2. The communication layer has access to several sets of interrelated variables, some of which it can control. The relevant sets of variables are:

- The application’s *offered behavior*, which traditionally is quantified by parameters such as rate and burstiness of data generation, but may also be characterized by information about the type and importance of data being transferred. The application controls the offered behavior.
- The communication layer’s *communication configuration*, which captures the set of protocols and mechanisms used by the communication layer to enhance the service received from the network. For example, this variable captures whether forward error control or retransmission is used for error recovery. The communication layer controls the communication configuration directly.
- The communication layer’s *communication behavior*, i.e. what the network “sees” from the communication layer. This includes traditional QoS dimensions such as peak and mean transmission rate, burst length, etc. The communication layer controls communication behavior directly.
- The *network service*, i.e. what the communication layer “sees” from the network on the receiving side. This is quantified by parameters including loss, delay, jitter, and cost. The parameters for network service are determined by the network, but are affected directly by the communication behavior. For example, an increased transmission rate over some interval may result in an increase in cost over that interval.
- The *channel service*, i.e. the behavior actually seen by the application. This will include quantities similar to network service, but measured at the interface between the communication layer and the application. The channel service is a function of communication behavior and communication configuration.

In general these parameters will represent system behavior over some time interval, rather than instantaneous values. For the purposes of this discussion, we view them as single-valued (i.e. points in some multidimensional space).

The channel's *service value* is a function of the channel service parameters. In order to solve the optimization problem of maximizing service value, the communication layer examines three relationships.

- The relationship between channel service and service value. We assume this is defined by a function which is provided by the application. This function defines the quantity to be maximized.
- The relationship between network service and communication behavior. In general, this can be defined by a curve (or plane) giving the value of the communication behavior that can be expected for each value of network service. We assume that the network service provides this curve.
- The relationship between communication configuration, communication behavior, network service and channel service, i.e. how the channel service delivered to the application is affected by the communication layer's protocol configuration and communication characteristics and the network service. For example, the use of retransmission as an error control strategy can decrease loss rate, but may increase delay. The use of forward error control, on the other hand, can decrease loss at the expense of increased bandwidth. (The assumption is that increasing delay or bandwidth decreases value.)

The characterization of this last relation is a major challenge in the design and implementation of the communication layer. It might be realized through analysis, simulation, measurement, or (most likely) some combination of all three.

After discussing the target application, the remainder of this section describes an empirical approach to the implementation of such a communication layer. Section 3.3 discusses our approach for specifying the value of the channel service to the application. In section 3.4 we address some of the issues involved in network service specification. And in section 3.5, we present our techniques for using the information provided by these relationships to adaptively configure the communication.

3.2 Target Application

The specific application addressed in this paper is a distributed virtual environment operating across the Internet. The environment was designed as a collaboration tool for supporting concurrent interactions among multiple users. The environment is a complex simulation in which users can navigate through a virtual world organized into multiple rooms. The virtual world as well as the objects in the world are represented using graphical, audio, video and data visualization techniques. The users interact with objects in the world, with the world itself, and with the other users in the world. Although the rooms may be separated by walls, there may also be doors and windows present that allow the user to see parts of other rooms or hear pieces of conversations being conducted elsewhere. Objects themselves may be stationary, movable, or they may even move on their own (i.e. other users or autonomous agents). Objects may be simple or complex graphical representations, still or motion images, or data visualizations, and they may also have audio associated with them.

The dynamic nature of the virtual world coupled with runtime changes in users' interests results in frequent and dynamic changes in users' service requirements. For example, while a user is relatively "far" from a video monitor visible in the environment, there is no need to offer high resolution real-time images for display on this monitor. However, both display rate and image quality must be improved to simulate the fact that the user can see the image more clearly as the user approaches the monitor.

The communications in our distributed virtual environment fall into three categories. The first category includes data that must be transmitted reliably, including control messages and the one time transmission of some object descriptions. This type of data is very sensitive to packet loss and is never willing to sacrifice reliability. The second category is comprised of data streams, where data must be broken up into multiple messages in order to be transmitted. Examples of this type of data are movies, changing texture maps, or large scientific data. This type of data often offers some level of redundancy. At the application level,

it may be possible to compensate for data loss by inferring the lost data from other messages that are “spatially” close to it. For example, a lost message in an image can often be recreated by examining the image surrounding the lost message and performing some type of averaging function. Such recovery has limitations in the number of “nearby” or consecutive data losses it can tolerate. The third category consists of continuous streams of relatively small messages, each of which is an update to a specific piece of application data. Examples of this type of data are object position updates or tracker updates, which have the property that, if a message is lost, the next message will provide the newest information for the data. Due to this property, there is little tolerance for waiting for the retransmission of lost data, since the application can approximate much of the lost data from the next message much faster than it could receive the original data from a retransmission. Although continuously updated, this type of data is very sensitive to loss due to the effect on the lag perceived by the user.

The techniques described in the remainder of this section address the first two categories of data transmitted in virtual environments, thereby targeting a large class of common multimedia applications. The third class of data, continuously transmitted updates, will be addressed in our future research.

3.3 Application Service Specification

Service requirements in virtual environments vary for each user and across multiple users. For instance, in the virtual world organized as a set of rooms, at any one time, a user may only be interested in one specific room, perhaps with cursory interests in surrounding rooms. We believe that it is important that such application-level knowledge about information needs be available when configuring the communication for the application. Toward this end, we offer user-level specifications of the *value* of parameters of the communication service to the application. To clarify, consider an image of interest to the user. Clearly, the quality of the image and, therefore, the resolution at which it must be transmitted is directly related to the user’s current level of interest in the image’s contents. In this case, the actual “value” ascribed to the image’s transmission may be computed with some application-specific “value function” that uses parameters quantifying the user’s current level of interest (e.g., distance from the image in the virtual world) and quantifying the actual level of service at which the image is transmitted by the network.

With a “value function”-based specification of desired service quality, it is possible to capture the tradeoffs the user is willing to make to realize certain actual levels of service. Typically, such tradeoffs represent relationships between conflicting service parameters. For example, in a lossy network, transfer rate may be sacrificed for a high reliability transfer. To study the utility of value functions in making such tradeoffs, we utilize particular value functions, called *payoff functions* in the remainder of this paper. Payoff functions are associated with specific connections, and each such function states the value to the application of receiving a certain level of service via this connection. Specifically, from the application’s point of view, a point in the payoff function represents the benefit gained by the application of achieving that specific level of service quality. Many such functions exist and may be formulated for the diverse multimedia applications considered in our work, and in previous work, such functions have been formulated for real-time applications[JLT85].

Figure 3(a) depicts two sample payoff functions addressing transmission reliability. The plot for image 1 represents a payoff function for an image that has been requested at a range from 100 percent reliability down to 75 percent reliability. The shape of the curve within the range of 75 to 100 percent reliability represents the relative value to the application of achieving that level of reliability. In this example, the application prefers 100 percent reliability and will still be somewhat satisfied at 90 percent reliability, but will be less satisfied as the reliability approaches 75 percent. This is represented in the payoff function by having the payoff for reliability highest when 100 percent reliability is achieved. The payoff slowly diminishes until the reliability level reaches 90 percent and then drops off quickly until the payoff is zero at the limit of acceptable reliability, 75 percent. In comparison, if the image can be accepted anywhere from 40 to 80 percent, then the maximum payoff will be at the 80 percent reliability level as shown in the plot for image 2 in figure 3(a). If the image is transmitted with more than 80 percent reliability, there is no additional payoff, since the application is not requesting a higher payoff level. This payoff will also diminish until it reaches the limit

of acceptable quality, in this case 40 percent. The general scale shown in these examples always places the payoff between 0 and 1. This allows us to normalize the payoff functions between service parameters.

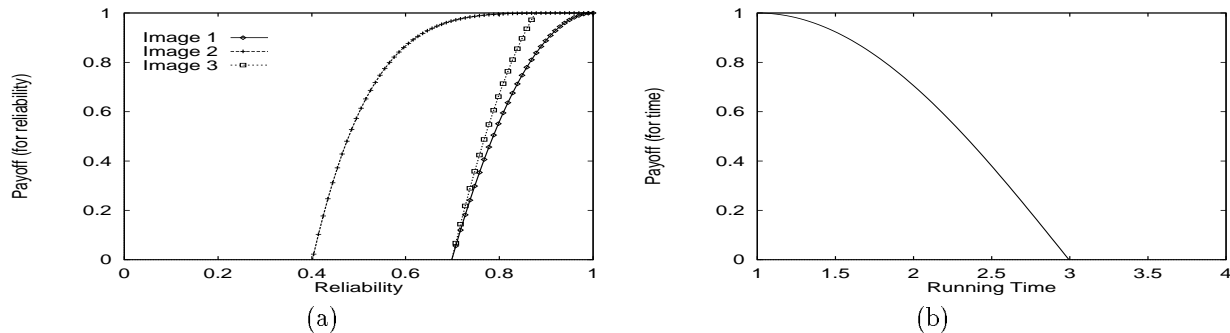


Figure 3: Sample Payoff Functions for Reliability and Time

The specification of a payoff function for one service parameter does not capture the tradeoffs the application is willing to make with respect to other service parameters in order to achieve that service level. To do this, the application must supply a second payoff function for a service parameter which represents some notion of “cost” to the application. In other words, the increased service level of the first parameter will have a negative effect on the service level of the second parameter. For illustration, again consider transfer time vs. reliability. Figure 3(b) represents a payoff function for transfer time. In this example, the application has requested a transfer rate from 1 time unit to 3 time units. As we can see from the payoff function, the shorter transfer time is valued higher than the longer transfer time. The application has now specified payoff functions for two competing service parameters. In a lossy network, losses will cause the payoff for reliability to decrease, but recovering those losses will increase the transfer time and decrease the payoff for transfer time. By supplying payoff functions for both parameters, the application provides the information necessary to determine how to find an acceptable operating point.

Payoff functions can also be used to represent the comparative value between two pieces of data. Consider two images of interest to the user, where the payoff functions for reliability for both of these images are those depicted by the plot for image 1 in figure 3(a). Such a formulation is adequate if the interest level of the user is the same for both images. In the case where one image is of less interest to the user, payoff for that image may be adjusted to reflect the level of interest, as shown in the plot for image 3. For this image, the payoff function has been scaled up to a point where the maximum payoff is achieved for a lower reliability level, but the curve still captures the shape of the original curve and maintains the original minimum acceptable reliability level. This scaling is performed to minimize the effect of a lower service level in the image that is of less interest.

The utility of payoff functions should be clear from the examples presented above: they may be used to present the providers of communication services with application-level semantics with which differences in requested vs. delivered levels of service may be evaluated. In general, there is a payoff function, $P_{(l,d)}^p$, for the service parameter, p , associated with each service level, l , and data, d , pair. In this paper we focus on the service parameters end-to-end loss and transfer time. Given such service parameters and the payoff functions formulated in this section, we next discuss how to describe resource availability appropriately so that payoff may be maximized for these services.

3.4 Communication Resource Specification

Applications typically have limited knowledge about the availability of communication resources. This lack of information forces the application to guess at the best way to behave. For example, a user of the virtual environment may be receiving object updates and a video stream. When communication resources are limited, both the video stream and object updates will be negatively affected, one by dropping frames, the

other by losing object updates. Such behavior is undesirable, especially if it can be remedied at the application level by lowering the quality of the video stream in order to reduce bandwidth needs. Similarly, when communication resources become available, it should be possible for the application to respond by improving video quality. In general, then, opportunities for optimization are created by exposing communication resource information in order to coordinate with application-level service information.

The specific optimizations considered in our work focus on how an application can benefit from knowledge about the relationship between requested service and the actual quality of that service. Toward this end, the application is supplied with a set of transfer rates and their respective loss rates, which jointly describe the current behavior of the available communication resources. In general, this implies that the application has knowledge about what quality of service to expect for a range of different service requests. This relationship can be represented through the use of *service availability curves*. This type of service has been proposed [WC91] in the form of Loss-Load Curves. Loss-load curves characterize the dynamic service a network can provide to its clients. Given a specific transmission rate for a sender, loss-load curves provide the sender with an expected loss rate. This information allows the application itself to determine the tradeoff between higher output rates and higher loss rates. When service availability curves are not available, communication resource availability may be determined by the application itself, perhaps by gathering statistics based on recent communication history and by projecting such behavior into the near future [Bol].

The sample loss-load curve depicted in figure 4 exhibits an increasing loss rate as the application increases its transfer rate. In this example, the network provides a fixed set of possible transfer rates and supplies the respective loss rates. 10 percent loss is imposed for a transmission rate of 400KBps. The loss rate increases 5 percent for each 100KBps increase in transmission rate.

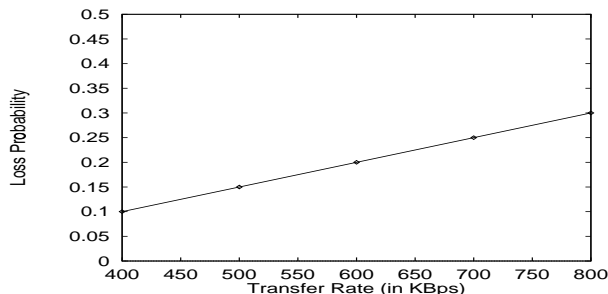


Figure 4: Example Loss-Load Curve

3.5 Communication Adaptation

Payoff and service availability functions provide the communication framework with extensive information about an application’s service requirements and the network’s service availability. The topic of this section is the use of such information to configure communications during application execution. Specifically, we demonstrate the use of *payoff adaptation* to optimize the use of available communication resources. The payoff adaptation method employed in this work is reactive. Specifically, it responds to changes in application requirements and network resources rather than attempting to anticipate them, as in predictive [BS91, CKV93] and in learning-theory based adaptation methods [KLV95, KLP⁺95]. Payoff adaptation techniques allow us to balance application requirements like quality level and running time with resource availability. Resource information may be defined as execution time, available bandwidth, or message loss rate.

Two stages of adaptation. Payoff adaptation involves two stages. In the first stage, the communication layer determines a communication configuration for a specific point in time, given current application requirements and network resource availability. In the second stage, the communication layer monitors changes in application requirements and network resources. The effects of such changes on application performance are considered to determine a new configuration.

1. *Using payoff functions to evaluate resource requirements.* This first stage of payoff adaptation uses the application-supplied payoff functions to evaluate application resource requirements. Given a fixed application resource specification and a fixed network resource specification, payoff adaptation considers a range of possible operating parameters and chooses those that best fit the application. As an example, again consider the plot for image 2 in figure 3 (a), which depicts a payoff function for an image that has lower value to the user. By including the information supplied by the loss load curve in figure 4, the communication layer can determine the effect of operating at a specific set of transmission rates and their respective loss rates. This determination is made by evaluating the service parameters and determining the payoff for those values.

To evaluate individual service parameters, we need information about the inter-relationship between multiple communication variables as discussed in section 3.1. This relationship may be obtained via a function $f(\text{State}) \rightarrow \text{Service}$. In the context of our example, this function supplies the mapping from transmission rate and loss rate to image quality, $f(\text{TransmissionRate}, \text{LossRate}) \rightarrow \text{ImageQuality}$. In general, such inter-relationships may be collected using profiling techniques prior to running the application, from an analytical model, or they may be gathered by the application during execution (e.g., based on recent history). Once the communication layer has evaluated the service parameters for each operating point, it may then evaluate the benefit to the application of operating with these service parameters. To make this determination, all valued resource parameters are examined in order to determine the net payoff for a given configuration. As discussed in section 3.3, there is a payoff function, $P_{(l,d)}^p$, for each service parameter associated with a service level and data pair.

Since the application indicates multiple service parameters, the payoff for each of these parameters must be considered to determine a net payoff. In other words, the net payoff, P_{total} , is a function, Q , over all the defined $P_{(l,d)}^p$, where p represents the service parameter, l represents the service level and d represents the specific data in question. If we consider an image with two payoff functions, one for reliability and one for transmission time, the payoff for the image, i , is determined by evaluating the payoff functions, $P_{\text{rel}_k,i}^{\text{rel}}$ and $P_{\text{time}_k,i}^{\text{time}}$, at each transmission rate, k . By using the relationship functions described in the previous paragraph, the communication layer can determine the appropriate reliability and transmission time, rel_k and time_k , expected for each transmission rate, k . The goal is for the communication layer to choose the transmission rate that maximizes the total payoff. The communication layer determines the total payoff for each transmission rate as a function of $P_{s_k,i}^{\text{time}}$ and $P_{s_k,i}^{\text{quality}}$ and chooses the k that maximizes total payoff. The values for the individual payoffs are determined from the application supplied payoff functions. The function for composing payoffs for multiple application parameters is supplied by the application either in the form of a specific composition function, or by allowing the communication layer to use some default function, such as multiplication or addition.

2. *Adapting to parameter changes.* The second stage of payoff adaptation involves adapting to changes in both application requirements and network resources. As application requirements change, the application may provide the communication layer with new payoff curves for its data or it may simply change the characteristics of its communication. At this point, the communication layer must reevaluate the communication configuration and determine if the communication needs to be reconfigured. Changes in network resource parameters can either come from the network, in the form of a new loss load curve, or from observing the network. Again, the communication layer must adapt the communication to the changes. In either case, the communication layer repeats the process described for the first stage to determine the best fit for the application.

The benefit of using payoff-based adaptation comes from the depth of the information provided by the payoff functions to the communication layer. The communication layer is given a wide range of resource allocations that will satisfy the application and has the information necessary to determine what inside this range would best benefit the application. This freedom allows the communication layer to base adaptations on the requirements of the application. In comparison, adaptation schemes that use specification ranges or regions have the freedom to work within those areas, but have no guidance as to what would benefit the application. We can mimic such functionality by setting the service levels within the range or region to 1 and the payoff for the area outside to zero.

4 Evaluation

Our techniques are being evaluated in the context of a distributed virtual environment. The experiments discussed in this section address the reliability requirements of large data sets within such an application. Specifically, some of the graphical objects in the environment are “continuous objects” in the sense that their display requires continuous updates to images or texture maps or even motion JPEG. This section describes the benefits of payoff adaptation for such continuous objects.

Image transfer is the basis for many multimedia applications and is often tolerant to some loss. This tolerance enables the use of payoff-based communication adaptation. Turner and Peterson describe a retransmission-free protocol for image transfer [TP92]. They make the assumption that applications can give up some degree of image quality for a decrease in delay. The application is given the choice of getting all retransmissions or none. We believe that applications like ours can benefit from the ability to make more precise tradeoffs between delay and reliability.

The variable reliability framework discussed in this example is the basis on which we build to take advantage of application tolerance for loss. On this basis, each application may define “reliability” in terms that are most useful to it. [KCKS97] describes the design and implementation of the variable reliability protocol itself. For this paper, it is necessary to understand that this protocol delivers a service based on simple loss measurement and retransmission policies and that it provides applications with hard guarantees about reliability based on the loss allowances they specify. In this solution, the application can fine tune service to fit its requirements.

For the virtual environments application, our experimental evaluation uses two payoff functions. The first payoff function concerns quality, which is equated with the amount of the original data transmitted that is actually received. We call this *observed reliability* (OR) and we represent it with the payoff function P^{rel} . The observed reliability is determined by the function $OR = F_{or}(r_i, l_j)$, where r_i is the reliability level and l_j is the loss rate. The second payoff function concerns image transfer time and it is represented by P^{time} . The transfer time (TT) is determined by the equation $TT = F_{tt}(t_k, r_i, l_j)$, where t_k is the transfer rate. Note that since we are using loss load curves, the loss rate is implied from the specific transmission rate.

Total payoff is defined as $P_{total} = Q(P_{OR}^{rel}, P_{TT}^{time})$. The communication layer evaluates P_{total} over all possible reliability levels and transmission rates. From the values of r_i , l_j and t_k , the communication layer can determine OR and TT from the equations above. The results of these functions are used in turn to determine P_{total} for each reliability level and transmission rate pair. The function for determining total payoff for this experiment was chosen to be a multiplicative function of P^{rel} and P^{time} which implies that the total payoff is zero if either individual payoff is zero.

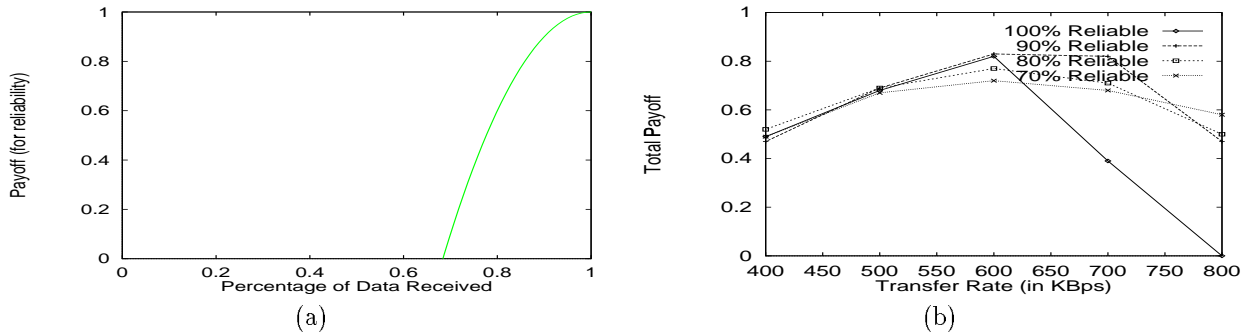


Figure 5: Payoff Functions for Reliability and Net Payoff Curves

Network resource information is represented in a fashion similar to the loss-load curves described in Section 3.4. By supplying the communication layer with a specific loss rate for each transfer rate, the communication layer can use this information to determine expected transfer time experienced at each rate. For

these experiments, the functions F_{OR} F_{TT} were determined from baseline timing runs. Data points were gathered for reliability levels of 70, 80, 90 and 100 percent; transfer rates of 400KBps, 500KBps, 600KBps, 700KBps and 800KBps. Each of these runs was subjected to loss probabilities varying from 0 to 50 percent in increments of 5. From loss load curves in the form $F_{LLCurve}(\text{TransferRate}) = \text{LossRate}$, the communication layer can determine the estimated transfer time for each transmission rate and reliability level. Next, transfer time and end-to-end loss are used as parameters to the application-supplied payoff functions P_{OR}^{rel} and P_{TT}^{time} . Finally, by evaluating the payoff functions to maximize P_{total} , the communication layer can determine the transfer rate/reliability pair that it considers the best fit for the application’s requirements. In this sections experiments, the payoff function for reliability is shown in Figure 5(a), while the payoff function for time is the one shown in Figure 3(b). Adaptation is performed in response to changes in network resource availability, as specified by changing loss-load curves. As the communication layer is informed about a change in the loss-load curve, it reevaluates the payoff functions to determine whether it should reconfigure the communication parameters.

For a specific example, consider Figure 4 which shows a loss-load curve with simultaneous increases in loss and transfer rates. The second graph depicts the payoff values obtained for this specific loss-load curve over varying transfer rates. The net payoff is calculated as the product of P^{rel} and P^{time} . Each line in Figure 5(b) represents the transfer of the image at the given reliability for the set of transfer rates. Using these curves, we can determine the optimal transfer rate and reliability for this specific loss-load curve and the requirements of the application by finding a maximum for P_{total} . In this example, the communication layer should choose to transfer data at a speed of 600KBps and a reliability level of 90%.

4.1 Experimental Setup

For our experiments, we use an application that transfers a 6.5Mbyte image 30 times across a 10Mb Ethernet connection, where the communication layer monitors the loss-load curves supplied by the network. The communication layer has the ability to change two parameters: reliability and transfer rate. For reliability, the communication layer can change the acceptable loss-burst size and allowable loss percentage in a window. In theory, the communication layer could allow a continuous choice for reliability. Our current implementation permits the choice between four distinct reliability levels: 100%, 90%, 80% and 70%. For the transfer rate, rates from 400KBps to 800KBps in increments of 100 are chosen.

The experimental results in this section demonstrate the effect of changes in loss-load curves over time, which model changes in network behavior (e.g., the network becoming more and less congested). The loss load curves used are based on the one in figure 4. This linear loss-load curve imposes a five percent increase in loss for each 100KBps increase in requested bandwidth. Changes in the loss-load curve are represented by shifting the entire curve up or down along the y-axis, thus imposing more or less loss depending the in which it is shifted. In this experiment, the loss-load curve is shifted every 15 seconds. The solid line in figure 6 represents the loss over time seen by an application that is transmitting at 800KBps.

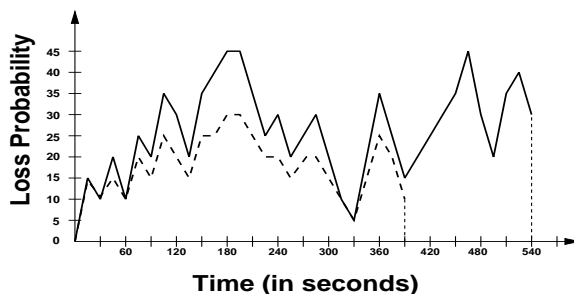


Figure 6: Observed Loss over Time

4.2 Results

The experiment compares the results of two runs of the application. In the first run, the communication is static. The application transmits at 800KBps and requires 100% reliability. In the second run, the communication is dynamically controlled through payoff-based adaptation. The communication layer monitors the loss-load curves and recalculates the communication parameters as the loss-load curves change. The dashed lines in figure 7 show the dynamic choices made by the communication layer as to reliability and offered load. Due to the choice of offered load, the loss rate experienced over time is represented by the dashed line in figure 6. These decisions were based on the payoff calculations made by the communication layer at each change in the loss load curves. The total payoff for both runs at each change is shown in figure 8. The solid line represents the total payoff for the first run and the dashed line represents the total payoff for the second run.

In figure 8, we can see that, given the ability to choose communications parameters, the application’s payoff can be maximized at discrete points in time. In addition, final payoff is shown in tables 1 and 2, which depict the total amount of data received and the total transfer times for both runs of the application. As expected, the first run receives the maximum payoff for the amount of data received. In comparison, the second run has relatively high payoff for the amount of data received. The biggest difference is in the payoff for the transmission time. The combination of these two payoff values gives us the total payoff for the transfer. The first run pays off at 0.8165, while the second run pays off at 0.9627. As expected, the application that can adapt to fluctuations in network resources is able to make intelligent decisions based on its willingness to trade off reliability for transfer time.

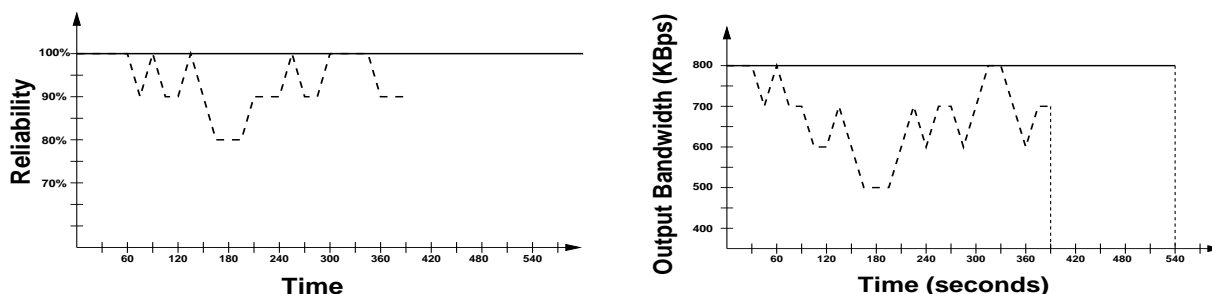


Figure 7: Application Reliability and Bandwidth over Time

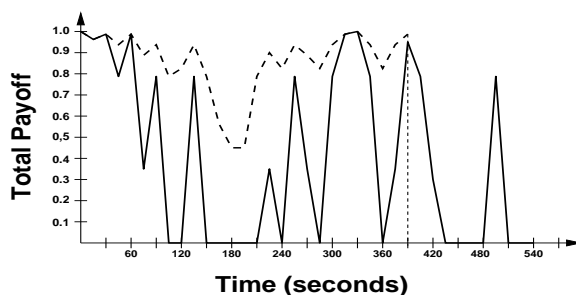


Figure 8: Total Payoff over Time

5 Conclusion

The key innovative contribution of our work is a cooperative solution to dynamic service management: the communication layer uses both application resource requirements and network resource availability to de-

	Data Received	Percent of Total Data Sent	Payoff for Data
Static	109350 msgs	100	1.000
Adaptive	104874 msgs	95.9	0.9832

Table 1: Results for amount of data received

	Transmission Time	Percent of Minimum Time	Payoff for Time
Static	537 sec	177	0.8165
Adaptive	381 sec	126	0.9784

Table 2: Results for transfer time

termine how to best configure the application’s communications. Specifically, we developed an adaptive communication layer that configures the operation of its communication protocols based on the perceived benefits to the applications using it. By permitting applications to share dynamic “service quality” requirements with the communication layer, in the form of *payoff functions*, the communication layer is able to make informed decisions about the communication configuration that best suits the current state of the application. Similarly, by dynamically monitoring resource availability from the network and then describing such availability as *service availability curves*, the communication layer is able to make suitable decisions about the communication configuration that best matches the current availability of network resources.

We have presented the results of experiments evaluating the effects of payoff-based adaptation in the context of a distributed virtual environments application. Specifically, we demonstrate that, given detailed information about tradeoffs available in network service, the value of the service provided to the application can be maximized. The focus of this paper has been on the continuous transfer of large data sets. We are currently running experiments addressing the transmission of small relatively small messages, such as position updates or tracker updates. Additionally, we are investigating the effects of managing multiple, diverse data streams in our virtual environment.

References

- [AMZ95] Elan Amir, Steven McCanne, and Hui Zhang. An application level video gateway. In *ACM Multimedia '95*. 255-265, 1995.
- [BHD⁺96] B.B. Bederson, J.D. Hollan, A. Druin, D. Rogers, J. Stewart, and D. Vick. A zooming web browser. In *Multimedia Computing and Networking (MMCN) '96*, 1996.
- [BMR97] N. Brownlee, C. Mills, and G. Ruth. Traffic flow measurement: Architecture. Technical Report RFC 2063, Internet Engineering Task Force, January 1997.
- [Bol] J-C. Bolot. Cost-quality tradeoffs in the internet. *Computer Networks and ISDN Systems*.
- [BS91] Thomas Bihari and Karsten Schwan. Dynamic adaptation of real-time software. *ACM Transactions on Computer Systems*, 9(2):143–174, May 1991.
- [CKV93] K. Curewitz, P. Krishnan, and J. S. Vitter. Practical prefetching via data compression. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 257–266, May 1993.
- [HW96] J. Huang and P.-J. Wan. On supporting mission-critical multimedia applications. In *IEEE Int. Conference on Multimedia Computing and Systems*, 1996.

- [JLT85] E. Douglas Jensen, C. Douglass Locke, and Hideyuki Tokuda. A time-driven scheduling model for real-time operating systems. In *IEEE Real-Time Systems Symposium*, pages 112–122, December 1985.
- [KCKS97] Robin Kravets, Ken Calvert, P. Krishnan, and Karsten Schwan. Adaptive variation of reliability. In *to appear in the Seventh IFIP Conference on High Performance Networking (HPN'97)*, April 1997.
- [KLP⁺95] S. Keshav, C. Lund, S. J. Phillips, N. Reingold, and H. Saran. An empirical evaluation of virtual circuit holding time policies in ip-over-atm networks. In *Proceedings of IEEE INFOCOM 95*, 1995.
- [KLV95] P. Krishnan, P. M. Long, and J. S. Vitter. Learning to make rent-to-buy decisions in probabilistic environments. In Armand Prieditis and Stuart Russell, editors, *Machine Learning: Proceedings of the Twelfth International Conference*. Morgan Kaufmann, 1995.
- [OSF⁺97] Seiwoong Oh, Hiroyuki Sugano, Kazutoshi Fujikawa, Toshio Matsuura, Shinji Shimojo, Masatoshi Arikawa, and Hideo Miyahara. A dynamic qos adaptation mechanism for networked virtual reality. In *Proceedings of Fifth IFIP International Workshop on Quality of Service*, May 1997.
- [TP92] Charles J. Turner and Larry L. Peterson. Image transfer: An end-to-end design. In *Proceedings of the SIGCOMM '92 Symposium*, Baltimore, Maryland, August 1992.
- [WC91] Carey L. Williamson and David R. Cheriton. Loss-load curves: Support for rate-based congestion control in high-speed datagram networks. In *Proceedings of the SIGCOMM '91 Symposium*, 1991.
- [Wil96] Carey Williamson. Dynamic bandwidth allocation using loss-load curves. *IEEE/ACM Transactions on Networking*, 4(6):829–839, December 1996.