An Overlay Network Transport Service for Teleoperation Systems

Zhiwei Cen, Matt W. Mutka and Danyu Zhu
Dept. of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824
Email: {cenzhiwe,mutka,zhudanyu}@cse.msu.edu

Ning Xi
Dept. of Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824
Email: xin@egr.msu.edu

***Abstract*— In real-time Internet based teleoperation systems, the Internet is the communication medium through which the operator sends control commands to the robot and receives feedback. Teleoperation systems support robotic control commands, video, audio, haptic feedback, and other sensory information, which are called supermedia. Traditional transport services of the Internet may not be able to meet the timely transmission requirements and dynamic priorities of supermedia streams. This paper aims to design an efficient and reliable transport service for teleoperation applications. Built upon multiple disjoint paths in overlay networks, Supermedia TRansport for teleoperations over Overlay Networks (STRON) uses forward error correction encodings to reduce end-to-end latency for the transmission of supermedia streams. Network routes and encoding redundancy may be adjusted dynamically to meet the QoS requirements of the supermedia streams in the face of networking performance degradation. At the cost of minimal encoding computation, the system achieves better performance in transporting supermedia streams than available transport services and at the same time remains friendly to other Internet traffic. Evaluations using PlanetLab available bandwidth traces show that STRON can significantly reduce latency.**

***Index Terms*— Teleoperation, Overlay networks, Forward error correction.**

I. Introduction

TELEOPERATION systems allow people to control automatic systems operating at remote sites where they are inaccessible by the operators. Traditional teleoperation systems use dedicated communication channels between the operator and the robot, which introduce higher costs and less flexibility. However, the growth of the Internet opens a new stage for teleoperation systems. The Internet not only enables the operator to control the robot at a remote site, but also transfers video, audio and haptic feedback information back to the operator. The operator can watch, hear, touch and feel the environment in which the robot is working. The video, audio and haptic feedback extends people's sensing ability from the approachable physical vicinity to the far reaching ends of the Internet.

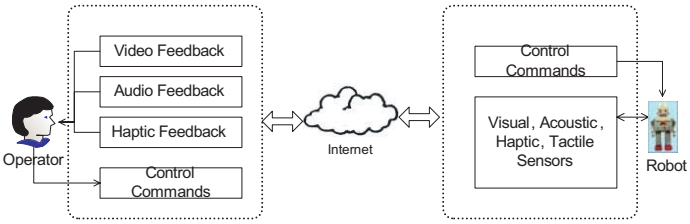


Fig. 1. An Internet Based Teleoperation System

Figure 1 shows the structure of a teleoperation system. The operator manipulates a control device to issue task commands to the robot. The feedback information, including video, audio and haptic information, enables the operator to be informed of the current state of the robot. The haptic or tactile feedback relates to the sense of touch. Haptic information is captured by robot sensors and reproduced at the operator side using force generating devices. Other sensing information such as heat, radiation and distance to obstacles may also be amalgamated into haptic feedback. With the control commands and the feedback information communicated through the Internet, the operator and the robot form the global control loop of a teleoperation system. The Internet serves as an action superhighway instead of an information superhighway as seen by traditional networking applications.

We call all the information flowing in a real-time teleoperation system supermedia [1]. Supermedia includes video, audio, haptic, temperature, control commands and other media. Supermedia differs from traditional multimedia in that a larger variety of media are involved, all of which are to be transmitted through a shared path in the Internet and each of them has a different quality of service requirement.

The biggest challenge of Internet based teleoperation is making the unpredictability caused by the Internet to be transparent to the control loop of the system. Random time delay, packet loss, network buffering effects and disconnects in the Internet's best-effort service model present major difficulties. Among all the uncertainties, time delay is one of the biggest obstacles to build a stable teleoperation control system. High latency may cause the robot to stall in the middle of a task or the operator to lose control of the remote robot. The uncertain variations of the network latency makes the control system unstable, which in turn affects of the performance of the teleoperation application. This problem may be approached through two directions:

1) Try to modify the control mechanism of the teleoperation system to accommodate the unpredictable nature of the Internet. The traditional control system assumes a dedicated and reliable communication channel between the operator and the robot. In order for the system to work over the Internet, some of its assumptions must be changed.
2) Try to improve the quality of service of the Internet to make it close to the dedicated communication channels used by the traditional system. One of the possible ways to help build a stable control system is to reduce the end-to-end latency of the communication channels.

Event based teleoperation systems [1] were presented to address the problem from the first direction. The performance difficulty caused by the Internet based teleoperation system is a result of using time as a reference for different system entities. In the event-based control approach, a non-time based reference is used. An event reference is a monotonically increasing parameter to synchronize the operator and the robot. Through the event reference, low level sensing and control modules are integrated with high level task scheduling and action planning. The system is able to cope with the uncertainties caused by the Internet and provide event transparency to the system user. Fung, et al. introduced a Task Dexterity Index (TDI [1]) of the robotic task associated with each supermedia stream. The dexterity of a task represents the complexity of the controlled movement that the robot makes and the degree of attention the operator should give to tasks of the robot. A Task Dexterity Index (TDI) is generated for each data stream using a fuzzy logic system to describe the bandwidth requirement of the robotic task. Later, this TDI may be used in the bandwidth allocation algorithm to rank the data streams that need to be transmitted.

Extensive research efforts have been dedicated to provide QoS aware transport service over the Internet. These efforts are closely related to our efforts to the second direction of approaching the teleoperation problem. Some of the most notable work, including Integrated Service (IntServ) and Differentiated Services (DiffServ), aims to change the best-effort service model to a QoS aware infrastructure. However, IntServ and DiffServ are not deployed over major networks due to a variety of reasons. Many research efforts aim to improve QoS levels for multimedia applications over the current best effort Internet [2], [3]. However these transmission mechanisms do not solve issues in supermedia transmission due to the following reasons:

1) A teleoperation system involves several kinds of media types, such as video, audio, control commands, haptic information, and code uploads. Different types of supermedia stream have different QoS requirements. Video and audio streams may be unreliable but need to have an adaptive real time transmission service. Control commands and haptic information need a timely and reliable transmission service. Code uploads need reliable transmission service but are not very time sensitive.
2) Supermedia streams have dynamic QoS requirements associated with the task execution process. The priority of a supermedia stream will change dynamically when the teleoperation system switches from one task to another. For example, video streams may have higher priorities when the robot is approaching an object or trying to circumvent obstacles, while haptic feedback may become the highest priority when the robot starts to manipulate the object.

The purpose of the Supermedia TRansport for teleoperations over Overlay Networks (STRON) approach is to improve the reliability and efficiency of teleoperation systems from the second perspective. STRON aims to provide a fast transport service to transmit latency sensitive supermedia streams over current non-QoS capable wide area networks. STRON takes advantage of multiple disjoint overlay paths and forward error correction encodings to improve the QoS performance. The networking routes and encoding redundancy may be adjusted dynamically to meet the QoS requirements of the supermedia streams in face of networking performance degradation. TCP Friendly Rate Control (TFRC [2]) is used as the congestion control mechanism for each overlay connection, which ensures that the supermedia traffic remains friendly to other Internet traffic.

The rest of this paper is organized as follows. Section II covers related work. In section III, the architecture of

STRON is presented with a detailed design of the optimal path selection module. We discuss the simulation methodology and results in section IV. Section V provides the conclusions.

II. Related Work

An overlay network is composed of a set of IP-layer network paths, which are called overlay links. The end hosts of the network paths are overlay nodes. Packets may be routed among overlay nodes according to overlay link performance measurements. Since the setup of an overlay network does not require changing the underlying network infrastructure, many overlay networks are used to deploy emerging networking applications. Other overlay networks are used to improve the performance of the Internet by circumventing the inefficiencies of BGP4 in the face of link and router failures. Common overlay networks include RON [4], QRON [5], SON [6], and PlanetLab. Some research focuses on implementing multicasting through overlay networks. Others focus on peer-to-peer file sharing systems.

A Resilient Overlay Network [4] allows distributed applications to detect and recover from link failures or performance degradation, which may be much quicker than solely relying on the routing protocols. The RON nodes monitor the Internet paths among them and decide whether to allow the Internet to route the packets or relay the packets among RON nodes to achieve a better performance. In order to meet the reliable and fast transmission requirement of teleoperation systems, the STRON approach may utilize several overlay network paths supported by the overlay nodes. The use of parallel connections to increase throughput was extensively studied in the high performance computing community. Some research shows that when the network is under utilized, parallel connections can achieve fair utilization among common connections [7]. Statistics shows in large academic networks such as Abilene [8], most network links are usually under utilized.

In order for the STRON system to provide an effective transport service, the overlay paths should be physically as disjoint as possible. Currently there are several research efforts related with disjoint paths selection in overlay networks. Cui uses a probability model to represent the disjoint degree of two overlay paths [9]. Any pair of links of the two paths is given a probability of breaking at the same time. The disjoint degree of the two paths is thus in reverse proportion to the sum of the breaking probability of the link pairs of the two path. Finding the minimum sum has been proved to be an NP complete problem. This paper presents an alternative approach to provide an approximation of the optimal case. In this approach, a shortest latency based path is selected as a primary path and another path is selected to be backup path based on its failure probability. The Control Overlay Protocol [10] divides the overlay nodes into regions. Each region has a super node. The super node probes its subordinate nodes to collect their disjoint information. Different super nodes coordinate to provide a set of approximately disjoint paths. A routing underlay approach for overlay networks [11] introduces a routing underlay that sits between the overlay network and the Internet. The routing underlay collects topological information from the Internet and this information is used in the overlay network to provide a more reliable and efficient service. This paper shows that by detouring into another AS, 1157 out of 1235 direct paths (93.7%) have at least one disjoint secondary path.

STRON uses forward error correction encoding to provide a reliable and fast transmission service, which is applicable for real time media. TCP provides the standard reliable transport service but it is not designed to serve the requirement of real time supermedia streams. UDP and its variant transport services, such as TCP Friendly Rate Control (TFRC [2]), Datagram Congestion Control Protocol (DCCP) and Stream Control Transmission Protocol (SCTP), are designed for multimedia transport but each has difficulty to provide reliable transport services by taking advantage of possible performance enhancements of overlay networks. SCTP also supports transport of the data over multiple paths but it requires the sender and receiver nodes to have multiple physical networking connections. RTP and RTCP provides support for real time application controls such as timing reconstruction, loss detection, security and content identification. However, they lack the congestion control and error recovery mechanisms that are essential for supermedia applications.

Other research work also uses overlay networks as a means to improve quality of service for multimedia applications. QRON (QoS-aware Routing in Overlay Networks [5]) is a general unified overlay network framework. One or more Overlay Brokers (OBs) in each Autonomous System (AS) form an Overlay Service Network (OSN). OSN provides services such as resource allocation and negotiation, overlay routing, and topology discovery. A hierarchical QoS-aware routing protocol is used to setup overlay paths between OBs that meet the QoS requirement of different applications. OverQoS [12] is an architecture to provide QoS using overlay networks. Flows are aggregated into a bundle in order for OverQoS to exercise control over the available resources among the flows. Controlled-Loss Virtual Link (CLVL) is an abstrac-

tion of the physical overlay links to characterize the service received by a bundle. By policing at the entry and exit nodes of CLVL, a bound of the loss rate can be provided to the overlay applications. OverQoS does not provide a mechanism to reduce end-to-end latency, which is essential in supermedia applications. The method presented by Nguyen et al [13] uses a $traceroute$ based heuristic scheme to find redundant paths and transmit FEC encoded data over multiple paths to reduce packet loss. However, the $traceroute$ approach is traffic intensive and the heuristic scheme may not always be able to find the optimal path set. Andersen, et al [14] examined the performance of transporting data over multiple overlay network paths. The measurement results showed that "the chances of losing two packets between the same hosts is nearly as high when those packets are sent through an intermediate node (60%) as when they are sent back-to-back on the same path (70%)". STRON may also use multiple overlay paths. However the FEC encoding and TCP friendly rate control of STRON make sure the latency between the end hosts is reduced even if the overlay network paths experience high packet loss rates.

III. System Architecture

A. System Overview

The basic idea of STRON may be explained as follows. Each supermedia stream contains a series of messages generated by the teleoperation application in a variable or constant bit rate. The message is again chopped into packets with a certain size (such as the Maximum Transfer Unit or MTU) determined by the networking layer. Given p packets for a certain message that needs to be transmitted, STRON encodes the p packets into αp packets, where α $(\alpha \geq 1)$ is called the stretch factor. The encoded data packets are scheduled to be transmitted over multiple overlay paths. As soon as the receiver collects $(1+\epsilon)p$ distinctive encoded data packets, the decoding algorithm can reconstruct the original data packets. Here ϵ is called the reception overhead. The reception overhead is zero for some encoding algorithms and a small number for others. A class of erasure codes that has this property is called a digital fountain code [15]. By using a digital fountain code, the transport protocol can provide a rather reliable transport service without using acknowledgments and retransmissions. The most common digital fountain codes are Reed-Solomon codes and Tornado codes [15]. Simulation using Reed-Solomon codes shows the encoding and decoding time for 100KB data (100 1KB-sized packets) are around 0.12 seconds and 0.054 seconds (the simulation is done on an AMD Duron 1.26GHz CPU with 512MB memory). However, the Tornado codes can decrease the encoding and decoding time by over 100 times [15]. For Reed-Solomon codes, the reception overhead is zero. For Tornado codes, the reception overhead is around 0.05. The advantage of the Tornado codes is that they trade off a small increase in the reception overhead for a big improvement of the encoding and decoding efficiency.

Figure 2 illustrates the mechanisms of transporting supermedia streams over multiple paths. Source data were encoded with stretch factor 2 and sent over two disjoint paths. Six packets were sent over path 1, which has a higher bandwidth and lower loss rate. Four packets were sent over path 2, whose bandwidth is low and the loss rate is high. As soon as the receiver collects 5 data packets, which are enough for the decoder to work, it signals the sender to stop sending packets and decodes the information. Because STRON eliminates the requirement of most retransmissions when packet loss occurs, it has great advantage over traditional transport protocols.

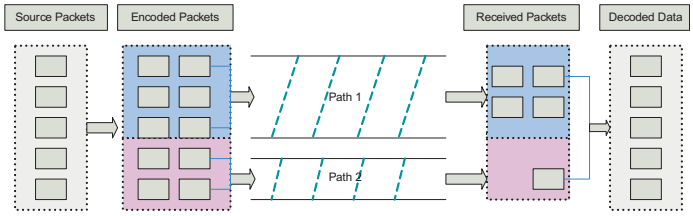


Fig. 2. An Example of Transporting Messages over Multiple Paths

The system architecture is shown in Figure 3 and Figure 4. At the sender side, the supermedia streams are classified according to their roles in the teleoperation system by the Classifying System (CS). Each class of stream has its own QoS requirement. For teleoperation applications, the supermedia streams have three classes. Class 1 is for small but high frequency data streams, such as temperature and force feedback. These data streams have a low bit rate but whenever a data packet is available it requires timely delivery. We may not apply redundancy encoding to supermedia of this class and simply transfer copies of the data over several overlay network paths. Class 2 also works for low bit rate and high-frequency supermedia streams, such as the force feedback when multiple force detectors are present. For class 2 supermedia streams, the data generated during each sample interval cannot fit into one packet. Hence redundancy encoding is needed to provided a reliable transmission. However, since the amount of data being sent is not large, a high redundancy level may be used to ensure reliability and timeliness. Class 3 is designed to transmit high bit rate supermedia streams that do not have a strict requirement on reliability. Ex-

amples of class 3 supermedia streams include video and audio. A certain level of redundancy is used to transmit class 3 data over multiple overlay paths. We may also differentiate video and audio streams since sometimes audio streams are more sensitive to data loss than video streams. Differentiated treatments of each supermedia class are implemented in the γ updating algorithm of the Optimal Path Selection Module (OPSM).

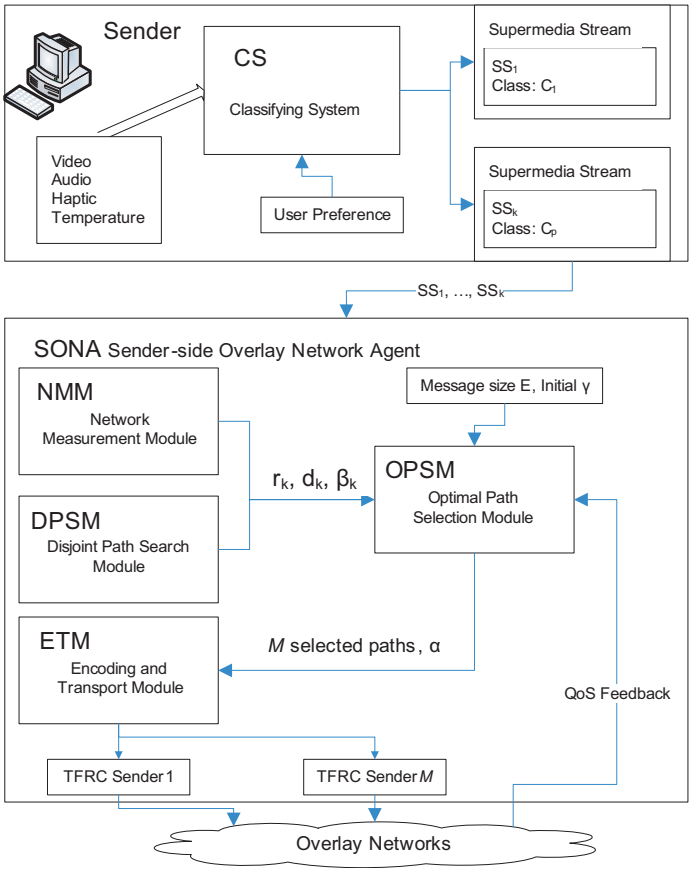


Fig. 3. The Architecture of the Sender Side

The Sender-side Overlay Network Agent (SONA) is responsible for transporting each class of streams according to its QoS specifications. The Disjoint Path Search Module (DPSM) runs a disjoint overlay network path searching algorithm [9], [10], [11] through the connected overlay nodes. The paths found by DPSM may not be totally physically disjoint. Such paths may be called semi-disjoint paths. A disjoint degree shows the correlation of the network performance fluctuation of a pair of overlay paths. A higher disjoint degree may enhance the reliable transmission service of the system. The Network Measurement Module (NMM) is usually deployed in overlay nodes as a common service. Active measurements are launched periodically to probe the networking conditions. Some research efforts have been dedicated to making accurate measurements of the network. *Pathload* [16] takes

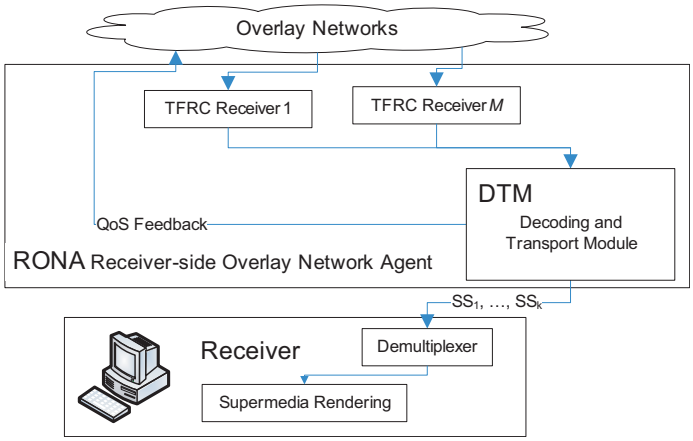


Fig. 4. The Architecture of the Receiver Side

advantage of the fact that the one-way delays of a periodic packet stream show an increasing trend when the stream rate is larger than the available bandwidth. The measurement results (available bandwidth r_i, single trip delay d_i and packet loss rate β_i) of the disjoint path set are fed into the Optimal Path Selection Module (OPSM).

The objectives of OPSM include: (1) find the optimal disjoint path set to be used as active transmission paths; (2) decide the amount of data sent over each active path; (3) decide the stretch factor according to the networking condition of the chosen active paths. In the case when the available disjoint paths set are not too large (which is true in common cases), a brute force method may be used to find an optimal subset of disjoint paths to minimize the overall transmission delay. The amount of data sent over each active path is determined according to the capacity of each path. The redundancy coefficient γ is a safety factor to make sure enough packets are received by the receiver under the presence of unpredictable loss patterns. The loss rate of each active path and γ jointly decides the stretch factor α. OPSM also receives QoS feedback from the receiver. The feedback tells the sender to what degree the redundancy in the encoding is effective or whether retransmission is needed for reliable transport services. Hence OPSM uses the feedback to decrease or increase the redundancy by calculating a new stretch factor α. The design of the OPSM is discussed in detail in the latter parts of this section.

The Encoding and Transport Module (ETM) encodes the supermedia streams using a stretch factor α and passes the encoded packets to the overlay transport service. The stretch factor α is found by the Optimal Path Selection Module. The encoding algorithm may be any of the digital fountain codes, such as Reed-Solomon codes [17] and Tornado codes [15]. TCP Friendly Rate Control (TFRC [2]) is used as the transport control mechanisms

for each overlay path. TFRC provides a congestion control mechanism that is more suitable to multimedia applications than TCP while at the same time remains TCP friendly.

At the receiver side, the Decoding and Transport Module (DTM) of the Receiver-side Overlay Network Agent (RONA) decodes the packet streams received by the TFRC receivers and notifies the sender when enough packets have been collected for a certain message or a message is lost if a timeout occurs. The feedback to the sender also includes information telling the sender to what degree the redundancy in the encoding is effective so that the sender can adjust the redundancy level accordingly.

In the following section we discuss the design details of the Optimal Path Selection Module and the transport protocol.

B. Optimal Path Selection Module (OPSM)

The OPSM receives input from NMM and DPSM specifying the available disjoint overlay paths and the quality parameters of each overlay path. The goals of the OPSM is to find the optimal disjoint path set to be used as active transmission paths, and decide the amount of data sent over each active path and the stretch factor for the digital fountain encoding. The OPSM consists of two parts: the path selection submodule and the γ adaption submodule.

1) The Path Selection Submodule: We formulate the problem of optimal path selection as follows. Suppose N disjoint (or semi-disjoint) overlay paths were found in DPSM. For each path k ($k \in [1, N]$), the following parameters are given: the single trip delay d_k in terms of seconds, the average throughput r_k in terms of bytes per second, and the packet loss rate β_k. Each supermedia stream contains a series of messages generated by the teleoperation application as a variable or constant bit rate. The size of the message, which is the amount of effective data we need to transmit from the sender to the receiver, is E in terms of bytes.

We need to choose M disjoint paths out of the N given paths to minimize the latency between the sender and receiver. For each path i ($i \in [1, M]$) of the M path, we also need to calculate V_i, which is the amount of data injected into this path by the sender. The system redundancy coefficient γ shows how much redundancy the system has over unexpected packet loss. Assume E_i is the effective data gathered from path i that is used in the data construction process, and we have

$$E_i = \frac{V_i(1-\beta_i)}{\gamma}. \tag{1}$$

Assuming t is the time for the receiver to receive and reconstruct the original data, we have

$$t = \frac{V_i}{r_i} + d_i \tag{2}$$

and

$$V_i = (t - d_i)r_i \tag{3}$$

According to (1)

$$E_i = \frac{r_i(t-d_i)(1-\beta_i)}{\gamma} \tag{4}$$

We have

$$\sum_{i=1}^{M} E_i = E(1+\epsilon) \tag{5}$$

where ϵ ($\epsilon \in [0, 1)$) is the reception overhead of the digital fountain code, or

$$E = \frac{\sum_{i=1}^{M}[(t-d_i)r_i(1-\beta_i)]}{(1+\epsilon)\gamma} \tag{6}$$

Solving t yields

$$t = \frac{E(1+\epsilon)\gamma + \sum_{i=1}^{M} r_i(1-\beta_i)d_i}{\sum_{i=1}^{M} r_i(1-\beta_i)} \tag{7}$$

To solve the problem, we may try set $s = \{(i_1, i_2, \cdots, i_M) | i_p \in [1, N], p \in [1, M],$ and for any $p \neq q, p \in [1, M], q \in [1, M], i_p \neq i_q\}$, which is a combination of set $[1, N]$ to minimize (7), which is

$$t = \frac{E(1+\epsilon)\gamma + \sum_{j\in s} r_j(1-\beta_j)d_j}{\sum_{j\in s} r_j(1-\beta_j)} \tag{8}$$

The most straightforward method is to enumerate all the $\binom{N}{M}$ sets to find the subset s of $[1, N]$ that minimizes (7). When N is small, which is true in most cases, this method works well. When N is large, more efficient algorithms are needed.

The volume that needs to be sent over a selected disjoint path i, which is V_i, may be found by using (3). The stretch factor α of the digital fountain encoding may be calculated as

$$\alpha = \frac{\sum_{i=1}^{M} V_i}{E} \tag{9}$$

A *transport plan* consists of a series of active overlay network paths, the QoS parameters of these paths, the redundancy factor γ, the stretch factor α and the number of packets (or the volume of data) to be sent over each active overlay path.

We have the following example to illustrate how the path selection process works. Assume the message size is

TABLE I
TWO DISJOINT PATHS

Path	r_k (KB/sec)	d_k (sec)	β_k
1	600	0.1	0.01
2	1000	0.2	0.005

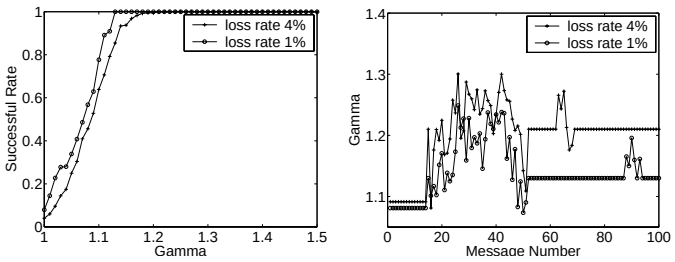
100 packets and the packet size is $1K$ bytes. Thus $E = 100KB$. We assume the redundancy coefficient γ is 2 and the reception overhead ϵ is 0.05. Assuming we have two disjoint paths to choose (shown in Table I), we have the following possible strategies.

1) Choosing path 1 only, we have $t = \frac{E(1+\epsilon)\gamma + r_1(1-\beta_1)d_1}{r_1(1-\beta_1)} = 0.4535$.
2) Choosing path 2 only, we have $t = \frac{E(1+\epsilon)\gamma + r_2(1-\beta_2)d_2}{r_2(1-\beta_2)} = 0.4111$.
3) Choosing both paths, we have $t = \frac{E(1+\epsilon)\gamma + r_1(1-\beta_1)d_1 + r_2(1-\beta_2)d_2}{r_1(1-\beta_1)+r_2(1-\beta_2)} = 0.2948$.

It is clear that strategy 3 is the best one. Given this strategy, we can calculate the V_i for each path according to (3): $V_1 = (t-d_1)r_1 \approx 117KB$ and $V_2 = (t-d_2)r_2 \approx 95KB$. The stretch factor α is $\frac{V_1+V_2}{E} = 2.1164$.

2) The γ Adaption Submodule: The redundancy coefficient γ is a user defined parameter specifying the amount of redundancy to be used in the system. The value of γ influences the efficiency and reliability of the system and should be adjusted dynamically when the quality of the overlay paths changes. The larger γ is, the more reliable the transport service. However, a larger γ requires more encoding and decoding overhead. Figure 5(a) shows the relationship of γ and the successful rate.

In the simulation scenario, two candidate disjoint overlay network paths were established (detailed description of the simulation methodology may be found in Section IV). The slower path $path0$ has an available bandwidth of 1MB and a single trip delay of 100ms. The faster path $path1$ has an available bandwidth of 10MB and a single trip delay of 100ms. In both cases, 100 supermedia messages were sent. Each message contains 50 packets whose size are 1000 bytes. The MMPP model was used to simulate the packet loss events (see Section IV-B for details). A message is said to be transmitted successfully when the receiver is able to reconstruct the original message using the packets received over different overlay paths. In the first case, both overlay paths were using an MMPP error model with an expected packet loss rate of 4%. We see that $\gamma = 1.2$ ensures the transport to be 100% reliable under this network condition. In the second case, both overlay paths used an MMPP error model with an expected packet loss rate of 1%. A smaller γ was able to ensure that the transport is 100% reliable. Considering the difficulty of specifying the redundancy coefficient by the user and the fact that the quality of the network is always changing, we need an automatic γ updating algorithm. The γ adaption submodule updates the redundancy coefficient according to the quality of the overlay paths. Figure 6 shows the gamma adaptation algorithm. The variables `actual_succ_rate` and uf_rate are feedbacks received from the receiver by the sender. The target successful rate of this supermedia class is denoted as `target_succ_rate`. The variables `gamma_roof` and `gamma_floor` are the minimal and maximal value of γ for this supermedia class. The variables `diff_allowable`, `uf_floor` and `uf_roof` are other QoS parameters for this supermedia class and will be explained in the following paragraphs.



(a) Effects of Gamma over Successful Rate

(b) Automatic Gamma Adaptation

Fig. 5. Simulation of the Influence of γ

The variable `actual_succ_rate` is the ratio of the messages successfully delivered by the system up to the present. A message is regarded as delivered successfully if enough packets are received by the receiver before a timeout occurs and the original message is successful decoded by the decoder. The variable uf_rate means "under-full rate." The under-full rate is calculated by the receiver after a message is successfully decoded, otherwise the under-full rate is set to -1. The definition for uf_rate is

$$uf\_rate = \frac{(AllPkts - PktsSent) \cdot \frac{PktsReceived}{PktsSent}}{AllPkts} \tag{10}$$

where $AllPkts$ is the total number of packets after the message is encoded and may be calculated as

$$AllPkts = \lceil \alpha \cdot EffectivePkts \rceil. \tag{11}$$

α is the stretch factor and $EffectivePkts$ is the original data packets of the message.

Upon decoding, $PktsReceived$ should equal $EffectivePkts$. $PktsSent$ is the number of pack-

ets sent by the sender up to now. The receiver obtains this information from the packet ID of the last packet received. Although there may be some packets in transit or lost after the last packet was sent, those packets are negligible.

The variable uf_rate shows the percentage of the packets that would have been useless if they were received successfully by the receiver. Thus, uf_rate is a parameter that shows how much redundancy was "wasted" during previous transmissions. To keep the system stable, uf_rate is updated in the following way:

$$uf\_rate_{new} = uf\_rate_{old} \cdot \alpha_{uf} + uf\_rate_{last\_time} \cdot (1 - \alpha_{uf}) \quad (12)$$

where α_{uf} is a stabling parameter.

The γ adaptation algorithm works as the following. If the actual successful rate of the supermedia stream is below the target successful rate by a difference more than `diff_allowable` or uf_rate is below the floor threshold and γ has not reached the roof value, γ is increased. Otherwise if the under-full rate is over the roof value (`uf_roof`) and γ is over the floor value, γ is decreased.

```
bool gamma_changed = false;
if ( (uf_rate < uf_floor ||
      target_succ_rate - actual_succ_rate
   > diff_allowable) &&
   gamma < gamma_roof) {
   increase gamma;
   gamma_changed = true;
}
else if ( uf_rate > uf_roof &&
     gamma > gamma_floor) {
   decrease gamma;
   gamma_changed = true;
}
if( gamma_changed)
   run path selection algorithm;
```

Fig. 6. γ Adaptation Algorithm

The system adjusts the quality of service for different classes of supermedia by setting appropriate γ adaptation parameters for them. The default values of the γ adaptation parameters used in the simulation system are shown in TABLE II. β_{avg} is the average loss rate of the active overlay paths. The parameters for class 1 supermedia are trivial since class 1 supermedia may not need redundancy encoding but simply send duplicate data packets over different paths. If the quality of the overlay paths is stable, the value of γ will converge and the system reaches a stable state. Figure 5(b) shows the working process of the automatic γ adaptation system. The simulation scenario is the same as the one described at the beginning of this section. In both cases, 100 class 2 supermedia messages were sent. After a certain number of transmissions, γ tends to converge to a stable value.

TABLE II
THE γ ADAPTATION PARAMETERS

Class	2	3
Initial γ	$\beta_{avg} + 2$	$\beta_{avg} + 1$
`target_succ_rate`	100%	98%
`gamma_roof`	$\beta_{avg} + 4$	$\beta_{avg} + 3$
`diff_allowable`	0.000001	0.001
`uf_roof`	0.16	0.12
`uf_floor`	0.12	0.10

C. The Transport Protocol Design

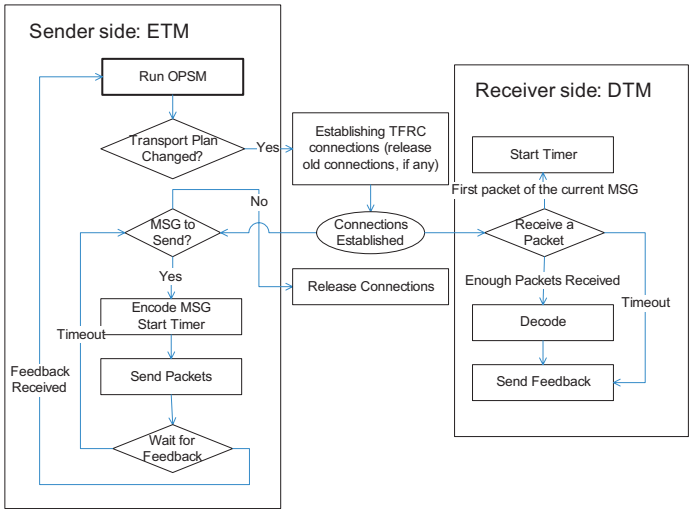


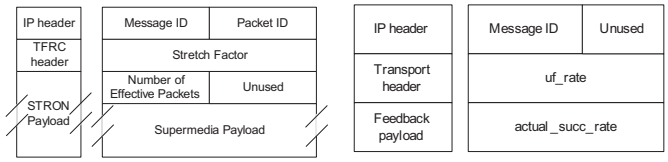
Fig. 7. The Transport Protocol State Transitions

The state transitions of the transport protocol are shown in Figure 7. The two major transport entities are the Encoding and Transport Module (ETM) at the sender side and the Decoding and Transport Module (DTM) at the receiver side. The ETM calls OPSM when the system initializes and when a feedback is received. OPSM in turn gets inputs from NMM and DPSM, which provide the overlay network information (as seen in Figure 3). OPSM produces a transport plan using the overlay network information and the feedback. According to this transport plan, ETM initializes the TFRC connections and encodes the messages to be sent. Before sending the encoded packets, ETM starts a timer. If the timer timeouts before feedback is received, ETM stops waiting and tries to send a new message using the old transport plan. The value of the timer may be specified in the form of δt, where t is the speculated transmission time found in (8) and δ is an adjustable parameter. For certain classes of supermedia

streams, a reliable transport service may be needed. In such cases, ETM will repeat transmitting the same message until a positive feedback is received.

The receiver side transport entity DTM starts a timer when the first packet of a new message is received. After enough packets were received for the current message (the number of effective packets required to decode the message is specified in the data packet, see Figure 8(a)), ETM decodes the message and sends a feedback to the sender. If the timer timeouts before enough packets were received, a negative feedback (with the under-full rate set to -1) is sent back to the sender. Similar to the sender side timer, the value of the receiver timer may also be specified in the form of δt, where t is the speculated transmission time found in (8) and δ is an adjustable parameter. However, the sender side feedback timer should be a bit longer than the receiver timer considering the single trip delay of the feedback packet.

Since the feedback only contains a small amount of data, no redundancy encoding is needed. The feedback may be sent over the primary overlay path (the overlay path with the best quality) through the TFRC connection of that path. When none of the overlay paths has satisfactory quality, the feedback may also be sent with duplicate copies over several overlay paths. Besides the existing TFRC connection, an additional TCP connection can be setup to transmit the feedback data. The



(a) The Data Packet Format of STRON

(b) The Feedback Packet Format of STRON

Fig. 8. STRON Packet Formats

packet formats of STRON data and feedback are shown in Figure 8(a) and Figure 8(b). Stretch factor, uf_rate and `actual_succ_rate` are double float numbers. The sender and receiver should agree on a format of these numbers.

IV. Simulation

To evaluate the effectiveness and efficiency of the system, the ns2 network simulator was used to construct a simulation environment for the overlay network transport protocol. An Optimal Path Selection Module was implemented in the application layer of the simulator. An implementation of Reed-Solomon codes [18] was used to estimate the encoding and decoding overhead of the algorithm. We did not use Tornado codes in the simulation due to its proprietary nature.

A. Simulation Methodology

The OPSM was implemented in ns2 as an Application layer protocol. According to the overlay network information and the supermedia transport scenarios provided by the simulator input, the OPSM constructs a transportation plan. The TFRC module in ns2 was modified to support a supermedia payload. One unreliable TFRC connection is established over each chosen overlay path in the supermedia transport plan. The receiver side of the application collects successfully transmitted packets and carries out a simulated decoding process. The application layer of the simulator consists of the following interfaces:

- Adding paths: Add an overlay network path to the system. For each path, the system is provided with the available bandwidth r_i, single trip delay d_i and packet loss rate β_i.
- Removing paths: Remove a previously added overlay network path.
- Modifying paths: Modify the parameters of a previously added path.

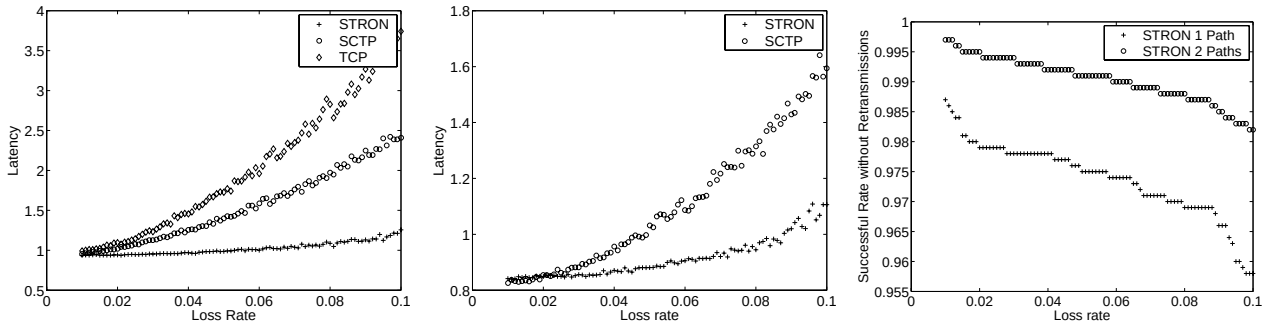
After collecting simulation setup information from the user, the system application layer runs the path selection submodule using the initial γ adaptation submodule parameters to choose a set of paths as an active overlay path. The simulator scheduler starts sending supermedia messages at a specified time. For each message received, the receiver generates the QoS feedback needed for the γ updating algorithm. When a QoS feedback is received by the sender, the sender runs the gamma adaption submodule and path selection submodule, which yield a new transport plan.

B. Packet Loss Models

In order to simulate the behavior of the system under lossy network conditions, it is important to choose good packet loss models. Extensive research [19] has been done on simulating packet loss of the Internet.

Trivial stochastic packet loss models drop a subset of packets according to a probabilistic process, such as uniform or exponential processes. A Poisson arrival process assumes the packet loss event to be independent and the loss event inter-arrival time conforms to an exponential distribution. Some measurements [20] show that the packet loss events of the Internet may be independent and fit an exponential distribution.

A more sophisticated error model uses a two-state continuous time Markov chain. The model has two states that



(a) Latency vs Loss Rate: STRON, TCP and SCTP (with 1 path)

(b) Latency vs Loss Rate: STRON and SCTP (with 2 Paths)

(c) Successful Rate without Retransmissions

Fig. 9. Simulation with Variable Packet Loss Rates

have different packet dropping rates. One common configuration is to set one state to have a zero dropping rate, corresponding to a stable period of the network, and the other state have a certain dropping rate, corresponding to the lossy period of the network. The state sojourn time conforms to an exponential distribution. Although the Markov model retains memory of previous states, some work [21] found that the Markov model tends to underestimate the probability of consecutive packet losses.

To depict packet loss events more accurately, the Poisson process is modified into a Markov Modulated Poisson Process (MMPP). The MMPP is a doubly stochastic Poisson process whose arrival rate is given by an m-state irreducible continuous time Markov chain. The Markov chain should be independent of the Poisson process. The MMPP can be implemented in the ns2 simulator using the MultiState loss module.

C. Simulation with Variable Packet Loss Rates

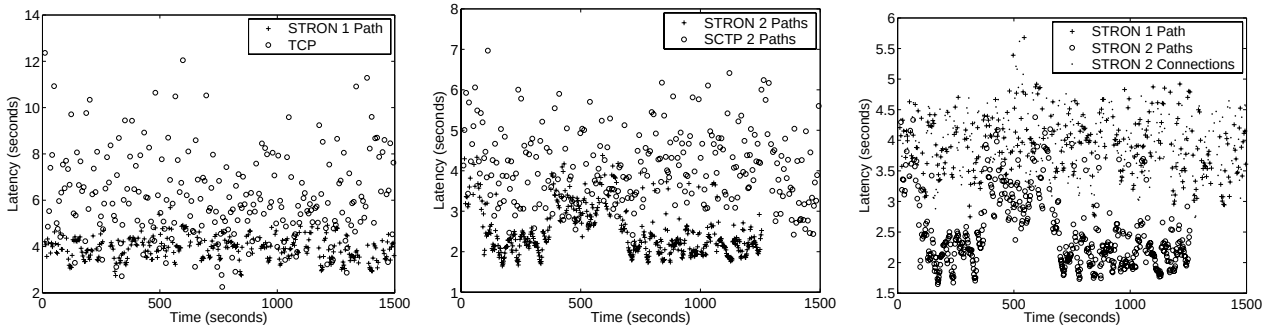
In the following simulations, two candidate disjoint overlay network paths were established. The slower path $path0$ has an available bandwidth of 1MB and a single trip delay of 100ms. The faster path $path1$ has an available bandwidth of 10MB and a single trip delay of 100ms. We simulated the transmission of class 2 supermedia. Each message contains 50 packets, each of which is 1000 bytes. In the following two subsections, we compared the performance of STRON with TCP and SCTP under variable packet loss rates. Since TCP does not support multiple transmission paths, only $path1$ is used in the simulation when TCP is involved in the comparison.

Unless explicitly specified, we used the exponential packet loss model in this simulation since the exponential model is more suitable to generate packet loss events according to a given packet loss rate. In the following simulation, the packet loss rate was increased from 0.01 to 0.1. We see from Figure 9(a) that while the packet loss rate increases, STRON performs better than TCP and SCTP. For example, when the loss rate is 0.05, STRON takes 0.98 seconds to transmit a 50KB message, while TCP takes 1.73 seconds and SCTP takes 1.40 seconds. An experiment with the Reed-Solomon codes shows with stretch factor 1.2, the encoding and decoding of 50KB data takes 0.014 and 0.002 seconds under the experimenting computer (an AMD Duron 1.26GHz CPU with 512MB memory). Thus overall STRON takes 0.996 seconds, with an improvement of 42% compared with TCP and 29% compared with SCTP. If Tornado codes were used as the digital fountain encoding method, the improvement would be even higher.

Unlike TCP, SCTP is able to support multiple transmission paths. In Figure 9(b) two paths were used for the transmission. The path $path0$ has an MMPP packet loss model with expected packet loss rate 0.01. The path $path2$ has an exponential packet loss model with the packet loss rate increasing from 0.01 to 0.1. Although when the packet loss rate is low, SCTP performs a little better than STRON, STRON may take better advantage of the overlay paths as the loss rate increases. For example, when the loss rate is 0.05, STRON takes 0.88 seconds to transmit a 50KB message, while SCTP takes 1.03 seconds. Using the same encoding and decoding time in the previous simulation, we have a 13% improvement with STRON over SCTP. The successful rates without retransmissions of STRON are shown in 9(c). It shows an additional overlay path helps improve the successful rate dramatically.

D. Simulation with PlanetLab Traces

In this part, we simulated the behavior of STRON using available bandwidth traces collected under Planet-



(a) Comparison of STRON and TCP Using BW 1

(b) Comparison of STRON and SCTP Using BW 1 and BW 2

(c) STRON under Different Conditions

Fig. 10. Simulation Using PlanetLab Available Bandwidth Traces

Lab. The available bandwidth traces were measured using *Pathload* [16] among three PlanetLab nodes that are located in Michigan State University (node MSU), Japan (node JPN) and Australia (node AUS), respectively, as shown in Figure 11. The measurements were done among

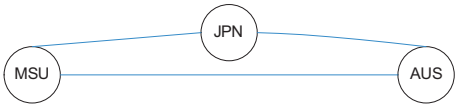
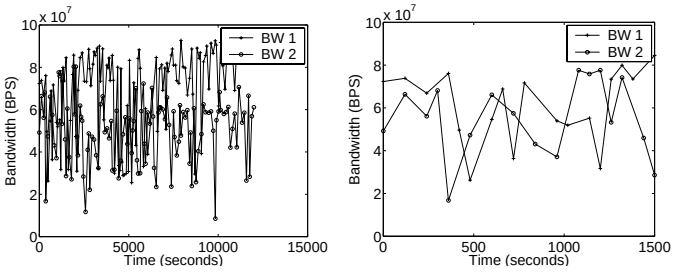


Fig. 11. Three PlanetLab Nodes



(a) The Available Bandwidth Series

(b) The Available Bandwidth Series Used in the Simulation

Fig. 12. PlanetLab Available Bandwidth Series

the three nodes at the same time. One *Pathload* probe started at the beginning of every minute. Most measurements finished within one minute. If *Pathload* does not converge within one minute, the launching of the next measurement was delayed until the beginning of the minute that goes right after the previous *Pathload* measurement finishes. The measurement results generated two bandwidth series as shown in Figure 12(a). $BW1$ was the available bandwidth between node MSU and AUS. $BW2$ was the minimum of the available bandwidth of the path between MSU and JPN, and the path between JPN and AUS. $BW2$ can be regarded as the available bandwidth series of an overlay path between MSU and AUS that traverses JPN. Results of $traceroute$ show that the paths of MSU-AUS and MSU-JPN-AUS are quite disjoint except for some overlaps on the continents of North America and Australia. A snapshot of the available bandwidth measurement (bandwidth series from 1 to 1500 second in Figure 12(b)) was used in the simulation. We modified the simplex-link object in ns2 to enable the bandwidth input. In the following scenarios, a series of supermedia messages were sent from node MSU to AUS. Each message has 250 original packets and the packet size is 1000 bytes. An MMPP packet loss model with expected packet loss rate 1% was assumed on all paths. Figure 10(a) shows the latency variance when the messages were sent over the direct connection using the available bandwidth series $BW1$. STRON achieves better performance and has less variance compared with TCP. Figure 10(b) shows the situation when both the direct path and overlay path are used by STRON and SCTP. In the simulation of Figure 10(c), we established two TFRC connections under the direct path (with available bandwidth $BW1$). The results show that two TFRC connections under one path do not necessarily decrease the latency compared with one TFRC connection under one path. However, a second overlay path can improve the performance of the system under the same packet loss conditions. The average and standard deviation of the latency under different schemes are shown in TABLE III. Considering the encoding and decoding time of STRON (which are 0.071 seconds and 0.014 seconds for 250KB data on the experimental computer, see Section IV-C), in the case of one path, STRON has an improvement of 34% over TCP. Using the same encoding and decoding overhead, in the case of two paths, STRON

TABLE III
THE AVERAGE AND STANDARD DEVIATION OF THE LATENCIES

Simulation	Average	STD
STRON 1 path	3.9268	0.4947
STRON 2 paths	2.4604	0.5886
TCP 1 path	6.0985	1.8439
STRON 2 connections	3.9022	0.4899
SCTP 2 paths	4.0613	0.9387

has an improvement of 37% over SCTP. STRON has reduced the latency by 37% by using an additional transport path. STRON with 1 path and STRON with 2 paths have an average successful rate without retransmissions 98% and 99%, respectively.

V. CONCLUSIONS

Designing a transport service for an Internet based teleoperation system is challenging because of diverse media forms in the application and the timely transmission requirement of supermedia streams. The end-to-end latency of the communication channel is crucial for the teleoperator to send commands to the robot and receive haptic feedback from the robot, both of which are important for the operator to maintain smooth control of the robot. The STRON system is built to reduce the end-to-end latency and improve other QoS parameters for teleoperation systems.

Disjoint overlay network paths and forward error correction encoding are used to improve the reliability and efficiency. An automatic redundancy factor adjustment system can update the redundancy level used to ensure the right amount of redundancy is used to make the transmission reliable. An optimal path selection module can adjust the transmission paths according to the dynamic network conditions.

The results of the simulation demonstrates that the supermedia transport system performs well under different network conditions. In the case of network paths with heavy packet loss rates, it has a significant improvement over TCP and SCTP.

REFERENCES

[1] Wai Keung Fung, Ning Xi, Wang-tai Lo, BooHeon Song, Yu Sun, and Yun hui Liu, "Task Driven Dynamic QoS based Bandwidth Allocation for Real-time Teleoperation via the Internet," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[2] Mark Handley, Sally Floyd, Jitendra Padhye, and Joerg Widmer, "TCP Friendly Rate Control (TFRC): Protocol Specification," *RFC 3448, Proposed Standard*, 2003.

[3] Hugh M. Smith, Matt W. Mutka, and Eric Torng, "Bandwidth Allocation for Layered Multicasted Video," in *ICMCS*, 1999.

[4] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris, "Resilient Overlay Networks," in *18th ACM Symposium on Operating Systems Principles*, 2001.

[5] Z. Li and P. Mohapatra, "QRON: QoS-aware Routing in Overlay Networks," *To appear in the special issue on Service Overlay Networks in the IEEE Journal on Selected Areas in Communications*, 2004.

[6] Zhenhai Duan, Zhi-Li Zhang, and Yiwei Thomas Hou, "Service Overlay Networks: SLAs, QoS and Bandwidth Provisioning," *10th IEEE International Conference on Network Protocols (ICNP '02)*, 2002.

[7] Thomas J. Hacker, Brian D. Noble, and Brian D. Athey, "The Effects of Systemic Packet Loss on Aggregate TCP Flows," *Conference on High Performance Networking and Computing*, 2002.

[8] Abilene, "Indiana University Abilene NOC Weathermap," *http://loadrunner.uits.iu.edu/weathermaps/abilene/*.

[9] Weidong Cui, Ion Stoica, and Randy H. Katz, "Backup Path Allocation Based on a Correlated Link Failure Probability Model in Overlay Networks," in *10th IEEE International Conference on Network Protocols (ICNP'02)*, Paris, France, 2002.

[10] Rebecca Braynard and Amin Vahdat, "Masking Failures Using Anti Entropy and Redundant Independent Paths," in *SOSP Work in Progress Presentation*, 2001.

[11] Akihiro Nakao, Larry Peterson, and Andy Bavier, "A Routing Underlay for Overlay Networks," in *SIGCOMM*, Karlsruhe, Germany, 2003.

[12] L. Subramanian, I. Stoica, H. Balakrishnan, and R.H.Katz, "OverQoS: Offering Internet QoS using Overlays," *Proc. HotNet-I Workshop, October 2002*, 2002.

[13] Thinh Nguyen and Avideh Zakhor, "Path Diversity with Forward Error Correction (PDF) System for Packet Switched Networks," in *IEEE INFOCOM*, 2003.

[14] David G. Andersen, Alex C. Snoereny, and Hari Balakrishnan, "Best-Path vs. Multi-Path Overlay Routing," in *IMC'03*, Miami Beach, Florida, USA, 2003.

[15] John W. Byers, Michael Luby, Michael Mitzenmacher, and Ashutosh Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," in *SIGCOMM*, 1998.

[16] Manish Jain and Constantinos Dovrolis, "Pathload: A Measurement Tool for End-to-end Available Bandwidth," in *Proc. of Passive and Active Measurement Workshop*, Fort Collins, CO, 2002.

[17] L. Rizzo, "Effective Erasure Codes for Reliable Computer Communication Protocols," *Computer Communication Review*, 1997.

[18] Johannes Blomer, Malik Kalfane, Richard Karp, Marek Karpinski, Michael Luby, and David Zuckerman, "An xor-based erasure-resilient coding scheme," *Technical report, International Computer Science Institute, Berkeley, California*, 1995.

[19] T. Hacker, B. Athey, and B. Noble, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network," *Proceedings of the 16th Intemational Parallel and Distributed Processing Symposium (IPDPS)*, 2002.

[20] M. Yajnik, S. Moon, J. Kurose, and D. Towsley, "Measurement and Modeling of the Temporal Dependence in Packet Loss," *Proceedings of IEEE INFOCOM'99*, 1999.

[21] W. Jiang and H. Schulzrinne, "Modeling of Packet Loss and Delay and Their Effect on Real-Time Multimedia Service Quality," *10th International Workshop on Network and Operating System Support for Digital Audio and Video*, 2000.