SmartTunnel: A Multipath Approach to Achieving Reliability in the Internet

Yi Li, Yin Zhang, LiLi Qiu, Simon Lam
Department of Computer Sciences, University of Texas at Austin
{ylee, yzhang, lili, lam}@cs.utexas.edu

***Abstract*— Reliability is critical to a variety of network applications. Unfortunately, due to lack of QoS support across ISP boundaries, it is difficult to achieve even two 9s (99%) reliability in today's Internet. In this paper, we propose *SmartTunnel*, an end-to-end approach to achieving reliability. A SmartTunnel is a logical point-to-point tunnel between two end points that spans multiple physical network paths. It achieves reliability by strategically allocating traffic onto multiple paths and performing FEC coding. Such an end-to-end approach requires no explicit QoS support from intermediate ISPs, and is therefore easy to deploy in today's Internet. To fully realize the potential of SmartTunnel, we analytically derive the near-optimal traffic allocation schemes that minimize loss rates. We extensively evaluate our approach using trace-driven simulations, ns-2 simulations, and experiments on PlanetLab. Our results clearly demonstrate that SmartTunnel is effective in achieving high reliability.**

I. Introduction

Many applications, such as Voice over IP (VoIP), video conferencing, streaming media, gaming, and online trading have stringent requirements on end-to-end reliability. Unfortunately, today's Internet does not even provide two 9s (99%) reliability [27], [9], [21], [15]. This is considerably lower than what the public switched telephone network (PSTN) offers today (three to four 9s).

Achieving end-to-end reliability is hard in the Internet for several reasons. First, there lack both incentives and mechanisms for ISPs to cooperate. This implies that it is hard to provide reliability guarantees across ISP boundaries. As a result, while it may be possible to achieve high reliability within an individual ISP, the reliability of end-to-end paths, spanning over multiple ISPs, is significantly lower. Second, numerous measurement studies [38], [32], [5], [6], [20], [39] have shown that Internet packet loss often exhibits burstiness (or temporal dependency). Bursty losses pose significant challenges to protect against. While forward error correction (FEC) coding is useful to protect against random losses, bursty packet losses significantly affect the effectiveness of FEC.

To address these challenges, we propose *SmartTunnel* to achieve reliability in today's Internet. A SmartTunnel is a logical point-to-point tunnel between two communicating end points that may physically span multiple Internet paths. It achieves reliability by properly allocating traffic across different paths and applying FEC coding. Such an end-to-end approach requires no explicit QoS support from intermediate ISPs, and can be directly applied to end hosts or Internet gateways. Therefore it is easy to deploy in today's Internet. Moreover, the simple tunnel abstraction can be naturally incorporated into applications and services such as virtual private networks to improve reliability.

To realize the potential of SmartTunnel, we analytically derive near-optimal traffic allocation schemes that optimize end-to-end loss rates under bursty loss.

Our key contributions can be summarized as follows:

- We propose SmartTunnel abstraction to provide end-to-end reliability. It serves as a thin waist at the network layer. This abstraction immediately supports diverse upper and lower network layer technologies without modification. It can also be easily incorporated into VPN services. Therefore SmartTunnel is easy to deploy in today's Internet.
- We analytically develop near-optimal traffic allocation schemes to optimize end-to-end loss rate in the presence of bursty packet losses. We further implement SmartTunnel using the click modular router [24].
- We extensively evaluate the effectiveness of SmartTunnel using trace-driven simulation, ns-2 simulations, and PlanetLab experiments. Our results show that SmartTunnel is effective in achieving high end-to-end reliability and multiple SmartTunnels can co-exist well.

The rest of the paper is organized as follows. In Section II, we survey related work. In Section III, we describe SmartTunnel architecture. In Section IV, we present algorithms that determine optimal traffic allocation. We describe our evaluation methodology and results in Section V. Finally we conclude in Section VI.

II. Related Work

We broadly classify the related work into the following three areas: (i) measurement of Internet reliability, (ii) overlay routing and multihoming, and (iii) FEC based loss recovery.

Measurement of Internet reliability: Several measurement studies have reported that Internet reliability is quite limited. In particular, Paxson observed a routing pathology arises 1.5% - 5% during 1994-1995. Dahlin [9], Jiang et al. [21], and Gummadi [15] conducted large-scale measurement studies of Internet path failures, and reported that Internet reliability is often below two 9s (99%).

Overlay routing and multihoming: Several studies, such as [33], [36], [30], have shown that the default routing path is often suboptimal in terms of latency, loss rate, and TCP throughput. To address these issues, a variety of overlay-based techniques have been proposed to improve network performance and resilience. For example, RON [30] allows

distributed applications to recover network path failures by routing through alternative overlay paths. Harrison et al. [17] uses edge-to-edge congestion control in an overlay framework. It requires modifications to edge routers, which is hard to achieve in practice. Duan et al. [10] provides QoS guarantees by buying bandwidth with certain SLAs from ISPs. As a result, its quality is highly dependent on the underlying network provider. OverQoS [35] uses a controlled loss virtual link abstraction to bound the loss rate. Since OverQos uses only a single path, it cannot protect against highly bursty losses in a timely fashion.

More recently, scalable one hop source routing (SOSR) [15] proposes that upon path failures the source node randomly chooses four nodes as relay nodes to re-route traffic. Their results show that it can recover 20-56% failures. SOSR is a reactive approach and requires failure indications. Therefore it is suitable for recovering long-term failures, but not for recovering bursty loss, which is our main focus.

There are quite a few research studies on the design and evaluation of route control schemes for multihomed users. For example, Cao et al. [8] propose using hash functions to achieve load balancing among multiple links. In [16], the authors compare several route selection schemes in a local area network and show that hashing can achieve performance comparable to load-sensitive route selection. Akella et al. [2] quantify the potential performance benefits of multihoming using real Internet traces. Their results show that smart routing has the potential to achieve an average performance improvement of 25% or more for a 2-multihomed user in most cases, and that most of the benefit can be achieved using 4 providers. In their follow-up work, the authors implement a Linux-based route control based on either passive or active monitoring schemes. Their experimental evaluation show that the route control schemes offer 15% to 25% performance improvement. The authors in [14] develop novel smart routing algorithms to simultaneously optimize cost and performance for multihomed users, and study the interactions between multihomed and single-homed users.

SmartTunnel can be applied to both overlay paths and multihoming paths. One of the fundamental differences between SmartTunnel and the previous work is that the previous work uses only a single path (e.g., send traffic along the best performing path). Due to bursty packet losses in the Internet, using a single path yields limited reliability. In comparison, SmartTunnel can achieve high reliability by simultaneously using multiple paths.

FEC based loss recovery: Significant research work has been done on the design and evaluation of forward error correcting code, such as [29], [26]. Our work is orthogonal to the development of FEC coding algorithms, and can directly apply the existing systematic FEC coding schemes (i.e., FEC codes that include unmodified original data packets in the FEC group).

FEC coding has also been applied to multicast transmission [7], [25]. In these studies, only a single network path is used for sending data to each destination.

Jain *et al.* [19] study the problem of traffic allocation onto multiple paths to achieve high reliability. This is a pioneering work on this subject. It is also the work closest to ours. Different from our work, [19] targets delay tolerant networks, and the proposed approach is based on very different loss models from the Internet. In particular, they do not consider bursty loss. As a result, their approach does not work well for Internet paths, as we will show in Section V. Note that in order to cope with bursty loss, it is necessary to develop a completely different traffic allocation scheme (as opposed to a simple extension to [19]). We will further elaborate this point and compare SmartTunnel with [19] in Section IV-A.

III. SmartTunnel Architecture

SmartTunnel is a logical point-to-point tunnel between two communicating end points that may physically span multiple Internet paths. It sits at network-layer, and is transparent to applications. As shown in Figure 1, a tunnel source continuously monitors the network paths, and provides the performance of network paths to the controller. The controller applies the traffic allocation algorithm, described in Section IV, to distribute traffic onto multiple physical paths. On the data plane, data is first delivered to FEC encoder, which generates redundancy packets and hands over the resulting data and redundancy packets to the traffic distributor that stripes packets according to the controller's specification. The tunnel destination decodes and buffers data. Buffering is necessary to reduce packet reordering, which can degrade TCP [18] performance. A packet is delivered to uppper layers either when all the packets before it have been received (or recovered) or when buffer is full.

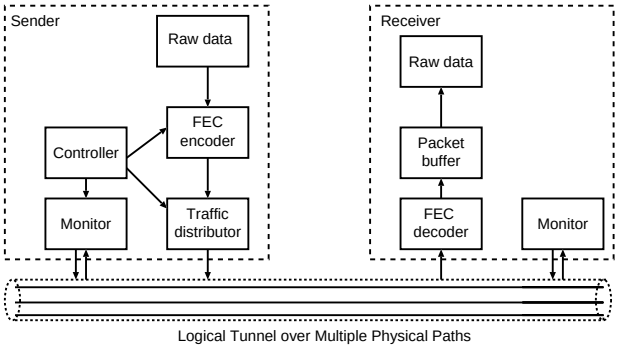


Fig. 1. SmartTunnel architecture.

SmartTunnel can be deployed either at the two communicating end-points or at their Internet gateways. SmartTunnel can also be easily integrated with current VPNs. Today's VPNs only provide reachability and security, but no reliability. By running SmartTunnel sender and receiver at the VPN servers of the two communicating points, network providers can offer value-added services.

IV. SmartTunnel Algorithms

To fully realize the potential of SmartTunnel, we need to address the following issues: (i) how to allocate traffic onto multiple paths to minimize loss rates under realistic Internet loss models, and (ii) how to measure path properties which can be used for the allocation schemes. In this section, we examine these issues in turn.

A. Allocation Problem Formulation

First we formulate the traffic allocation problem for minimizing packet loss rate. Table I summarizes our notations.

N	number of physical paths
D	number of data packets per FEC group
R	number of redundant packets per FEC group
G	FEC group size ($G = D + R$)
d_i	number of allocated data packets on path i
r_i	number of allocated redundant packets on path i
x_i	number of lost data packets on path i
y_i	number of lost redundant packets on path i
X	total number of lost data packets $X = \sum_{i=1}^{N} x_i$
Y	total number of lost redundant packets $Y = \sum_{i=1}^{N} y_i$
$\bar{X}_{\rm FEC}$	expected number of lost data packets after applying FEC: $\bar{X}_{\rm FEC} = \sum_{\ell=1}^{D} \ell \cdot Pr[X = \ell \wedge X + Y > R]$
$\bar{X}^{\approx}_{\rm FEC}$	continuous approximation of $\bar{X}_{\rm FEC}$ (see Section IV-B.1)

TABLE I
NOTATIONS

Consider a SmartTunnel from node s to node d. Traffic from node s to node d can take several different *physical paths*, which can be provided by either multihoming or overlay routing. Let N denote the number of physical paths available to the SmartTunnel. When a packet is transmitted along a physical path, it can get lost due to a variety of reasons, such as routing loops, failures, and network congestion. To achieve reliability, node s applies forward error correction (FEC) code to protect data packets that enter the SmartTunnel. Specifically, for every D data packets, node s creates R redundant packets, which together with all the data packets form an FEC group of size $G = D + R$. The FEC code is designed to recover from R packet losses. That is, when node d receives any D out of G packets in the group, it can reconstruct the entire FEC group. If more than R packets are lost, however, node d can only deliver those successfully received data packets and cannot recover the lost data packets.

Note that in this paper we only consider *systematic* FEC codes, which include the unmodified input data packets in the FEC group. Compared with non-systematic FEC codes that alter the input data packets, one major advantage of systematic FEC codes is that even if more than R packets are lost in an FEC group, all the received data packets can still be delivered, whereas with non-systematic FEC codes the entire group is lost. Another advantage of systematic FEC codes is that no decoding is needed when no data packet is lost. As a result, most FEC codes in practice are systematic. One of the most famous systematic FEC codes is the Reed-Solomon code [29], which we will use in our SmartTunnel prototype implementation. Our experience suggests that highly optimized software implementation [28] of the Reed-Solomon code can achieve an encoding/decoding rate of over 4 Gbps on a 2.4 GHz Pentium IV processor.

Given an FEC group with D data packets and R redundant packets, there are N^{D+R} different ways of allocating packets in an FEC group onto N physical paths. Different allocations can result in different numbers of packet losses after applying FEC. The goal of the SmartTunnel is to derive an optimal allocation that minimizes the expected number of packet losses after applying FEC. This problem can be formally specified as follows.

Definition 1 (Optimal Allocation Problem): Let d_i and r_i be the number of data packets and redundant packets allocated on path i. Let random variables x_i and y_i be the number of data packets and redundant packets that are lost on path i. Let random variables $X = \sum_{i=1}^{N} x_i$ and $Y = \sum_{i=1}^{N} y_i$ be the total number of lost data packets and lost redundant packets, respectively. The optimal allocation problem is to determine an allocation $\{(d_i, r_i)\}$ under which the expected number of lost data packets after applying FEC is minimized. That is,

$$\text{minimize} \quad \bar{X}_{\rm FEC} \equiv \sum_{\ell=1}^{D} \ell \cdot Pr[X = \ell \wedge X + Y > R] \quad (1)$$

where $\ell = 1, \cdots, D$ enumerates all possible values for X (*i.e.*, the total number of lost data packets), and the summation gives the expected number of lost data packets when FEC cannot reconstruct the entire FEC group (*i.e.*, when $X + Y > R$).

Modeling Temporal Loss Dependency in the Internet: To solve the above optimization problem, we first need to understand the behavior of Internet packet loss. Numerous measurement studies [38], [32], [5], [6], [20], [39] have shown that Internet packet loss often exhibits burstiness. Burstiness affects the performance of FEC because when a large burst of packets are lost in an FEC group, FEC cannot recover the lost data packets. It is therefore important to explicitly model bursty loss in SmartTunnel.

A variety of models have been proposed to capture temporal loss dependency, including the Gilbert Model, the Extended Gilbert Model, and the Markov Chain Model [38], [32], [5], [6], [20]. In this paper, we use the following General Markov Model, which generalizes all the above models.

A General Markov Model has the following parameters:

- n states: $0, 1, \cdots, n-1$.
- A length-n loss probability vector $L = [\ell_0, \cdots, \ell_{n-1}]$, where ℓ_i denotes the packet loss probability in state i.
- A $n \times n$ transition matrix $P = [p_{ij}]_{n \times n}$ ($0 \le i, j \le n-1$), where p_{ij} denotes the probability for the next state to be j provided that the current state is i.

Suppose the current state is i. A newly transmitted packet first results in a state transition – the probability for the next state to become j is given by p_{ij}. The loss probability ℓ_j associated with the new state j then determines whether this packet gets lost. Temporal loss dependency is captured by the different state transition probabilities p_{ij} and the different packet loss rates in different states. To better illustrate the model, below we consider two important special cases.

- *Gilbert Model.* The Gilbert Model [13] is a special case of the General Markov Model that is simple to understand and to implement in monitoring applications. It has been analyzed in [32], [38], [6]. As illustrated in Figure 2, a Gilbert Model has two states: 0 for "no-loss" and 1 for "loss". The corresponding loss probabilities are: $\ell_0 = 0$ and $\ell_1 = 1$. Normally, $p_{11} \ge p_{01}$. So a packet is more likely to get lost if the previous packet is already lost. When $p_{11} = p_{01}$, the Gilbert Model further reduces to a Bernoulli model, which has no temporal dependency. Note that it is also possible to use non-binary loss probabilities $0 < \ell_0 <$

$\ell_1 < 1$. The resulted model is often termed a Gilbert-Elliott Model [13], [11].

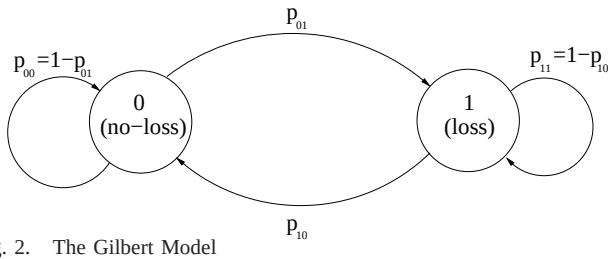


Fig. 2. The Gilbert Model

- *Extended Gilbert Model.* Sanneck *et al.* [32] proposes an Extended Gilbert Model, which generalizes the simple Gilbert Model. The Extended Gilbert Model captures changes in the loss burst length, as shown in Figure 3. Specifically, in an n-state Extended Gilbert Model, state i ($i = 0, 1, ..., n-2$) means that there are exactly i consecutive losses since the beginning of the current loss burst, whereas state $n-1$ means that $n-1$ or more consecutive losses have occurred. The corresponding loss probability vector is $L = [0, 1, \cdots, 1]$ (*i.e.*, $\ell_0 = 0$ and for $\forall i > 0, \ell_i = 1$). Suppose the current loss burst length is i. For a newly transmitted packet, it will either get lost with probability $p_{i(i+1)}$ and cause the burst length to increment by one, or get through successfully with probability $p_{i0} = 1 - p_{i(i+1)}$ and reset the burst length to 0. No other state transitions are allowed. So the model is fully specified by n parameters p_{i0}, and the corresponding transition matrix has only $2n$ non-zero entries.

$$P = \begin{bmatrix} p_{00} & p_{10} & p_{20} & \cdots & p_{(n-2)0} & p_{(n-1)0} \\ p_{01} & 0 & 0 & \cdots & 0 & 0 \\ 0 & p_{12} & 0 & \cdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & \cdots & p_{(n-2)(n-1)} & p_{(n-1)(n-1)} \end{bmatrix} \quad (2)$$

Note that all our analytical results in this paper apply to the General Markov Model (thus also to the other models). *For practical implementation purposes, however, we only focus on the Extended Gilbert Model*. As shown in [32], [20], the Extended Gilbert Model achieves a good balance between model accuracy and simplicity – it is much more accurate than the 2-state Gilbert Model, while only requires n parameters to be estimated (as oppose to n^2 in the General Markov Model).

Why are new algorithms and techniques required? The problem of using redundancy to cope with failures has recently been considered in the context of Delay Tolerant Networks (DTNs) by Jain *et al.* [19]. They use a similar problem formulation to optimally allocate packets in an FEC group onto different paths. Given such similarity, one may be tempted to directly apply the techniques developed in [19] to the Internet. Unfortunately, several important reasons prevent us from applying these techniques to achieve high reliability in the Internet. Therefore, we have to develop new algorithms and techniques for use in SmartTunnel.

- *Burstiness in Internet packet loss.* Two different loss models are considered in [19]: independent packet loss (which exhibits no temporal loss dependency) and complete path failure (where a path delivers either all packets successfully or no packet at all). While these models may be useful in a DTN, neither model captures the commonly observed bursty loss behavior in the Internet. Since the goal of SmartTunnel is to minimize loss in the Internet context, it is essential to use models that capture the Internet loss behavior more accurately.
- *The use of different performance metrics.* Jain *et al.* [19] try to minimize the expected FEC *group* loss probability (as opposed to the *packet* loss probability), which is defined as the probability for FEC to be unable to recover an entire FEC group (*i.e.*, $Pr[X + Y > R]$). While this metric captures the performance of non-systematic FEC codes, it is too conservative for systematic FEC codes. As noted above, a major advantage of systematic FEC codes is that even if the entire group cannot be recovered (*i.e.*, $X + Y > R$), all the unmodified data packets that arrive successfully are still available. To capture the performance of systematic FEC codes, we use a new metric ($\bar{X}_{\text{FEC}}$), which is more difficult to optimize and calls for the development of new optimization algorithms (in Section IV-B).

Later in Section V, we will thoroughly compare our algorithms with the algorithms proposed in [19] along with several other baseline algorithms, and show that our algorithms significantly out-perform the existing ones under bursty losses.

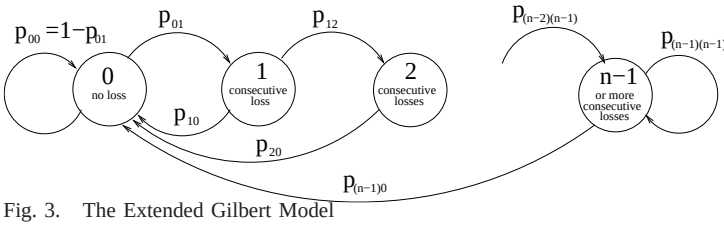


Fig. 3. The Extended Gilbert Model

B. Allocation Algorithms

We decompose the original optimal allocation problem into the following two sub-problems:

1. *Given an allocation $\{(d_i, r_i)\}$, how to compute $\bar{X}_{\text{FEC}}$, the expected number of lost data packets after applying FEC?* This is challenging because random variables X and Y are convolutions of random variables x_i and y_i (*i.e.*, the numbers of lost data and redundant packets on path i) and have no close form in general. In Section IV-B.1, we address the challenge by approximating the joint distribution of X and Y as a bivariate normal distribution.
2. *How to find an allocation $\{(d_i, r_i)\}$ that minimizes $\bar{X}_{\text{FEC}}$?* The key challenge here is the enormous search space. Given D data packets and R redundant packets, there are N^{D+R} different ways of allocating them onto N different physical paths. For even moderate FEC group sizes, this is already a too big search space for a brute-force approach to work (e.g., 20-packet FEC group using 3 paths has 3486784401 combinations!). To address the issue, we develop an efficient dynamic programming algorithm to find an optimal allocation in Section IV-B.2.

1) Approximating $\bar{X}_{\rm FEC}$: Our first task is to estimate $\bar{X}_{\rm FEC}$, the expected number of lost data packets after applying FEC under a given allocation $\{(d_i, r_i)\}$. As noted above, our basic strategy is to approximate the joint distribution of X and Y with a bivariate normal distribution. Such normal approximation is reasonable when the number of independent paths is large and the allocation (d_i, r_i) on each path is relatively small compared to (D, R). In addition, our experience suggests that even when these conditions do not hold, the allocation obtained using the normal approximation tends to work well in practice. Similar positive experience has been reported in [19].

As shown in the Appendix , we can derive $E[x_i]$, $E[y_i]$, $V[x_i]$, $V[y_i]$, and $cov[x_i, y_i]$ as functions of a given allocation $\{(d_i, r_i)\}$. Our analysis takes into account the fact that different values of (d_i, r_i) can affect the burstiness of the packet loss observed on a path i. This phenomenon has been reported by several recent studies. For example, the authors in [34], [4] show that burstiness of probing traffic significantly affect the burstiness in the observed loss rates. As the inter-arrival of probing packets increases, the observed loss burstiness decreases.

Based on $E[x_i]$, $E[y_i]$, $V[x_i]$, $V[y_i]$, and $cov[x_i, y_i]$, we then compute the statistics of the total numbers of lost data and redundancy packets as follows. Let μ_X and σ_X denote the mean and standard deviation of X, respectively. Let μ_Y and σ_Y denote the mean and standard deviation of Y, respectively. Let $\sigma_{XY} = cov[X, Y]$ be the covariance of X and Y. In order to derive μ_X, σ_X, μ_Y, σ_Y, and σ_{XY} as functions of a given allocation $\{(d_i, r_i)\}$, we will assume that *losses on different paths are independent from each other*. Later in Section IV-E, we will discuss techniques that can be used to detect and remove paths with shared congestion. Under the independence assumption, we have

$$\mu_X = \sum_i^N E[x_i], \qquad \sigma_X^2 = \sum_i^N V[x_i]$$
$$\mu_Y = \sum_i^N E[y_i], \qquad \sigma_Y^2 = \sum_i^N V[y_i]$$
$$\sigma_{XY} = \sum_i^N cov[x_i, y_i]$$

.

We can then approximate the joint distribution of X and Y by a bivariate normal distribution with probability function

$$P(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y\sqrt{1-\rho^2}} \exp\left[-\frac{z}{2(1-\rho^2)}\right], \quad (3)$$

where

$$z \equiv \frac{(x-\mu_X)^2}{\sigma_X^2} - \frac{2\rho(x-\mu_X)(y-\mu_Y)}{\sigma_X\sigma_Y} + \frac{(y-\mu_Y)^2}{\sigma_Y^2},$$

and

$$\rho \equiv cor[X,Y] = \frac{\sigma_{XY}}{\sigma_X\sigma_Y}$$

is the correlation of X and Y.

In the special case when $\rho = 0$ (*i.e.*, X and Y are independent), (3) can be further simplified into

$$P(x,y) = \frac{1}{2\pi\sigma_X\sigma_Y} \mathbf{e}^{-\frac{(x-\mu_X)^2}{2\sigma_X^2} - \frac{(y-\mu_Y)^2}{2\sigma_Y^2}}. \quad (4)$$

Our experience through extensive simulations suggests that the correlation of X and Y (*i.e.*, ρ) is often very small when multiple paths and sufficiently large FEC group size are used. For practical purposes, its effect can be safely ignored compared to the effects of μ_X, σ_X, μ_Y, and σ_Y. *We therefore will use (4) in the rest of the paper in the interest of simplicity*.

By replacing the discrete summation in (1) with a continuous integral, we can then approximate $\bar{X}_{\rm FEC}$ by

$$\bar{X}_{\rm FEC}^{\approx} \equiv \int_{x=0}^{D}\int_{y=R-x}^{\infty} x\,P(x,y)\;dy\;dx \quad (5)$$
$$= \int_{x=0}^{D}\int_{y=R-x}^{\infty} \frac{x}{2\pi\sigma_X\sigma_Y} \mathbf{e}^{-\frac{(x-\mu_X)^2}{2\sigma_X^2} - \frac{(y-\mu_Y)^2}{2\sigma_Y^2}}\;dy\;dx \quad (6)$$
$$= \int_{x=0}^{D} \frac{\mathbf{e}^{-\frac{(x-\mu_X)^2}{2\sigma_X^2}}}{2\sqrt{2\pi}\sigma_X} x\left(1 + \text{erf}\left[\frac{\mu_Y + x - R}{\sqrt{2}\sigma_Y}\right]\right) dx \quad (7)$$

where $\text{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z \mathbf{e}^{-t^2} dt$ is the error function [37]. We can then numerically evaluate $\bar{X}_{\rm FEC}^{\approx}$ using standard software package such as Matlab.

2) Dynamic Programming Based Solution: Our second major task is to find an optimal allocation $\{(d_i, r_i)\}$ that minimizes $\bar{X}_{\rm FEC}^{\approx}$ as defined in (7). The main challenge here is that $\bar{X}_{\rm FEC}^{\approx}$ does not have close-form and cannot be easily transformed into simple objective functions. Fortunately, from (7) we can show that $\bar{X}_{\rm FEC}^{\approx}$ is monotonically decreasing with respect to both σ_X and σ_Y.

Our high-level approach to traffic allocation under no capacity constraints is as follows. We enumerate all possible values of μ_X, and use dynamic programming to find the data allocation that results in the minimum σ_X for each given μ_X. Similarly, for all possible values of μ_Y, we determine the allocation of redundancy packets that minimizes σ_Y for each given μ_Y. Then we plug all $(\mu_X, \mu_Y, \sigma_X, \sigma_Y)$ into Equation 7, and find the allocation that minimizes $\bar{X}_{\rm FEC}^{\approx}$. The monotonicity of $\bar{X}_{\rm FEC}^{\approx}$ with respect to σ_X and σ_Y ensures that the final solution gives the best possible $\bar{X}_{\rm FEC}^{\approx}$. Later we will extend the idea to handle capacity constraints.

Below we first describe how to determine the allocation that minimizes σ_X and σ_Y given μ_X and μ_Y. Then we show how to use these solutions to solve the original allocation problem.

Subproblem: Variance Minimization. Let $E\{x_i|d_i = k\}$ and $V\{x_i|d_i = k\}$ denote the average and variance of data losses on path i when path i is allocated $d_i = k$ data packets. To apply dynamic programming, we need integer values of $E\{x_i|d_i = k\}$ and $V\{x_i|d_i = k\}$. So we scale them by λ. Let $ed[i,k] \equiv \lfloor E\{x_i|d_i = k\}\lambda \rfloor$ and $vd[i,k] \equiv \lfloor V\{x_i|d_i = k\}\lambda \rfloor$ be the scaled, discretized mean and variance of x_i, where $\lfloor\cdot\rfloor$ is the floor function (i.e., taking the largest integer no larger than the input). We can pre-compute $ed[i,k]$ and $vd[i,k]$ for $\forall 1 \le k \le D$ and $\forall 1 \le i \le N$ as shown in Appendix .

Define a *data loss variance minimization* problem $\mathbf{dlvm}(d, p, e, c)$ as the problem of allocating d data packets

onto the first p paths to minimize the total variance while satisfying capacity constraints c and the constraint that the total mean is e. Formally,

$$\mathbf{dlvm}(d,p,e,c): \text{minimize} \quad \sum_{i=1}^{p} vd[i,d_i]$$

$$\text{subject to} \quad \begin{cases} \sum_{i=1}^{p} d_i = d \\ \sum_{i=1}^{p} ed[i,d_i] = e \\ d_i \le c[i], \forall i \end{cases}$$

Let $opt(d,p,e,c)$ be the variance achieved by an optimal solution to $\mathbf{dlvm}(d,p,e,c)$. We have

$$opt(d,p,e,c) = \min_{\substack{0\le k\le d \\ k\le c[p]}} \{vd[p,k] + opt(d-k, p-1, e-ed[p,k], c)\}$$

So we can apply a dynamic programming algorithm to solve $\mathbf{dlvm}(d,p,e,c)$ for all $0 \le d \le D$, $1 \le p \le N$, $0 \le e \le E_{\max}$, where E_{max} controls the granularity of the solutions. The complexity for this algorithm is $O(D^2 N E_{\max})$.

Similarly, we can define a *redundancy loss variance minimization* problem $\mathbf{rlvm}(r,p,e,c)$ as the problem of allocating r redundant packets onto the first p paths to minimize the total variance subject to the constraint that the total mean is e. We can solve $\mathbf{rlvm}(r,p,e,c)$ again using dynamic programming.

Traffic allocation under no capacity constraints. We are now ready to solve the original allocation problem. Let us first consider the unconstrained allocation problem, where $c_i = \infty$. In this case, data packets can be allocated independently from redundancy packets. So the final allocation is to allocate data packets such that $\mathbf{dlvm}(D,N,e_1,\infty)$ is minimized, and allocate redundancy packets such that $\mathbf{rlvm}(R,N,e_2,\infty)$ is minimized. The algorithm is illustrated in Figure 4. It searches through all possible values of μ_X and μ_Y, and determines the allocation that minimizes σ_X and σ_Y for each given μ_X and μ_Y. Note that the discretization may introduce some error. However, our experience from extensive simulations suggests that with sufficiently large $E_{\max}$ (*e.g.*, 500), the algorithm tends to perform close to optimal.

1. solve all $\mathbf{dlvm}(d,p,e,\infty)$ and $\mathbf{rlvm}(r,p,e,\infty)$. using dynamic programming
2. **for** $e_1 = 1$ **to** $E_{\max}$
3. $\{d_i\}$ = an optimal solution of $\mathbf{dlvm}(D,N,e_1,\infty)$
4. **for** $e_2 = 1$ **to** $E_{\max}$
5. $\{r_i\}$ = an optimal solution of $\mathbf{rlvm}(R,N,e_2,\infty)$
6. compute μ_X, μ_Y, σ_X, σ_Y under $\{(d_i,r_i)\}$
7. compute $\bar{X}^{\approx}_{\text{FEC}}$
8. store $\{(d_i,r_i)\}$ if it is the best so far
9. **end**
10. **end**
11. **return** $\{(d_i,r_i)\}$ that gives the minimum $\bar{X}^{\approx}_{\text{FEC}}$

Fig. 4. Traffic allocation under no capacity constraints.

Traffic allocation under capacity constraints. When network paths have capacity constraints, allocation of redundancy packets is dependent on the allocation of data packets to ensure the capacity constraints are satisfied. To incorporate capacity constraints, we modify the previous algorithm by introducing a new capacity constraint c' to ensure that the rate for sending redundancy packets on path i cannot exceed the residual capacity of path i (after sending its allocated data packet d_i).

The solution to the allocation problem with capacity constraints can be sub-optimal. Due to capacity constraints, the allocation of data packets is now coupled with the allocation of redundancy packets. Such coupling further increases the search space. For efficiency, we decouple the data and redundancy allocation by first optimizing data allocation (while ignoring the capacity consumed by redundancy packets), and then optimizing redundancy packet allocation based on remaining capacity. Such decoupling may result in sub-optimal solution. In practice, however, we find through extensive simulations that the solution we obtain tends to perform close to optimal.

1. solve all $\mathbf{dlvm}(d,p,e,c)$ via dynamic programming
2. **for** $e_1 = 1$ **to** $E_{\max}$
3. $\{d_i\}$ = an optimal solution of $\mathbf{dlvm}(D,N,e_1,c)$
4. $c'_i = c_i - d_i$, $\forall 1 \le i \le N$
5. solve all $\mathbf{rlvm}(r,p,e,c')$ via D.P.
6. **for** $e_2 = 1$ **to** $E_{\max}$
7. $\{r_i\}$ = an optimal solution of $\mathbf{rlvm}(R,N,e_2,c')$
8. compute μ_X, μ_Y, σ_X, σ_Y under $\{(d_i,r_i)\}$
9. compute $\bar{X}^{\approx}_{\text{FEC}}$
10. store $\{(d_i,r_i)\}$ if it is the best so far
11. **end**
12. **end**
13. **return** $\{(d_i,r_i)\}$ that gives the minimum $\bar{X}^{\approx}_{\text{FEC}}$

Fig. 5. Traffic allocation problem under capacity constraints.

3) Packet Spreading Algorithm: In Section IV-B.1 and Section IV-B.2, we derive the traffic allocation (i.e., $\{(d_i,r_i)\}$ for each path i). For the same allocation, different ways of assigning packets onto paths can result in different observed loss burstiness. The burstiness in the observed loss increases with the burstiness in traffic. So we should try to spread the packets allocated on the same path as evenly as possible. This reduces burstiness in experienced packet losses, and enhances effectiveness of FEC. To achieve this goal, we develop a simple packet spreading algorithm, as described in Appendix . For a given allocation, $\{(d_i,r_i)\}$, it determines exactly which packets in an FEC group should be allocated onto which paths so that the final loss rate is minimized.

C. Estimating Parameters for the Loss Model

The effectiveness of the above traffic allocation scheme depends on the accuracy of the loss model estimation. In our evaluation, we use extended Gilbert loss model, and estimate its transition matrix (in Equation 2), as shown in [32].

$$p_{01} = (\sum_{i=1}^{n-1} m_i)/m_0$$

$$p_{(k-1)k} = (\sum_{i=k}^{n-1} m_i)/(\sum_{i=k-1}^{n-1} m_i)$$

where m_i denotes the number of loss bursts with length i, where $i = 1, 2, ..., n-1$ and m_0 denote the number of delivered packets.

Since network path properties change over time, we predict future network performance using previous intervals.

D. FEC redundancy adaptation

We derive the allocation scheme given the number of redundant packets R per FEC group. In practice, we can do greedy search to find the minmum R value which can satisfy target loss rate (e.g.,1e-6). As a result, we can reduce the bandwidth overhead.

E. Handling Shared Congestion

In the previous discussion, we consider loss rates on the physical paths are independent. In practice, we may have some paths that share a common bottleneck. In this case, the loss rates on these paths are highly correlated. To handle such cases, we can apply an existing technique to detect shared congestion. A number of techniques have been proposed for this purpose, such as cross-correlation-based approach [31], entropy-based approach [22], and wavelet-based approach [23]. We then treat the set of paths that share congestion as one path, and apply our traffic allocation scheme to the merged paths.

V. Performance Evaluation

In this section, we first introduce our evaluation methodology and then describe evaluation results.

A. Evaluation Methodology

We evaluate the performance of SmartTunnels using the following three ways: (i) Internet trace-driven simulation, (ii) ns-2 simulation, and (iii) experiment on PlanetLab. These three evaluation methods are complementary to each other. Trace-driven evaluation allows us to extensively evaluate the performance of SmartTunnels under realistic Internet performance characteristics; ns-2 simulation allows us to study the interactions between multiple tunnels in a controlled environment; and real experiment allows us to understand the benefit and overhead of SmartTunnels in a real network.

We compare the following traffic allocation schemes:

- **SmartTunnel:** This is the algorithm we describe in Section IV.
- **Markowitz numeric (MkwNu):** This is the algorithm proposed in [19]. It maximizes the Sharpe-Ratio [3] by solving a series of quadratic optimization problems.
- **Round robin (RR):** Traffic is assigned to multiple physical paths in a round robin fashion.
- **Greedy:** Traffic is assigned to the path that has the lowest loss rate. When multiple paths experience the same loss rate, one path is selected randomly among them.

B. Trace-driven Simulation

We collect Internet traces by sending 16-byte ICMP echo packets from 57 hosts on PlanetLab to 55 popular Web sites, selected from the 100 popular websites listed at [1]. We run zing and tcpdump concurrently on each PlanetLab host. To capture bursty loss behavior, zing is modified to generate ICMP echo packets with an inter-packet arrival of 2 ms. Tcpdump is used to captured ICMP echo-reply packets. In order to avoid PlanetLab hosts to drop packets when the probing traffic are too bursty. We introduce 1 second idle time every 1 second bursty traffic. Each measurement experiment lasts at least 800 seconds. Figure 6 shows the CDF of raw loss rates. About 78.5% of paths have loss rates below 2%. The mean loss rates of these paths is 0.0175.

Each trace is divided into 20 intervals, so each interval is a 40-second trace. We apply different traffic allocation schemes on each 40-second interval. For all the evaluation, we use FEC group size of 40 packets (including data and redundancy packets), and adapt traffic allocation every interval. Two kinds of evaluation results are shown. One is oracle result in which we assume current network path performance can be known from Oracle and there is no prediction errors. The other one is prediction result in which current network path performance is predicted from previous intevals. Table II shows probabilities of SmartTunnel to achieve loss free reliability. When there is no prediction error, SmartTunnel can achieve loss free with probability up to 0.9991 if it uses 6 paths. It can also achieve around loss free with probability 0.94 even with only 2 paths. If we consider prediction errors, SmartTunnel can also achieve loss free with probability from 0.85 to 0.93.

	2Path	3Path	4Path	6Path
Oracle	0.94	0.9852	0.996	0.9991
Prediction	0.85	0.88	0.91	0.9267

TABLE II

Probabilities of SmartTunnel to achieve four 9s reliability

1) Oracle results: We compare different traffic allocation schemes by varying the number of available physical paths, redundancy level used in FEC, and quality of the paths. For each experiment configuration (e.g., a fixed number of paths to the same website and combination of path property), we conduct 20 random runs (i.e., selecting 20 different combinations of traces used for evaluation), and report the summary statistics from these runs.

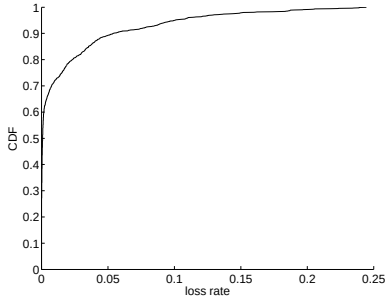


Fig. 6. Path raw loss rates

To systematically study the performance, we categorize results into different scenarios based on the number of low

loss paths selected. Low loss paths are paths whose loss rates are below 2%. This classification is used in [39]. Results are shown in Table III, Table IV and Table V. Let K denote the number of redundant packets and G denote the number of low loss paths chosen. We make the following observations. First, in all cases SmartTunel is the best performing algorithm. Second, when all paths are low loss paths, SmartTunel can achieve loss free reliability for 94.47%-99.81% of time. Third, SmartTunnel uses high loss paths much better than other algorithms. For example, when all paths are high loss paths, SmartTunnel can achieve loss free reliability for around 6%-35% more time intervals compared to other algorithm. It is interesting that when high loss paths are selected, Greedy algorithm is almost the second best algorithm. Fourth, in those cases with three physical paths selected, the difference between various traffic allocation schemes becomes smaller when the number of redundancy packets increases. This suggests that the choice of traffic allocation is more important when there is limited bandwidth.

	4Path, K = 7	6Path, K = 7
SmartTunnel	96.99%	98.65%
MkwNU	94.73 %	97.13%
RR	75.45%	90.16%
Greedy	94.73%	97.19%

TABLE V

PERCENTAGES OF INTERVALS TO ACHIEVE LOSS FREE RELIABILITY UNDER 4 PATHS OR 6 PATHS RANDOMLY CHOSEN FROM THE TRACES

Table VI and Table VII show mean loss rates of SmartTunnel with different number of low loss paths selected and different number of redundant packets. We can derive the expected loss rate of N-Path SmartTunnel as follow.

Let P denote the probability a selected path has loss rate lower than 2% and $L(G = i, K = j)$ as the mean loss rate of SmartTunnel when $G = i$ and $K = j$. Then we can compute the expected loss rate $L_{N,K=j}$ of N-Path SmartTunnel.

$$\begin{aligned} L_{N,K=j} &= \sum_{i=0}^{N} L(G=i,K=j) * P_r(G=i) \\ &= \sum_{i=0}^{N} L(G=i,K=j) * C_N^i * P^i * (1-P)^{N-i} \end{aligned}$$

For example, when N = 2, $L_{2,K=7} = 8{\times}10^{-4}$. We find out that the expected loss rate of 3-Path SmartTunnel can be as small as 5×10^{-5} when K is 14.

	K=7	K=10	K=14
G = 0	0.0101	0.0071	0.0054
G = 1	0.0009	0.0004	0.0003
G = 2	0.0001	0	0

TABLE VI

MEAN LOSS RATES OF SMARTTUNNEL USING 2 PATHS

2) Predictability of Path Properties: For a traffic allocation to work well, we need to be able to predict future network path performance. In this section, we study the predictability of loss rates.

We apply Fisher exact probability test [12] to compare the predicted and actual loss transition matrices. The Fisher test

	K=7	K=10	K=14
G = 0	0.0067	0.0046	0.0031
G = 1	0.0003	0.0002	0.0001
G = 2	0.0001	0	0
G = 3	0	0	0

TABLE VII

MEAN LOSS RATES OF SMARTTUNNEL USING 3 PATHS

is a statistical significance test for analyzing categorical data where sample sizes are small. Let $PN[RL \geq i]$ denote the number of predicted loss runs whose lengths equal or exceed i, and $AN[RL \geq i]$ denote the number of actual loss runs whose lengths equal or exceed i. We use Fisher test to test the hypothesis that $PN[RL \geq i]$ out of $PN[RL \geq i-1]$ is consistent with $AN[RL \geq i]$ out of $AN[RL \geq i-1]$. We compute P-value in the Fisher test. It measures statistical significance. When P-value is above 0.01, it indicates that there is *no* evidence that the predicted data is *not* consistent with the actual data.

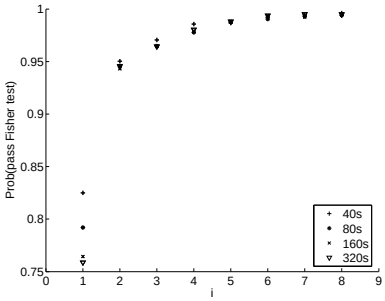


Fig. 7. Fraction of samples that pass the Fisher test for different loss run length values i.

Figure 7 plots the fraction of cases where the hypothesis passes the Fisher test. We use different length of history traces to predict current network performance. As it shows, test results are not sensitive to the length of history traces. The main difference is when $i = 1$. In practice, we prefer to use longer history trace to do the prediction because it is more stable. In the following evaluation, we use the loss transition matrix of previous 320 second trace (8 intervals) as the prediction of the current interval. 76% time intervals pass the Fisher test when $i = 1$, and over 94% intervals passes Fisher test for a larger i. This suggests that the temporal dependency between consecutive loss events is more predictable than the transition from no loss to loss. Overall we observe that the predicted loss transitions match reasonably well with the actual values. This suggests that it is possible to apply traffic allocation based on past performance.

3) Trace-driven evaluation results: Table VIII, Table IX and Talbe X show trace-driven results with prediction. SmartTunnel out-performs the other schemes in all senarios except one in which there are three low loss rate paths and K is 14. This is because the performance of SmartTunnel reduces from 99.81 % to 97.64% due to prediction errors while prediction errors do not affect the performance of round robin.

C. NS-2 Simulation

In this section, we study the interactions between multiple SmartTunnels using ns-2 simulations. Figure 8 shows the

	K = 7			K = 10			K = 14		
	G=2	G=1	G=0	G=2	G=1	G=0	G=2	G=1	G=0
SmartTunnel	94.47%	79.45%	31.12%	98.88%	89.62%	52.95%	99.09%	93.81%	63.30%
MkwNU	87.91 %	75.40%	25.51%	95.36%	84.58%	40.84%	98.20%	90.93%	53.40%
RR	82.82%	48.09%	16.73%	91.52%	64.51%	40.11%	92.04%	74.80%	49.53%
Greedy	88.86%	77.71%	27.85%	94.79%	85.87%	44.53%	97.80%	90.60%	53.21%

TABLE III

PERCENTAGES OF INTERVALS TO ACHIEVE LOSS FREE RELIABILITY UNDER 2 PATHS RANDOMLY CHOSEN FROM THE TRACES

	K = 7				K = 10				K = 14			
	G=3	G=2	G=1	G=0	G=3	G=2	G=1	G=0	G=3	G=2	G=1	G=0
SmartTunnel	98.71 %	96.80 %	88.20 %	48.82 %	99.61 %	98.82 %	93.58 %	65.86 %	99.81 %	99.62 %	97.25 %	77.84 %
MkwNU	96.35 %	93.84 %	84.96 %	33.09 %	98.80 %	97.26 %	90.76 %	47.73 %	99.67 %	99.02 %	94.78 %	64.79 %
RR	90.53 %	64.51 %	35.73 %	14.77 %	93.84 %	79.05 %	55.22 %	33.27 %	99.78 %	96.93 %	82.88 %	61.60 %
Greedy	96.00 %	94.49 %	86.08 %	42.82 %	97.73 %	96.60 %	90.43 %	53.23 %	98.92 %	98.44 %	93.94 %	61.08 %

TABLE IV

PERCENTAGES OF INTERVALS TO ACHIEVE LOSS FREE RELIABILITY UNDER 3 PATHS RANDOMLY CHOSEN FROM THE TRACES

	K = 7			K = 10			K = 14		
	G=2	G=1	G=0	G=2	G=1	G=0	G=2	G=1	G=0
SmartTunnel	90.46 %	77.25%	27.76%	96.18%	87.12%	50.21%	97.35%	91.25%	59.25%
MkwNU	83.66 %	73.53%	20.47%	91.27%	83.30%	37.47%	96.22%	89.25%	49.81%
RR	82.76%	48.19%	17.38%	91.36%	64.41%	39.89%	91.95%	74.45%	48.87%
Greedy	84.85%	76.18%	27.10%	90.82%	85.07%	42.63%	95.15%	89.22%	52.26%

TABLE VIII

PERCENTAGES OF INTERVALS TO ACHIEVE LOSS FREE RELIABILITY UNDER 2 PATHS RANDOMLY CHOSEN FROM THE TRACES

	K = 7				K = 10				K = 14			
	G=3	G=2	G=1	G=0	G=3	G=2	G=1	G=0	G=3	G=2	G=1	G=0
SmartTunnel	94.42 %	92.31 %	85.46 %	41.77 %	96.39 %	95.39 %	90.49 %	58.18 %	97.64 %	97.06 %	94.14 %	71.08 %
MkwNU	91.32 %	89.31 %	82.61 %	25.77 %	95.03 %	93.61 %	88.79 %	40.68 %	97.22 %	96.28 %	93.31 %	59.67 %
RR	90.52 %	64.48 %	35.43 %	15.27 %	93.84 %	78.54 %	54.90 %	33.05 %	99.76 %	96.92 %	82.46 %	61.31 %
Greedy	91.37 %	90.62 %	84.69 %	39.36 %	93.82 %	93.37 %	89.03 %	49.23 %	96.09 %	95.56 %	92.42 %	57.14 %

TABLE IX

PERCENTAGES OF INTERVALS TO ACHIEVE LOSS FREE RELIABILITY UNDER 3 PATHS RANDOMLY CHOSEN FROM THE TRACES

	4Path, K = 7	6Path, K = 7
SmartTunnel	92.43%	94.49%
MkwNU	90.01 %	93.82%
RR	75.51%	90.73%
Greedy	90.64%	93.06%

TABLE X

PERCENTAGES OF INTERVALS TO ACHIEVE LOSS FREE RELIABILITY UNDER 4 PATHS OR 6 PATHS RANDOMLY CHOSEN FROM THE TRACES

network topology used in the evaluation. Both senders use 3 redundancy packets per FEC group, where each FEC group consists of 13 packets in total. Figure 9 shows the evolution of loss rates experienced by 2 SmartTunnels that share physical paths. To stress test, we initialize their allocation to both use path A-B-D. Due to poor initial allocation, both tunnels initially experience high loss rates before and after FEC. This also highlights the importance of appropriate traffic allocation on end-to-end reliability. Then the two tunnels continuously adapt their traffic allocation according to their monitored performance every time interval. At interval 7, both tunnels converge to low loss rates before FEC, and close to 0 loss rate after applying FEC. Figure 10 further plots the data allocation of two tunnels on these paths. As we can see, they converge to an even share of network resources, both allocating 5 data packets on two paths. Similar fair allocation is achieved for redundancy packets (not shown in the interest of space). Overall both reliability and fairness are achieved.

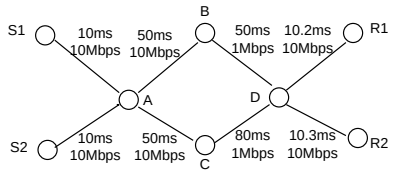
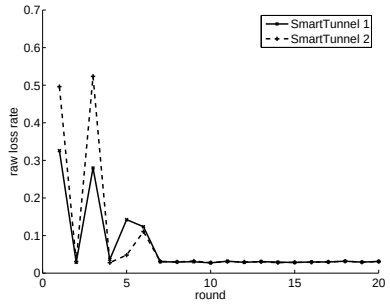


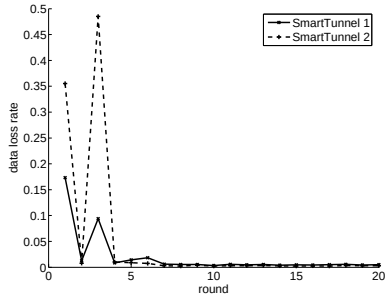
Fig. 8. Network topology used in n2-simulation. Two tunnels S1-R1 and S2-R2 share two physical paths. S1 sends 0.7 Mbps CBR traffic, and S2 sends 0.5 Mbps CBR traffic. Low-rate Pareto traffic is introduced as background traffic on links BD and CD. In addition, links BD and CD use Gilbert-loss models to drop traffic, where the loss transition matrix at B is [0.985 0.015; 0.45 0.55], and that at C is [0.99 0.01; 0.35 0.65].

D. Experiments on PlanetLab

We implement SmartTunnel using click [24] on PlanetLab. Our implementation is around 2500 lines of C++ code. Click is a flexible and configurable router architecture. It consists of packet processing modules, called *elements*. Existing elements in click include queueing, scheduling, and interfacing with network devices. We add the following elements to click to provide SmartTunnel functionalities: (i) monitors at both sender and receiver to cooperatively monitor network performance using either active probing or passive probing, (ii) a traffic distributor that stripes traffic according to the controller's specification, (iii) an encoder and decoder that apply FEC

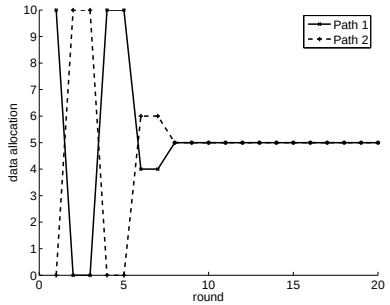


(a) Loss rates before applying FEC.

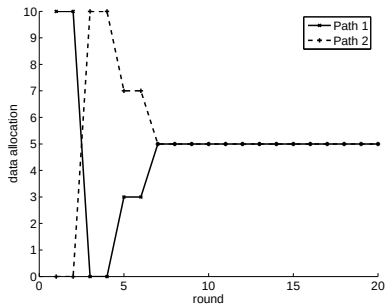


(b) Loss rates after applying FEC.

Fig. 9. Loss rates dynamics of the 2 competing SmartTunnels in ns-2.



(a) SmartTunnel S1-R1 data allocation on 2 paths.



(b) SmartTunnel S2-R2 data allocation on 2 paths.

Fig. 10. Data allocation dynamics of the 2 competing SmartTunnels in ns-2.

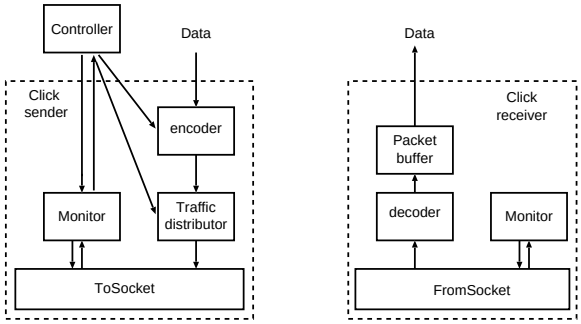


Fig. 11. SmartTunnel implementation in click.

encoding and decoding using the public FEC software [28] and add/remove SmartTunnel header, and (iv) packet buffering. Figure 11 shows the diagram of different components in our implementation. ToSocket and FromSocket in the figure are the existing elements in click to provide sending and receiving functionalities, and the other elements in the figures are what we implement. At the sender side, the controller, implemented outside the click, coordinates with different click elements by specifying monitoring instructions, an FEC coding scheme, a traffic allocation scheme. The receiver logic is much simpler: on the data path, it decodes and buffers packets; on the control path, the monitor responses to active probes from the sender, and also periodically sends back performance information for the paths that carry traffic.

Figure 12 plots the encoding time for different numbers of redundancy packets for a 40-packet FEC group, where each packet has 1024 bytes. The encoding time includes time to generate redundancy packets and time to add SmartTunnel headers. The number is measured when click is running at user-level mode on 3.2 GHz desktop. As we would expect, the encoding time increases with the number of redundancy packets we need to generate. The data rate we can support varies from 0.56 Gbps using 20 redundancy packets to 1.8 Gbps using 2 redundancy packets. Such data rate is sufficient for most Internet gateways. Moreover, running at a kernel mode can further increase the rate.

In the experiment, we construct a SmartTunnel on top of three overlay paths between two hosts. The SmartTunnel uses 10 redundancy packets per FEC group (each group with 40 packets). Figure 13 (a) shows 15 minutes time series of path loss rates before FEC and SmartTunnel loss rate after FEC. As we can see, these three paths experience substantial loss rates. Figure 13 (b) further shows the traffic allocation on these paths over time. Initially all traffic are allocated on Path I. FEC does not work well because the loss is too bursty. Every one minute, SmartTunnel computes the new traffic allocation based on its observed performance. SmartTunnel is pretty 'smart'. Among these three paths, Path III has lowest loss rate and smallest loss variance. SmartTunnel put around 60 % traffic on it. Instead of putting all traffic on the best path, SmartTunnel also uses worse paths (i.e, put around 13% traffic on Path I). After 4 minutes, SmartTunnel achieves almost full reliability.

E. Summary

To summarize, in this section we evaluate the performance of SmartTunnels using trace-driven simulation, ns-2 simulation, and PlanetLab experiments. Our results show that SmartTunnel can achieve high reliability over a diverse set of scenarios. Moreover our initial study of interactions between multiple smart tunnels suggests that they can co-exist well. We plan to further investigate their interactions more thoroughly in the future.

VI. Conclusion

In this paper, we propose *SmartTunnel*, an end-to-end approach to achieving high reliability. It applies FEC and allocates traffic onto multiple physical paths to minimize loss rates

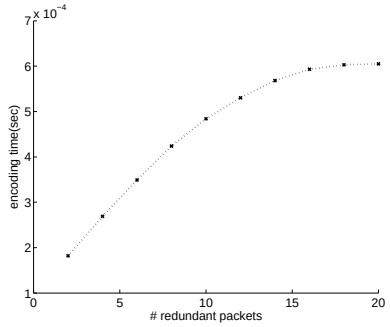
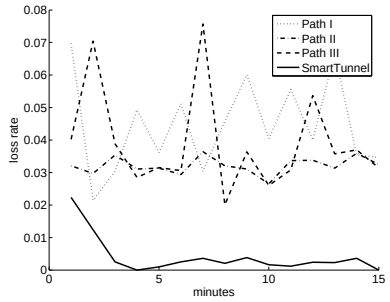
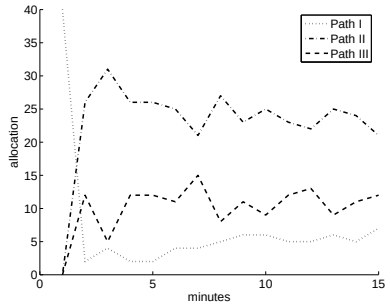


Fig. 12. Encoding time: it includes generating redundancy packets and adding SmartTunnel headers. The FEC group size is 40 packets and 1024 bytes.



(a) Evolution of loss rates before and after FEC.



(b) Evolution of traffic allocation across three paths.

Fig. 13. Time series of loss rates and traffic allocation on three paths spanned by SmartTunnel on PlanetLab.

under realistic Internet loss models. Using extensive simulation and real implementation, we demonstrate that SmartTunnel is effective in achieving high reliability.

As part of our future work, we are interested in applying SmartTunnel to wireless networks. An increasing number of wireless devices have multiple interfaces. Effectively utilizing these interfaces simultaneously has the potential to significantly improve reliability in wireless networks. Wireless link loss characteristics differ significantly from those of wireline links. For example, wireless links are more variable and unpredictable. We plan to study the performance of SmartTunnel in such a highly dynamic network. In addition, network paths involving wireless links may experience extended outage periods (e.g., due to mobility or environmental changes). Such outages may span multiple FEC groups, and significantly reduce the effectiveness of FEC. We are interested in extending SmartTunnel to handle such outages in addition to bursty losses.

REFERENCES

[1] 100 top web sites. `http://www.100hotsites.com`.

[2] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman. A measurement-based analysis of multihoming. In *Proceedings of ACM SIGCOMM '03*, Karlsruhe, Germany, Aug. 2003.

[3] G. J. Alexander and J. C. Francis. Portfolio analysis. 1986.

[4] P. Barford and J. Sommers. Comparing probe- and router-based methods for measuring packet loss. In *In IEEE Internet Computing - Special issue on Measuring the Internet*, Sept/Oct 2004.

[5] J.-C. Bolot. End-to-end packet delay and loss behavior in the Internet. In *Proceedings of ACM SIGCOMM '93*, San Francisco, CA, Sept. 1993.

[6] J.-C. Bolot, S. Fosse-Parisis, , and D. Towsley. Adaptive FEC-based error control for interactive audio in the Internet. In *Proceedings of IEEE INFOCOM '99*, New York, NY, Mar. 1999.

[7] J. W. Byers, M. Luby, and M. Mitzenmacher. A digital fountain approach to asynchronous reliable multicast. *IEEE JSAC, Special Issue on Network Support for Multicast Communication*, 2002.

[8] Z. Cao, Z. Wang, and E. Zegura. Rainbow fair queueing: Fair bandwidth sharing without per-flow state. In *Proceedings of IEEE INFOCOM '00*, Tel Aviv, Israel, Mar. 2000.

[9] M. Dahlin, B. Chandra, L. Gao, and A. Nayate. End-to-end WAN service availability. Apr. 2003.

[10] Z. Duan, Z. Zhang, and Y. T. Hou. Service Overlay Networks: SLA, QoS and bandwidth provisioning. In *Proc. of ICNP*, 2002.

[11] E. O. Elliott. Estimates of error rates for codes on burst-error channels. *Bell System Technical Journal*, page 1977, Sept. 1963.

[12] Fisher exact probability test. `http://home.clara.net/sisa/fishrhlp.htm`.

[13] E. N. Gilbert. Capacity of a burst-noise channel. *Bell System Technical Journal*, 39:1253–1266, Sept. 1960.

[14] D. K. Goldenberg, L. Qiu, H. Xie, Y. R. Yang, and Y. Zhang. Optimizing cost and performance for multihoming. In *Proceedings of ACM SIGCOMM '04*, Portlan, Oregon, Aug. 2004.

[15] K. P. Gummadi and H. V. Madhyastha. Improving the reliability of internet paths with one-hop source routing. In *Proc. of OSDI*, Oct. 2004.

[16] F. Guo, J. Chen, W. Li, and T. Chiueh. Experiences in building a multihoming load balancing system. In *Proceedings of IEEE INFOCOM '04*, Hong Kong, China, Apr. 2004.

[17] D. Harrison, S. Kalyanaraman, and S. Ramakrishnan. Overlay bandwidth services: Basic framework and an edge-to-edge closed-loop building block. In *Preprint*, Jan. 2001.

[18] V. Jacobson. Congestion avoidance and control. In *Proc. of ACM SIGCOMM*, pages 314–329, 1988. `http://doi.acm.org/10.1145/52324.52356`.

[19] S. Jain, M. Demmer, R. Patra, and K. Fall. Using redundancy to cope with failures in a delay tolerant network. In *Proc. of ACM SIGCOMM*, Aug. 2005.

[20] W. Jiang and H. Schulzrinne. Modeling of packet loss and delay and their effect on real-time multimedia service quality. In *Proceedings of the 10th International Workshop on Network and Operating System Support for Digital Audio*, June 2000.

[21] W. Jiang and H. Schulzrinne. Assessment of voip service availability in the current internet. In *Proc. of PAM*, Apr. 2003.

[22] D. Katabi, I. Bazzi, and X. Yang. A passive approach for detecting shared bottlenecks. In *Proc. of 10th IEEE International Conference on Computer Communications and Networks*, Oct. 2001.

[23] M. S. Kim, T. Kim, Y. Shin, S. Lam, and E. Powers. A wavelet-based approach to detect shared congestion. In *Proc. of ACM SIGCOMM*, Aug. 2004.

[24] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, pages 263–297, Aug 2000.

[25] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman. Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, 2001.

[26] M. Mitzenmatcher. Digital fountains: A survey and look forward. In *Information Theory Workshop*, 2004.

[27] V. Paxson. End-to-end routing behavior in the internet. In *IEEE/ACM Transactions on Networking*, 1997.

[28] L. Rizzo. Forward error correction (FEC) software. `http://info.iet.unipi.it/~luigi/vdm98/vdm980702.tgz`.

[29] L. Rizzo. Effective erasure codes for reliable computer communication protocols. *ACM Computer Communication Review*, Ap. 1997.

[30] RON measurement data. `http://nms.lcs.mit.edu/ron/data/`.

[31] D. Rubenstein, J. Kurose, and D. Towsley. Detecting shared congestion of flows via end-to-end measurement. In *IEEE/ACM Transactions on Networking*, pages 381–395, Jun. 2002.

[32] H. Sanneck, G. Carle, and R. Koodli. A framework model for packet loss metrics based on loss run lengths. In *SPIE/ACM SIGMM Multimedia Computing and Networking Conference*, Jan. 2000.
[33] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson. The end-to-end effects of internet path selection,. In *Proc. of ACM SIGCOMM*, Aug. 1999.
[34] J. Sommers, P. Barford, N. Duffield, and A. Ron. Improving accuracy in end-to-end packet loss measurement. In *Proc. of ACM SIGCOMM*, Aug. 2005.
[35] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. OverQoS: An Overlay based Architecture for Enhancing Internet QoS. In *Proc. of NSDI*, Mar. 2004.
[36] H. Tangmunarunkit, R. Govindan, and S. Shenker. Internet path inflation due to policy routing. In *Proc. of SPIE ITCOM*, Aug. 2001.
[37] E. W. Weisstein. Erf. From *MathWorld* – A Wolfram Web Resource. `http://mathworld.wolfram.com/Erf.html`.
[38] M. Yajnik, S. Moon, J. Kurose, and D. Towsley. Measurement and modelling of the temporal dependence in packet loss. In *Proc. of INFOCOM 99*, Mar. 1999.
[39] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the constancy of Internet path properties. In *Proceedings of Internet Measurement Workshop 2001*, San Francisco, CA, Nov. 2001.
[40] M. Zorzi and R. R. Rao. On the statistics of block errors in bursty channels. *IEEE Transactions on Communications*, 45(6):660–667, June 1997.

APPENDIX

Consider an FEC group that consists of D data packets followed by R redundant packets. Suppose we allocate d data packets and r redundant packets to an individual path. Let random variables x and y denote the number of lost data packets and the number of lost redundancy packets on this path, respectively. Our goal here is to derive statistics for packet loss, including mean ($E[x]$, $E[y]$), variance ($V[x]$, $V[y]$), and covariance ($Cov[x, y]$).

We will consider the General Markov Model for packet loss. So our results generalize the analysis by Zorzi *et al.* [40], which is based on the 2-state Gilbert-Elliott Model. Suppose the loss model for the data packets is a General Markov Model with state set S, loss probability vector $L = [\ell_i]$, and transition matrix $P = [p_{ij}]$. The loss model for the redundant packets has the same S and L but a different transition matrix $P' = [p'_{ij}]$. Note that data packets and redundant packets have different transition matrices because they may have different sending rates. In Appendix we will describe how the sending rate determines the transition matrix.

Let $\Phi_{ij}(\underline{k}|n)$ be the probability for "k out of n data packets are lost and the initial state $s_0 = i$ and end state $s_n = j$". Clearly, $\Phi_{ij}(\underline{k}|n)$ satisfies

$$\Phi_{ij}(\underline{k}|n) = \sum_{m\in S} [\Phi_{im}(\underline{k-1}|n-1)\ell_m + \Phi_{im}(\underline{k}|n-1)(1-\ell_m)]\, p_{mj} \tag{8}$$

Here $\Phi_{im}(\underline{k-1}|n\text{–}1)\ell_m p_{mj}$ gives the probability for "k–1 out of the first n–1 data packets are lost and the initial state $s_0 = i$ and end state $s_n = j$ and penultimate state $s_{n-1} = m$ and the n-th packet is lost". Similarly, $\Phi_{im}(\underline{k}|n-1)(1-\ell_m)p_{mj}$ gives the probability for "k out of the first $n-1$ packets are lost and the initial state $s_0 = i$ and end state $s_n = j$ and penultimate state $s_{n-1} = m$ and the n-th packet is not lost".

In addition, the following initial condition holds:

$$\Phi_{ij}(\underline{k}|n) = 0, \quad \text{if } k < 0 \text{ or } n < 0 \tag{9}$$

$$\Phi_{ij}(\underline{0}|0) = \begin{cases} 0, & \text{if } i \neq j \\ \pi_i, & \text{if } i = j \end{cases} \tag{10}$$

where π_i is the stationary probability for the General Markov Model to stay in state i, which can be computed from P.

Combining (8), (9) and (10), we can compute $\Phi_{ij}(\underline{k}|n)$ for all $n \leq d$ via dynamic programming.

Similarly, define $\Phi'_{ij}(\underline{k}|n)$ as the probability for "k out of n redundant packets are lost and the initial state $s_0 = i$ and end state $s_n = j$". We can compute $\Phi'_{ij}(\underline{k}|n)$ for all $n \leq r$ again via dynamic programming. We then have

$$E[x] = \sum_{i,j\in S} \sum_{k=1}^{d} k\Phi_{ij}(\underline{k}|d)$$

$$V[x] = -E[x]^2 + E[x^2] = -E[x]^2 + \sum_{i,j\in S} \sum_{k=1}^{d} k^2\Phi_{ij}(\underline{k}|d)$$

$$E[y] = \sum_{i,j\in S} \sum_{k=1}^{r} k\Phi'_{ij}(\underline{k}|r)$$

$$V[y] = -E[y]^2 + E[y^2] = -E[y]^2 + \sum_{i,j\in S} \sum_{k=1}^{r} k^2\Phi'_{ij}(\underline{k}|r)$$

$$\begin{aligned} Cov[x, y] &= -E[x]E[y] + E[x\,y] \\ &= -E[x]E[y] + \sum_{i,m,j\in S} \sum_{k_1=1}^{d} \sum_{k_2=1}^{r} k_1 k_2 \Phi_{im}(\underline{k_1}|d)\Phi'_{mj}(\underline{k_2}|r) \end{aligned}$$

The sending rate can have a significant impact on the burstiness of packet loss. Specifically, let P_D be the transition matrix for the General Markov Loss Model when we send D data packets along a given path (at the average data arrival rate of the SmartTunnel). Let P_d be the corresponding transition matrix for the General Markov Loss Model when we only send d data packets along the same path. It is easy to see that if we send these d data packets in an evenly spaced fashion, the sending rate is reduced by a factor of D/d. As a result, we have

$$P_d = (P_D)^{D/d} \tag{11}$$

which in general is less bursty than P_D itself.

Note that (11) assumes that d divides D. In a more general case, let $f_0 = \lceil D/d \rceil$, $f_1 = \lfloor D/d \rfloor$, $n_0 = D \mod d$, $n_1 = d - n_0$. Among the d data packets, n_0 will have transition matrix $(P_D)^{f_0}$ and n_1 will have transition matrix $(P_D)^{f_1}$. So we can approximate P_d as their weighted average

$$P_d = \frac{n_0}{d}(P_D)^{f_0} + \frac{n_1}{d}(P_D)^{f_1} \tag{12}$$

Similar results hold for the redundant packets.

Therefore, given an allocation $\{(d_i, r_i)\}$, we should try to spread the packets allocated on the same path as evenly as possible. To achieve this goal, we develop a simple packet spreading algorithm, shown in Figure 14. The algorithm is used to offline determine which packets should be allocated onto which paths within an FEC group. This offline computation incurs low overhead, and is one-time cost for a given FEC group size. Then the derived packet spreading strategy is applied to incoming traffic continuously.

As shown in Figure 14, the algorithm allocates traffic using a credit-based scheme. Each path is associated with a credit. The path with the largest number of credits is selected to transmit the next packet. The credit of a path i is updated as

```
1.  inc[i] = d_i/D, ∀i = 1, ··· , N;
2.  credit[i] = 0, ∀i = 1, ··· , N;
3.  for t = 1 to D
4.      credit[i]+= inc[i], ∀i = 1, ··· , N;
5.      i_max = arg max_i{credit[i]};
6.      t-th data packet is sent by path i_max;
6.      credit[i_max] −−;
8.  end
```

Fig. 14. A simple packet spreading algorithm.

follows. Each time a new data packet is transmitted (regardless of which path transmits it), path i earns d_i/D credits (so after seeing all D data packets, it earns d_i credits). Each time path i transmits a packet, it consumes 1 credit. We apply the same strategy for sending redundancy packets but change the credit increase rate to r_i/R.