Tavarua: Video Streaming with WWAN Striping

Asfandyar Qureshi
MIT-CSAIL
asfand@csail.mit.edu

Jennifer Carlisle
MIT-CSAIL
jc@csail.mit.edu

John Guttag
MIT-CSAIL
guttag@csail.mit.edu

ABSTRACT

Tavarua is a multimedia streaming system that leverages network-striping to deliver relatively high bit rate video over present-day cellular wireless wide-area networks. The Tavarua system achieves this by building on our previously developed flexible network-striping middleware. This paper describes a motivating mobile telemedicine application, and the design of the Tavarua system. It also describes experiments in which our initial Tavarua implementation was used to stripe video over multiple 3G cellular-phones from different providers.

Categories and Subject Descriptors

H.5 [**Information Systems**]: Information Interfaces and Presentation; J.4 [**Computer Applications**]: Life and Medical Sciences; D.4.4 [**Operating Systems**]: Communications Management—*network communication*

General Terms

Human Factors, Design, Measurement

Keywords

Video streaming, network striping, wireless wide-area networks, mobile systems, telemedicine.

1. INTRODUCTION

This paper describes a novel real-time multimedia communications sub-system designed to support mobile telemedicine applications.

Real-time telemedicine is not a new idea, it is at least as old as the telephone. However, over the last decade, the increasing availability of data communications has led to a dramatically increased interest in new applications of telemedicine. These applications can be viewed in terms of the communications technology employed:

- **Fixed installations with high bandwidth and ensured quality of service (QoS).** For example, the tele-stroke program at Massachusetts General Hospital (MGH) [2] uses dedicated tele-conferencing facilities to connect MGH with several distant sites. This allows stroke specialists at MGH to evaluate patients who have been brought to one of the remote sites.
- **Fixed or mobile facilities with high bandwidth but limited QoS guarantees.** For example, the Internet is often used to send medical images to radiologists, who then use the same network to send back their reports. Similarly, high quality photographic images are being used for remote dermatological consults [15].
- **Mobile facilities with low bandwidth and limited QoS guarantees.** For example, cell phones and radios are used to provide an audio link between EMT's in the field and care givers at hospitals.

We are in the process of developing a mobile application that occupies a different point in this space: high bandwidth, high QoS, mobile communications. In conjunction with teams in Orange County, Florida, and Massachusetts General Hospital we are building a system that will allow physicians to remotely evaluate the condition of patients in a moving ambulance. Potential uses include:

- For trauma victims, establishing effective communication between the emergency medicine system (EMS) pre-hospital team and the in-house trauma team is challenging. The physical distance and pressure of providing emergent care often prevent detailed and effective information exchange from one team to the other. A system that allows real-time audio and visual linkage of the two teams and real-time transmission of pre-hospital data to the hospital could lead to improved outcomes for patients.
- For some conditions, e.g., stroke or heart attack, prompt initiation of an appropriate therapy can have a dramatic effect on the outcome. Consultation with a physician during transport may reduce the time to intervention.
- When an ambulance picks up a patient following a 911 call, the patient is usually transported to an emergency department (ED). This may not always be the best choice: for instance, the patient may be suffering from

MM'06, October 23–27, 2006, Santa Barbara, California, USA.

an infectious disease, or the patient may not be in need of urgent care. Avoiding inappropriate transport is particularly important in geographical areas where over-crowding of ED's is a serious problem.

- In some cases, there can be questions about the appropriate destination for a patient. For example, should the patient be taken to a hospital with specialized facilities, to the nearest hospital, to a clinic, or perhaps nowhere (e.g., in the event of an inappropriate 911 call). A remote exam by a qualified physician, can be used to assist in making such decisions.

Building an application that meets these needs requires a mobile communications system that simultaneously handles real-time uni-directional video (on the order of $500kbps$, with a latency no more than a few seconds), bi-directional audio, and multiple physiological data streams (EKG, blood pressure, etc.).

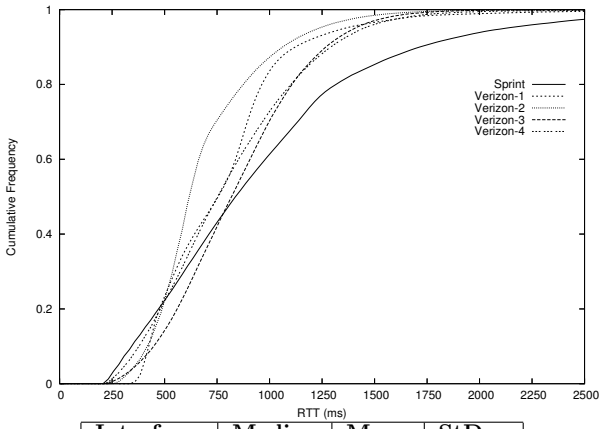
In most urban areas, there are a large number of public carrier wireless cellular channels providing mobile connectivity to the Internet. However, they are typically optimized for the downstream link, so the upstream bandwidth offered by these channels is far less than the advertised rates might lead one to believe. Furthermore, individual channels provide little in the way of QoS guarantees. These Wireless Wide Area Networks (WWAN's) are also dogged by high and variable round trip times, occasional outages, considerable burstiness, and high loss rates. The good news is that multiple WWAN providers provide overlapping coverage. Furthermore, our experiments in Orange County, Florida, and the Greater Boston area indicate that throughput from simultaneous WWAN connections can be combined to yield a high aggregate throughput [16].

These issues led us, over the last few years, to investigate using inverse multiplexing, or network striping, to aggregate several of these channels to provide virtual channels. By taking advantage of service provider diversity and overlapping coverage, we attempt to provide applications with the illusion that a reliable high-bandwidth channel is available.

In this paper, we describe Tavarua, a prototype system that builds on our network striping sub-system and on open-source video processing software, to develop a high-quality mobile telemedicine application. In Section 2, we report briefly on WWAN experiments we conducted to assess the actual behavior of the available cellular connections (which differs considerably from the best case behavior that is frequently advertised). In Section 3 we describe the various components of Tavarua. In particular, we describe Tribe, which provides the lowest level connection between Tavarua and the network interfaces; Horde, which provides the network striping layer; and Tavarua's video services subsystem. In Section 4 we report on an empirical evaluation of the most demanding component of the telemedicine system, the up-link video transmission. We transmit video, striping over multiple WWAN channels, and evaluate the impact of packet losses on the quality of the received video.

2. NETWORK CHARACTERISTICS

WWAN channel performance is hard to predict. Advertised network characteristics are rarely realistic. Furthermore, the performance at any given time depends on (among other things) the spatial placement of the WWAN interface relative to the provider's base-stations, competition with other users in the WWAN cell, and on artifacts arising from physical occlusion and electromagnetic interference.



Interface	Median	Mean	StDev
Sprint	845ms	979ms	655ms
Verizon-1	755ms	779ms	431ms
Verizon-2	715ms	793ms	317ms
Verizon-3	825ms	843ms	321ms
Verizon-4	755ms	806ms	380ms

Figure 1: Measured packet round-trip-time distributions from experiments conducted in Orlando.

Therefore, in order to ground the following discussion, we present a high-level summary of some WWAN experiments we have conducted in Orlando and Boston. The results of our experiments are presented in more detail elsewhere [16]. The goal of these experiments was to determine how well multiple WWAN interfaces would perform, if they were transmitting simultaneously in close proximity, inside a moving vehicle. The experiments consisted of driving over a large area constantly transmitting UDP packets, from multiple co-located WWAN interfaces, to a host on a wired ethernet in Boston. We used EV-DO interfaces from Sprint and Verizon.

Summary of results

In these experiments, we tested for available upload throughput, packet round-trip-times, and loss characteristics.

Packet Latency

The measured packet round-trip-times (RTTs) for 1024-byte packets were consistently high (around $600ms$) with high variance ($\sigma \approx 350ms$), even when the CDMA channels were otherwise well-behaved. Periods of elevated RTTs were not uncommon, with RTT spikes as high 3.5 seconds. Fortunately, the RTTs were not correlated across interfaces.

Disconnections

We experienced only a small number of short-lived (less than 30 seconds) disconnections during which an interface was unable to upload any data. Furthermore, these periods were not correlated across interfaces, suggesting that the diversity provided by multiple interfaces can be used to enhance reliability.

Figure 2: Measured coverage map for Boston. Darker squares indicate regions of higher throughput.

Throughput

Our analysis shows that in practice, the upload throughput seen by an interface varies considerably. However, contrary to our expectations, which were derived from simulation based network studies [9], we found no correlation between the vehicle's speed and achieved throughput. It seems that geographical location is the dominant factor leading to variation. This geography-based variation can be attributable to any of several factors including distance to base station, topography, occlusion by other structures, and electro-magnetic interference.

Figure 2 shows how aggregate throughput varied geographically during the Boston experiment. The map was constructed using GPS logs from the experiment.

Figure 3 shows how throughput varied over time during part of this experiment. The vehicle was moving relatively fast during this period. This snapshot covers four minutes from the middle of a long-running experiment. The ramp-up at the start of the Sprint graph is recovery from a short-lived disconnection.

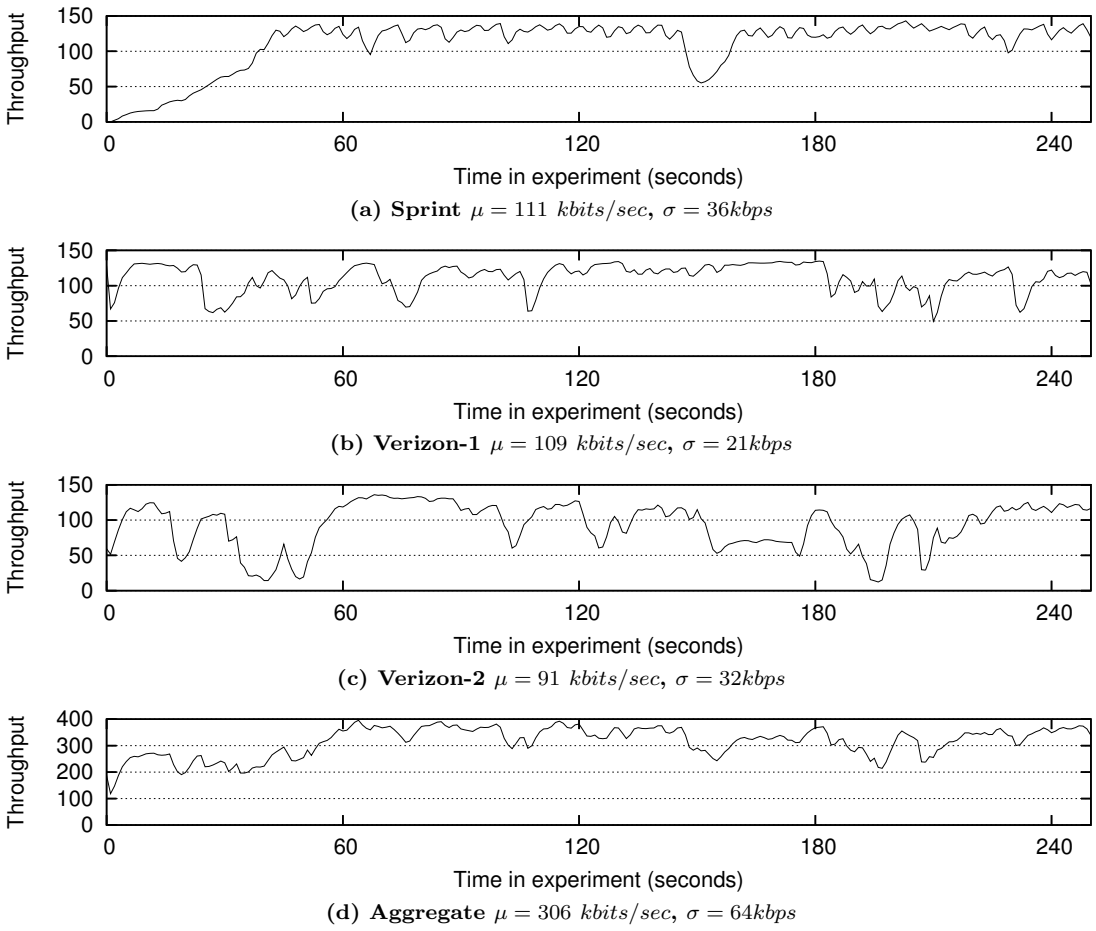
As indicated in the aggregate throughput graph, figure 3d, the use of multiple interfaces provides a considerable smoothing effect.

Disappointingly, the peak upload throughput never reached the rate advertised by network providers (never exceeding $140kbps$ per interface). We were also disappointed that our results suggest that the throughput provided by interfaces from the same provider can be strongly correlated in time.

Fortunately, our results indicate that it is possible to operate multiple WWAN interfaces to achieve a relatively consistent high aggregate throughput. Additional interfaces, even from the same provider, can boost aggregate throughput.

It is not immediately clear how many co-located WWAN interfaces one can operate simultaneously before experiencing cross-channel interference on either the air channel or further down stream. Our scalability experiments, using four interfaces from Verizon, made it clear that there is a point of diminishing returns for co-located network interfaces for the same carrier. Doubling the number of interfaces from Verizon in Orlando (from two to four) led to only a 50% increase in aggregate throughput.

From our analysis of these experiments, we estimate that we will need at least six WWAN interfaces to meet our goal of $500kbps$ video. We plan to use four from Verizon and two from Sprint. Verizon provides better coverage in our deployment area.



(a) Sprint $\mu = 111\ kbits/sec$, $\sigma = 36kbps$

(b) Verizon-1 $\mu = 109\ kbits/sec$, $\sigma = 21kbps$

(c) Verizon-2 $\mu = 91\ kbits/sec$, $\sigma = 32kbps$

(d) Aggregate $\mu = 306\ kbits/sec$, $\sigma = 64kbps$

Figure 3: An example of the dynamic variation in available upload throughput while moving in a vehicle.

3. TAVARUA: SYSTEM OVERVIEW

Figure 4 shows the structure of the Tavarua system.

The hardware in Tavarua consists of two main components, one placed on the ambulance, the other residing at a stationary command center, probably in a hospital. The component on the ambulance consists of a PC, multiple lightweight routers, WWAN interfaces, and video cameras.

The software in Tavarua consists of a collection of applications (video streaming, audio communications, telemetry transmission, etc), supported by networking middleware.

The networking middleware provides network striping capabilities to the applications. Application data is distributed over many network interfaces. The middleware arbitrates between different applications, and optimizes for each application's QoS requirements. The middleware also handles low-level issues related to the network interfaces (e.g., congestion control, disconnections and reconnections). An abstract striping interface is exposed to applications.

The set of applications is derived from the needs of our mobile telemedicine project. The video server has, by far, the highest network bandwidth requirements. The video server is designed to exploit the availability of multiple channels, and must be able to deal with network instability (both in terms of varying bandwidth and varying packet latencies). A separate application handles bi-directional voice commu-

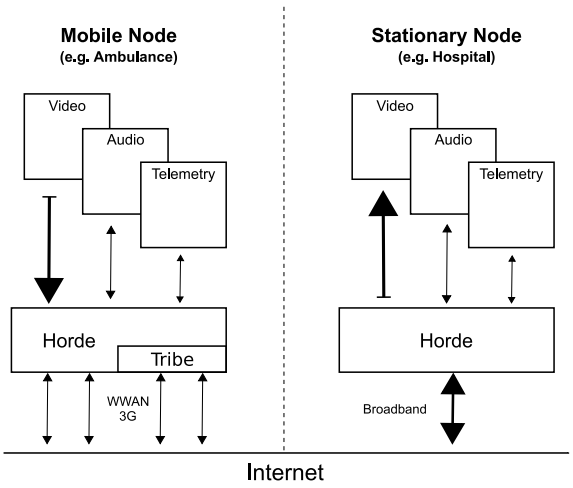


Figure 4: Overview of the Tavarua system.

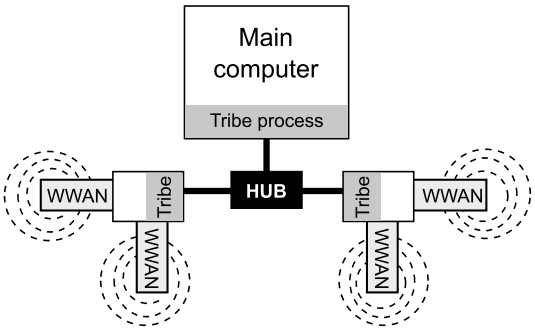


Figure 5: The Tribe protocol allows Horde to transparently connect to an arbitrary number of PCMCIA WWAN interfaces.

nications. Although the voice channel does not need to use network striping to overcome bandwidth limitations, it does take advantage of the multiple network channels to increase the reliability of the audio link. Finally, a third application, yet to be built, will handle the relatively low-bandwidth telemetry.

System reliability is an important issue. Tavarua is part of a multi-city multi-disciplinary research project. In the coming year, we expect to deploy Tavarua as part of a months-long medical study. During this period, the system needs to operate with high availability, and with minimal interaction with its engineers.

In order to ensure reliability, we have taken several steps to harden the system. The system is structured as a set of restartable components, keeping everything in user-space, and taking advantage of process-level isolation. Furthermore, we plan on using hardware watchdogs and idle-period detection to preemptively restart components.

In the following sections, we discuss aspects of the Tavarua system in more detail. We begin with the network subsystems, *Tribe* and *Horde*, and then address the applications. Because of the high network demands of streaming video and the ample opportunity for customization, we devote the application discussion to describing the video component. We consider the user interface component of the telemedicine system to be an important part of our research project, but that is not yet built.

3.1 Tribe: Connecting to Many Networks

Our network experiments imply that meeting our minimum bandwidth requirements using today's networks will require operating at least six WWAN interfaces simultaneously, and perhaps as many as ten. Each interface is a PCMCIA card. We considered using PCI-to-PCMCIA bridges to connect the necessary network interface cards to a single machine. In the end, however, we chose a more scalable software-based approach, using conventional off-the-shelf hardware. Figure 5 shows the structure of this solution.

The main computer is connected to a network of lightweight routers over a local high-speed ethernet. Each router box handles a maximum of two WWAN interfaces, forwarding packets to and from the main computer. In the present implementation, each box is a `net4521` embedded computer, from Soekris Engineering [3].

The *Tribe* protocol runs over the local ethernet. The protocol allows the main computer to keep track of active WWAN interfaces (as they come up and go down, in response to call disconnections and reconnections), and to remotely manage and restart services on the routers as necessary.

On the primary computer, Tribe emulates each active remote interface as a local interface. This emulation is accomplished using a standard `TUN/TAP` Linux kernel module. Packets sent to a `TUN` interface managed by Tribe are transparently tunneled over the local ethernet to the appropriate Soekris box, before being sent out over a WWAN interface.

With the Tribe in place, Horde has the ability to treat the remote WWAN interfaces as local interfaces. Tribe masks the additional complexity associated with managing the remote interface cards.

In addition to not requiring specialized hardware this solution also scales well. If additional bandwidth is desired and more WWAN interfaces are needed, we simply add more Soekris boxes.

Tribe is not Horde specific. It can support any Linux application that uses TCP, UDP, or ICMP. Like Horde, Tribe runs entirely in user-space, requiring only a small set of elevated privileges, such as raw socket capabilities.

3.2 Horde: Network Striping

Network striping takes data from a single source channel, sends it in some order over a set of smaller channels, and, if appropriate, reassembles the data in the correct order at the other end. A great deal of work has been done on network striping [6, 13, 14, 18, 20]. Most of this work is aimed at providing improved scheduling algorithms under the assumption that the underlying links are relatively stable and homogeneous.

Unfortunately, in our environment the underlying links are neither stable nor homogeneous (see §2). Furthermore, the applications in our telemedicine system generate multiple data streams that are heterogeneous with respect to their service needs. Some application streams have packet-level latency requirements and others have file-level latency requirements. Some application streams are more sensitive to packet losses than are others. Some application streams are sensitive to jitter, while others are not. Etc.

In many cases, researchers have decided to make the striping layer invisible to applications. Applications are written as if there is a single large transmission channel. However, when striping heterogeneous data streams over a multi-provider set of WWAN channels, the manner in which the striping middleware decides to schedule the transmission of application packets can have a large influence on observed packet latencies, stream loss rates, and throughput. This, in turn, can have a large impact on the utility the application layer derives from network service. This suggests that it is important to allow applications some control over the way in which striping is done.

One approach to doing this involves application-specific code to perform the striping. This is often the method chosen by multi-path video streaming applications [19, 7, 8]. However, this can lead to complicated application-level code, and typically incorporates implicit assumptions about the actual network channels.

We have been developing a different approach, based upon the Horde middleware system. Horde, which we have been developing over the last several years, allows a collection of application data streams to provide abstract information

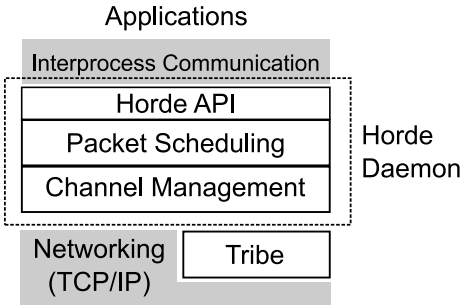


Figure 6: The network stack used by Tavarua.

about desired performance characteristics to a striping subsystem. The subsystem uses this information to stripe data from these streams across a set of dynamically varying network channels. The main technical challenge in Horde is giving the application control over certain aspects of the data striping operation (e.g., an application may want urgent data to be sent over low latency channels or critical data over high reliability channels) while at the same time shielding the application from low-level details.

The key idea in Horde is separating the striping mechanism from the striping policy. Horde is not meant to be a general networking middleware: if most application data can be sent down a single, stable link, using Horde is overkill. In situations where one is dealing with a fixed set of relatively homogeneous and stable channels, other techniques are probably more appropriate. The middleware is designed for QoS-limited applications operating in mobile settings over sets of heterogeneous wireless links. In such settings, the packet scheduler has an opportunity to significantly modulate observed QoS.

Figure 6 shows the network stack used by Tavarua. Tavarua applications communicate with a local Horde daemon which provides the network striping service. Horde can both connect directly to UDP sockets and go through Tribe. Inside the Horde daemon no distinction is made between Tribe-managed interfaces and native interfaces. The internal structure of the Horde daemon can be simplified as the three layers shown in the figure. These are discussed briefly below.

Horde API

Horde uses a connection-based model. Two hosts must first establish a shared session. Then the applications on the source and destination nodes must negotiate one or more connections, or *streams*, inside that session, before they start sending data to each other.

When sending/receiving data on a stream, applications communicate with Horde at the granularity of Application Data Units (ADU's). Our design draws from previous arguments for application level framing [12]. Callbacks are used to notify applications about receptions of ADU fragments and detections of fragment losses. Losses are detected as a side-effect of the congestion control algorithms in Horde.

If the application is sensitive to some network QoS aspects for an ADU, it can tag the ADU with various *objectives* in order to modulate the network QoS for that ADU. Figure 7 shows some of the objectives supported by the current Horde implementation. These objectives are treated as hints for Horde's packet scheduler.

Objective	Description
Fragments-Allowed	If true (default), then the ADU may be fragmented during packetization.
Loss-Threshold	Given a loss threshold T, an ADU will not be sent in a transmission slot if the estimated loss probability for that slot is greater than T.
Correlation-Group	If two ADU's, A and B, are in the same correlation group, they will be sent so as to minimize the joint loss probability $P(A\ lost \wedge B\ lost)$. Notably, this means that they will not be packed into the same network packet. Furthermore, they may be spaced out in time on the same channel, or sent simultaneously over uncorrelated channels.

Figure 7: Some supported QoS objectives.

Packet scheduling

In Horde's middle layer, an outgoing packet scheduler decides how to schedule ADU's from the unsent ADU pool over the available channels. The packet scheduler packs ADU's into packets as transmission slots become available. Multiple ADU's can be packed into a single packet, and ADU's may be fragmented during packetization.

The present Horde implementation uses a greedy scheduling algorithm that uses hard-coded knowledge to deal with each supported type of QoS objective. A detailed discussion of the scheduler is beyond the scope of this paper. The Horde architecture allows for different schedulers to be loaded for different sessions. Earlier, we have experimented with randomized algorithms that supported more objectives, but were far less efficient [17].

Channel Management

The lowest layer in Horde presents an abstract view of the network channels to the higher layers of the middleware. This layer deals directly with the network channels. In particular it: monitors interfaces as they connect and disconnect; handles network I/O; maintains a predictive model for the QoS on the channel, based on historical statistics; and also performs congestion control.

Horde runs an independent congestion control session for each underlying network channel. Congestion control is implemented below the striping layer because when data is being striped there are multiple channels, and so there are multiple congestion domains. Furthermore, keeping the congestion control logic below the packet scheduler allows us to transparently run congestion control schemes optimized for particular types of WWAN channels. For example, the Horde implementation evaluated in this paper uses a congestion control scheme optimized for CDMA2000 channels. A detailed description of the specific congestion control scheme is beyond the scope of this paper.

3.3 Video Services

Our telemedicine application must provide a number of image related services including high resolution still images, realtime video, non-realtime video (when network connectivity will not support realtime video), and various DVR-like functions. The most challenging of these is providing high quality realtime video, the topic of this subsection.

In the remainder of this subsection we first discuss the

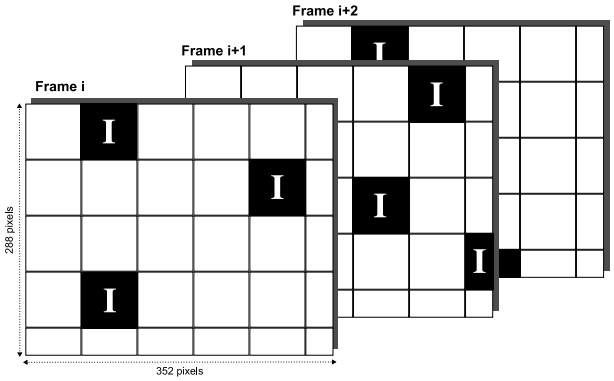


Figure 8: A grid of `x264` video encoders is run, with the encoders out of phase with each other. At most three encoders produce I-frames for any given frame. The I-frame pattern is randomly selected.

basic approach used for video encoding/decoding. The key problem to be solved is robustness in the face of packet losses and limited bandwidth.

Video Encoding/Decoding

Our video streaming application is built using the `ffmpeg` code-base [1]. `ffmpeg` is an open-source project that provides a feature-rich audio/video codec library. We use the H264 codec. Several video codecs were empirically evaluated at different frame rates and different bit-rates. H264 was chosen because it was observed to achieve the best quality at low bit-rates. We are using the `x264` encoder [5], which is an open-source implementation of H264.

Unfortunately, the standard `x264` encoder is not ideally suited to support video streaming in the face of restricted bandwidth and packet losses. The I-frames produced by the encoder are quite large and therefore span many packets, even at low bit-rates. For a `CIF`[1] video encoded at $300 kbits/sec$, more than sixteen UDP packets may be needed to hold a single I-frame. A single packet loss can corrupt an I-frame. In our experience, the `ffmpeg/x264` decoder does not handle partial I-frames well, and often crashes when asked to decode corrupted frames. We do not want to use retransmissions to mask losses because of the large packet round-trip-times in our system. Furthermore, given our limited bandwidth, adding forward-error-correction coding is not an appealing option.

To construct a video encoding resilient to packet losses, we use the following approach:

- Each frame is segmented in a grid of subframes. An example grid is shown in figure 8. The largest subimages in this grid are 64 pixels by 64 pixels.
- Each subframe is encoded and decoded independently of the other subframes. This requires grids of `ffmpeg/x264` encoders and decoders.
- Some of the subframes of each frame are intra-coded (i.e., treated as I-frames) and others are encoded as either P-frames or B-frames. See figure 8.

[1] 352 pixels wide, 288 pixels high.

The encoded subimages are linearized as follows:

$$..., D_j^{(0,0)}, D_j^{(1,0)}, ..., D_j^{(5,0)}, D_j^{(0,1)}, D_j^{(1,1)}, ..., D_j^{(5,4)}, D_{j+1}^{(0,0)}, ...$$

Where $D_j^{(x,y)}$ is the component for frame j from the encoder at (x, y) in the grid. Each $D_j^{(x,y)}$ is an ADU. Therefore, the video stream Horde sees consists of the above sequence.

The grid of video encoders is run so that encoders are out of phase with each other. All encoders do not begin encoding at the same frame. With a constant group of picture (GOP) size, this causes them to operate out of phase with each other. With 30 encoders and a GOP of 10 frames, at most three subframes of the original frame are intra-coded.

By creating more but smaller independently decodable I-frames, this encoding dramatically reduces the number of packets needed to transfer an I-frame component. Most I-frame components fit in a single packet. Consequently, the amount of received data that needs to be discarded due to a packet loss is also dramatically reduced.

Furthermore, a packet loss causes a localized corruption in part of the video, rather than corrupting the entire frame, and any other subsequent frames whose decoding depends on the missing data.

This approach also does not have to deal with bitrate spikes at each I-frame. For every frame, the network has to transfer some I-frame components, some P-frame components, and some B-frame components. This smooths out the transmission rate and improves packetization.

In order to avoid the correlated losses of I-frame components, these components can be tagged with the same correlation group G_I. Additionally, a loss threshold objective can be used for every ADU in the video stream, in order to avoid high-loss channels.

In the present implementation, the encoder phase differences are randomly selected on application startup. We may eventually use a non-random pattern. Our WWAN experiments imply that WWAN channels experience short burst losses. Furthermore, multiple encoded subimages are often packed into a single packet by horde's packetization algorithms. Therefore, we may want to avoid initializing contiguous encoders successively.

Network Bandwidth Variation

Given the mobile setting, the video application must deal with the frequent changes in available bandwidth, and still provide near real-time video. Variation is inevitable for a variety of reasons including congestion control, network interface disconnections, vehicular location, and competition with other users for base-station resources.

In an effort to mitigate the impact of variations in bandwidth, we dynamically adapt the encoder's Q parameter [21], which controls the bit-rate at which the video is encoded. Horde provides feedback to the application telling it the maximum available data transmission rate. Given this information, the video server can calculate the appropriate Q parameter value at which to encode the video.

The Q parameter is updated using two rules. Given slight changes in available bandwidth an additive-increase-additive-decrease rule is used to update Q. When a large difference between the encoding rate and the available bitrate is noticed (e.g., when a WWAN interface disconnects), a formula is used to predict the approximate Q value needed to encode at the new available bitrate.

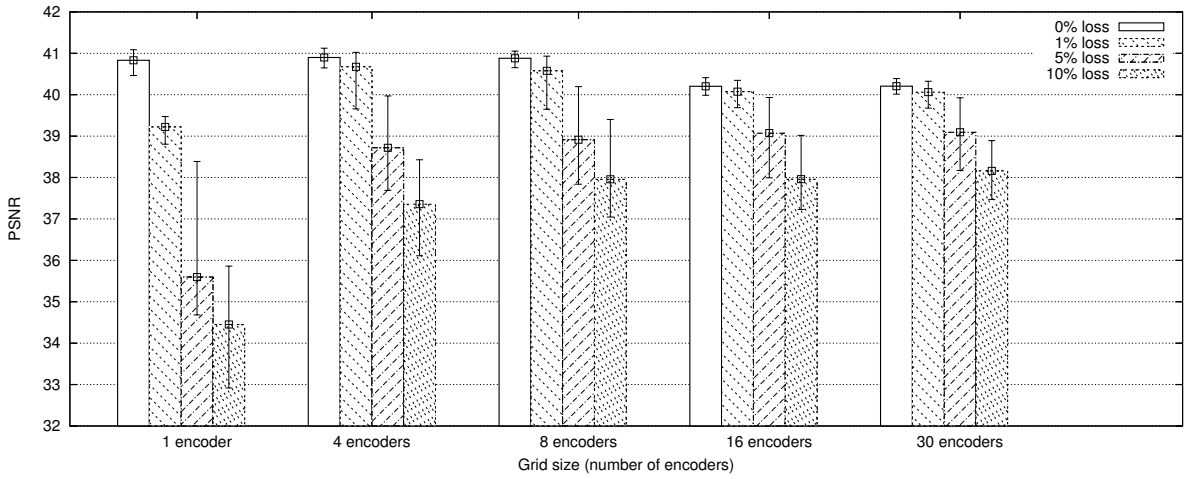


Figure 9: The impact of packet loss rates on the peak-signal-to-noise-ratio (PSNR) of decoded video for different types of grids.

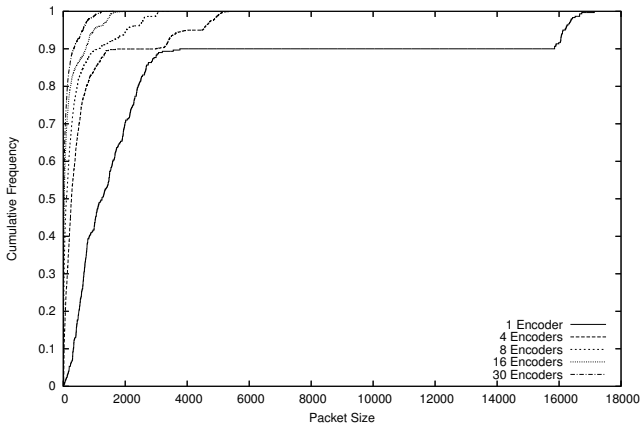


Figure 10: The distributions of ADU sizes for different types of grids.

4. EVALUATION

To evaluate the current implementation of our video streaming component we ran a series of tests. We first report on a simulation used to evaluate the resiliency of our encoder to packet losses. Then we present an experiment in which our implementation striped video over three WWAN interfaces.

Three undistorted original CIF sequences chosen from a standard video repository [4] were used as input data. The clips we chose were: `silent`, `mother-and-daughter`, and `bowing`. These videos were chosen to fit the model of our application, `bowing` provides a static background and a sudden jerky movement, not unlike a patient entering the ambulance and performing a task for the doctor. `mother-and-daughter` is a relatively static scene of two people talking, not unlike a paramedic talking with a patient. `silent` is a single subject displaying constant aperiodic motion.

The results presented in the sections below all use the `silent` clip. We chose `silent` because of the presence of motion throughout the clip.

4.1 Packet Loss Resilience

When transmitting encoded video over a network, packet losses can cause corruptions in the decoded video. We argued earlier that our grid-encoder would be resilient to packet losses. In this section we provide evidence for this claim by evaluating the impact of simulated packet losses.

We started by selecting five grid sizes: a single encoder; four encoders; eight encoders; sixteen encoders; and thirty encoders. This was done to gauge the performance of different grid sizes. For each grid size, we ran multiple experiments with simulated random packet losses. `silent` was encoded at around $500kbits/s$ and we simulated losses at 1%, 5% and 10% of total packets. Figure 9 shows the results of our experiments, showing the peak-signal-to-noise-ratio (PSNR) for the decoded video in each experiment. A higher PSNR implies higher quality video.

When there are no packet losses, the 4-encoder and 8-encoder grids performed almost as well as the single encoder. The 16-encoder and 30-encoder grids performed slightly worse. The problem is that working with smaller subframes adversely impacts compresssion. Therefore, when moving from the 8-encoder grid to the 16-encoder grid, we had to reduce the encoders' Q, by 1, in order to keep the bitrate roughly constant, leading to the signal degradation.

When a single encoder was used, increasing packet losses caused a sharp decrease in PSNR. In this case, when part of an I-frame was lost, all the packets for that I-frame (as many as 18) had to be discarded. This also caused subsequent frames dependent on that I-frame to become corrupted.

The 4-encoder case demonstrates that even a grid composed of relatively large sub-images yields a dramatic improvement. In this case, 1% packet losses did not significantly impact video quality.

Moving from four encoders to eight seems to yield a less dramatic improvement. Adding ever more encoders results in diminishing returns.

The quality improvement provided by the grid-encoder is due to two factors. First, smaller ADU's span fewer packets, so a single packet loss does not cause a large number of already received bytes to be discarded. Second, losing

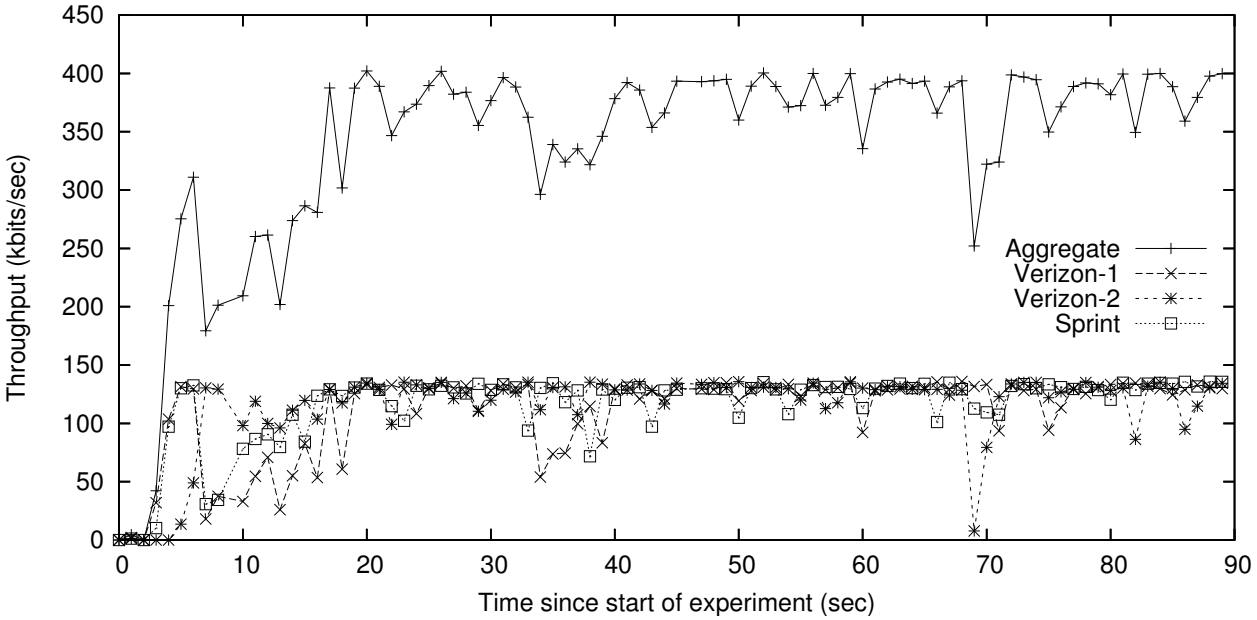


Figure 11: Network throughput in an experiment where Tavarua striped, over 3 CDMA interfaces, a looping `silent` clip using a grid of 30 encoders. Past the startup transient, the video was encoded at an average bitrate of around $375kbits/s$, and 1.5% of the packets were lost.

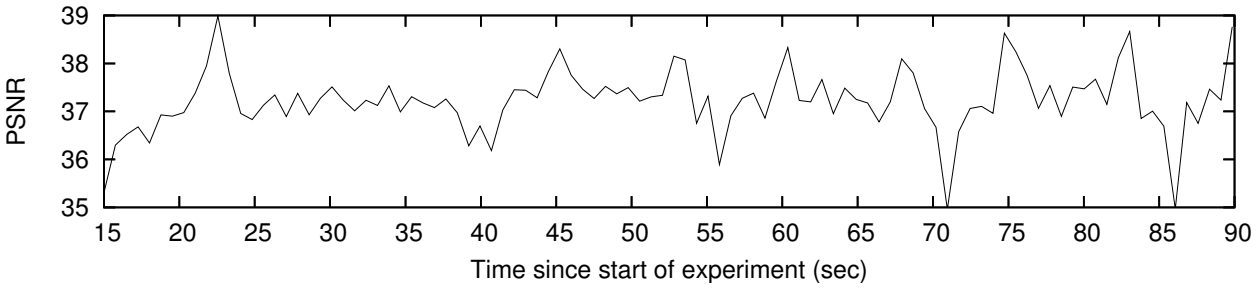


Figure 12: PSNR for decoded video from the experiment of figure 11.

packets causes localized image corruptions, rather than corrupting entire frames. For instance, losing a packet from a 30-encoder grid's video stream would lose less than 5% of the intra-coded information. Losing a packet from a single encoder's video stream would corrupt the entire frame.

The reduction in ADU size is likely the dominant factor here. Figure 10 shows the distribution of ADU sizes for each type of grid. Recall that in our video stream ADU's are encoded frames. Note the dramatic difference between the single encoder and 4-encoder cases. In contrast, all multi-encoder grids have relatively similar distributions.

4.2 Striping Video

Using the current implementation of Tavarua, we sent encoded video from a stationary host, striping over three WWAN interfaces (two EV-DO Verizon interfaces and one EV-DO Sprint interface), two of which were connected using Tribe (over 802.11). Figures 11 and 12 show the results of one such experiment.

Figure 11 shows the network throughput during the experiment. Each point represents the number of bits sent during one second-long non-overlapping windows. Beyond an initial startup transient, Horde's congestion control algorithm stabilizes and provides highly consistent throughput. The stability is achievable because Horde uses a channel-optimized congestion control scheme.

Figure 12 shows the PSNR for individual frames during the experiment. Figure 12 ignores PSNR's during the startup transient and the time-axis is slightly offset from the time-axis in figure 11 due to buffering at the receiver.

Part of the PSNR variance is due to changes in the Q parameter in response to changes in available throughput, and part of the variance is due to packet losses. For example, the PSNR peak at around 45 seconds in figure 12 is likely caused by the video encoder increasing its encoding bitrate in order to probe for additional bandwidth. The PSNR dip at around 70 seconds is caused by the throughput dip around the same time (see figure 11).

5. RELATED WORK

With the steady improvement of 3G technology, there have been several attempts at streaming multimedia using cellular networks [10]. While the fundamental issue of utilizing provider diversity to increase transmission reliability is similar, in many cases they do not aggregate the channels, and send packets redundantly over the multiple independent channels. Additionally, others have largely focused on the bandwidth-rich downlink channel, while our goals demand solutions for accommodating transmissions under the impoverished uplink channel.

The problems that surround sending delay-sensitive packets over unreliable IP links have also been studied. In [11] the idea of network striping to send multimedia data is addressed, but only for the case of constant transmission rates. Due to the mobile nature of our application, we need striping protocols that flexibly adapt to the available network.

Approaches such as multi-description video streaming [22] also address the issues of packet and compression losses.

6. SUMMARY AND CONCLUSION

This paper described the design and an evaluation of the current implementation of Tavarua, a novel real-time multimedia communications sub-system designed to support mobile telemedicine applications that require high bandwidth and QoS. The key hypothesis underlying Tavarua is that adequate bandwidth and QoS can be obtained by streaming data over multiple simultaneous public carrier data network connections.

The bulk of the paper is devoted to a description of the components of Tavarua: Tribe, which provides the lowest level connection between Tavarua and the network interfaces; Horde, which provides the network striping layer, including congestion control; and a video services subsystem.

We are quite encouraged by our initial evaluation of Tavarua. Our experiments demonstrate clearly that

- Our approach to video encoding significantly mitigates the impact of packet loss on video quality, and
- Our network striping system can be used to provide sufficient upstream service to transmit reasonably high quality video.

Based upon these results, we are moving forward on building a fully functional mobile telemedicine system.

7. ACKNOWLEDGEMENTS

This work was supported in part by the National Library of Medicine, N01LM33509 and by the Quanta Computer/MIT Project TParty. The authors would like to thank the Center for Integration of Medicine and Innovative Technology (CIMIT) for funding. The authors would also like to thank: the team in Orange County, FL, led by George Ralls, M.D., Salvatore Silvestri, M.D., Ana Handshuh; the team at Massachusetts General Hospital led by George Velmahos, M.D.; and Dorothy Curtis.

8. REFERENCES

[1] FFmpeg Multimedia System. http://ffmpeg.sourceforge.net.

[2] Massachusetts General Hospital Stroke Service. http://www.stopstroke.org.

[3] Soekris Engineering. http://www.soekris.com.

[4] Video Test Sequences. http://media.xiph.org/video/derf/.

[5] x264. http://developers.videolan.org/x264.html.

[6] H. Adiseshu, G. M. Parulkar, and G. Varghese. "A Reliable and Scalable Striping Protocol". In *SIGCOMM*, pages 131–141, 1996.

[7] J. Apostolopoulos and S. Wee. "Unbalanced Multiple Description Video Communication using Path Diversity".

[8] A. Begen, Y. Altunbasak, and O. Ergun. "Multi-path Selection for Multiple Description Encoded Video Streaming". In *IEEE ICC*, 2003.

[9] Q. Bi and S. Vitebsky. "Performance analysis of 3G-1X EVDO high data rate system". In *IEEE Wireless Communications and Networking Conference*, pages 389–395, March 2002.

[10] J. Chesterfield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez. "Exploiting Diversity to Enhance Multimedia Streaming over Cellular Links". In *IEEE INFOCOM*, March 2005.

[11] G. Cheung, P. Sharma, and S.-J. Lee. "Striping Delay-sensitive Packets over Multiple Burst-loss Channels with Random Delays". In *IEEE International Symposium on Multimedia*, 2005.

[12] D. Clark and D. Tennenhouse. "Architectural Consideration for a New Generation of Protocols". In *ACM SIGCOMM*, 1990.

[13] J. Duncanson. "Inverse Multiplexing". *IEEE Communications Magazine*, pages 34–41, April 1994.

[14] L. Magalhaes and R. Kravets. "Transport Level Mechanisms for Bandwidth Aggregation on Mobile Hosts". In *ICNP*, 2001.

[15] A. Oakley, M. Duffill, and P. Reeve. "Practising dermatology via telemedicine". In *The New Zealand Medical Journal*, pages 296–299, August 1998.

[16] A. Qureshi, J. Carlisle, and J. Guttag. "An Empirical Characterization of WWAN Performance". In *submission*, 2006.

[17] A. Qureshi and J. Guttag. "Horde: Separating Network Striping Policy from Mechanism". In *ACM MOBISYS*, 2005.

[18] P. Rodriguez, R. Chakravorty, J. Chesterfield, I. Pratt, and S. Banerjee. "MAR: A Commuter Router Infrastructure for the Mobile Internet". In *ACM MOBISYS*, 2004.

[19] E. Setton, Y. J. Liang, and B. Girod. "Multiple Description Video Streaming over Multiple Channels with Active Probing". In *IEEE International Conference on Multimedia and Expo*, 2003.

[20] A. Snoeren. "Adaptive Inverse Multiplexing for Wide-Area Wireless Networks". In *IEEE conference on Global Communications*, pages 1665–1672, 1999.

[21] T. Stockhammer and M. M. Hannuksela. "H.264/AVC video for wireless transmission". In *IEEE Wireless Communications*, August 2005.

[22] X. Su and B. W. Wah. "Multi-Description Video Streaming with Optimized Reconstruction-Based DCT and Neural-Network compensations". In *IEEE Transactions on Multimedia*, May 2001.