

Peer-to-Peer Error Recovery for Hybrid Satellite-Terrestrial Networks

Eric Weigle^{†‡}, Matti Hiltunen[‡], Rick Schlichting[‡], Vinay A. Vaishampayan[‡], and Andrew A. Chien^{†◇*}

{eweigle, achien}@ucsd.edu, {hiltunen, rick, vinay}@research.att.com

[†]Computer Science and Engineering
University of California, San Diego

[‡]AT&T Labs - Research

[◇] Intel Research

Abstract

Media companies (and other organizations with large amounts of digital content) require prompt broadcast of extremely large files from a single source to a collection of geographically dispersed destinations. Due to the high cost of terrestrial networks of sufficient bandwidth, satellite networks are commonly used for such transfers. However, current satellite transfers rely on expensive error correction via forward error correction and whole-file retransmission.

This paper presents a new, hybrid solution combining the advantages of satellite and terrestrial networks to provide cost-effective reliable file transfer. Specifically, we propose a new peer-to-peer scheme exploiting fast terrestrial networks and multiple receivers to recover from high loss rates (5% or more) in near real-time (latency <400ms). This solution is efficient, robust under variable packet loss and connectivity, user tunable, scales well, and doubles bandwidth compared to existing approaches. The system has been validated via extensive simulations using a terrestrial network based on the AT&T common backbone core network.

Keywords: hybrid network, peer-to-peer, broadcast, content distribution

1. Introduction

Demand for transfer of extremely large data files between widely-dispersed locations is increasing rapidly. Broadcast and media companies need to transfer HDTV files to affiliates for end-user broadcast or collaborative editing, computational scientists need to distribute scientific data across inter-continental grids, and enterprises use off-site data backups for disaster recovery.

The PlanetLab Grand Challenge with the Public Broadcasting Service (PBS) [1, 5] provides just one concrete example. Their requirements include the need to transfer up to 450GB/day from PBS headquarters to approximately 180 affiliates across North America. These sites have varying

degrees of terrestrial connectivity, typically in the range of DSL to T1 links (128Kbps-1.5 Mbps). However, such access links are too slow to satisfy the transfer requirements. Faster links can be acquired, but are expensive compared to satellite transponders which can receive broadcast data at higher speeds (e.g., 40 Mbps [5]).

While satellites avoid terrestrial bottlenecks, they suffer from packet corruption/loss due to weather, insufficient power, crashes, etc. For reliable transfers, they rely on forward error correction (FEC) utilizing up to 50% of the channel for redundant information, and fall back on whole file retransmission if errors are uncorrectable via FEC. Such an approach induces high overhead at even low loss rates. In comparison, terrestrial networks can efficiently retransmit individual blocks. We would like the best of both worlds.

This paper describes a new peer-to-peer broadcast/recovery scheme for reliably transferring large files from a single source to a number of geographically dispersed destinations. It takes advantage of *both* satellite and terrestrial networks: broadcast via satellite, recover via the terrestrial network. Receivers exchange data to detect and correct errors in a peer-to-peer fashion. Broadcast/recovery are pipelined and overlap for most of a transmission.

Our approach has the desirable properties that it: (1) corrects for large error/loss rates (5% or more); (2) scales in the file size (to gigabytes) and number of nodes (to 1000s+); (3) provides low block latency (<400ms); (4) is efficient (low network, memory overhead, globally minimizes cost function); (5) is fair (different peers share equally); (6) and it is user tunable (user may control peering behavior).

We validate claims 1-6 using “real-world” performance metrics on realistic networks, including scenarios in which the terrestrial network is based on the AT&T common backbone core network [3].

This paper is structured as follows. First, we discuss the problem and target environment in more detail. Then we present our solution, including the algorithms involved and their implementation. We then evaluate the implementation in depth using the ns-2 simulator, before concluding with a brief discussion and related work.

2. The Problem

The hybrid broadcast problem, as found in our target applications, is defined by the following characteristics:

*This work was part of the collaboration between AT&T Labs-Research and the UCSD Center for Networked Systems, and was done primarily while Eric Weigle was an intern at AT&T Labs-Research. Weigle and Chien were also supported in part by the National Science Foundation under awards NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from the UCSD Center for Networked Systems, BigBangwidth, and Fujitsu is also gratefully acknowledged.

1. A single *source node*— the node with data to send in the beginning— is connected to *destination nodes* via both a satellite link and a terrestrial network.
2. High data rate— on the order of 100-450 GB/day.
3. Continuous data transmission— little or no *inter-transmission* “free” time to recover from problems.
4. Relatively low *intra-transmission* delay tolerance— data must be received on the order of a few seconds after it is initially sent.
5. Loss varies between uplink and downlinks— loss tending to be higher on the downlinks.

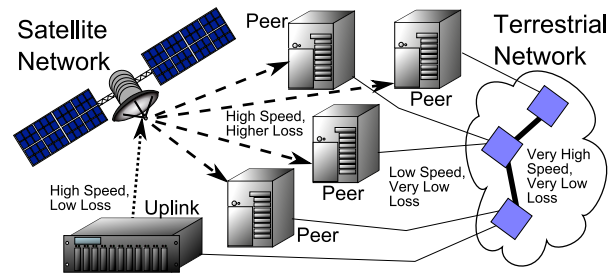


Figure 1. Satellite/Terrestrial Configuration

One motivating application is the distribution of a high-resolution digital television signal to affiliate TV stations. These data files are large (4-8GB), must be played out by the destination nodes with low latency (frames displayed within a few seconds of receipt, inter-frame jitter < 40ms), and uplink loss is smaller than downlink loss (by as much as a factor of 10). In this case, the received uplink power is larger than the received downlink power at an antenna due to higher transmit power and antenna gain on the uplink.

It is worth noting that error recovery from uplink losses and downlink losses can be viewed as distinct problems. For uplink losses, no node receives that block; it is only available from the source. For downlink losses, most peers can provide a missing block. Historically each has been solved using a separate mechanism. We show that one simple, unified solution can solve both with excellent performance.

2.1. Target Environment

Our target environment has high bandwidth in the terrestrial core network (10-40Gbps), moderate satellite bandwidth (20-40Mbps), and low access link bandwidth (128Kbps-1.5Mbps). As we will show, the exact values are less important than the ratios between them, i.e., the ratio of satellite bandwidth to access link bandwidth defines the effectiveness of this approach.

High bandwidth is realistic for the ISP core networks. We use the AT&T core network for our tests. Extensive studies have shown that such networks have minimal loss due to congestion [9], and that locality of hosts is unimportant since terrestrial delay is insignificant compared to satellite delay.

Satellite links are the primary source of loss. We assume burst errors occurring uniformly and independently across links using a standard Gilbert-Elliot model per link.¹ Satellite broadcast packets arrive in order, allowing immediate detection of such loss for most cases.

¹More complex correlated error models do not qualitatively change our results.

One common configuration for affiliates is to have T1 links (1.5 Mbps) to a fast core network and 40 Mbps satellite transponders, of which 17 Mbps is used for forward error correction (FEC) and 23 Mbps is used for user data. Error rates on satellite links will vary between 0.05% to 5% depending on FEC and weather. In such an environment, the 15:1 ratio between satellite and terrestrial link speeds is sufficiently dramatic that one must use the satellite for any large data broadcast.

3. Transmission/Recovery Mechanism

In this section, we describe our solution to the hybrid broadcast problem. First we give a high level design and then discuss the algorithms involved. In all discussion, one source (uplink) transmits data to the satellite, which then broadcasts it to all nodes. Recovery is initiated as soon as a node detects a loss.

3.1. Design

We compare two designs. The first is a simple approach with a single *Scheduler* node that collects and maintains metadata information, and the second is a fully peer-to-peer system. In both designs there is a single source *uplink* node with the original copy of the data.

The Scheduler provides performance enhancements when the number of nodes is on the order of hundreds, but may be a bottleneck for larger networks. Its purpose is (1) to provide a database of received blocks for peers to query and (2) to enforce global load-sharing and fairness. Both of these features are provided in a fully peer-to-peer system at the cost of slightly higher latency and potentially lower fairness.

The uplink broadcasts its data over the satellite and concurrently acts as a peer node in the recovery algorithm. Figure 1 shows this structure. Note that any of the peers or the uplink could act as the Scheduler, or the Scheduler could be a designated node in the core network.

3.2. Error Recovery Algorithm

The error recovery algorithms, presented in Figure 2, address both metadata and data transfer. In both recovery algorithms, data block loss is detected based on a gap in the block sequence numbers in the incoming packet stream. In the scheduler-based algorithm, a peer missing a block can ask the Scheduler for the best peer that can provide the missing block. Note that the best peer may be the original source node. Peers periodically provide information to update the Scheduler’s state that it uses to assign recovery peers.

In the fully distributed peer-to-peer algorithm, peers randomly ask k other peers for block information. With independent and uniform downlink loss, the probability that no peer (out of k peers) receives a block from the satellite is $(1 - \text{loss rate})^k$, which falls rapidly when k increases. If no peer receives the block, it is probably due to an uplink loss and the peers will ask the source.² If the source gets several such requests, it is effectively certain that the block was lost on the uplink and should be rebroadcast over the satellite link or multicast over the terrestrial network. Otherwise the block is sent to nodes individually.

Note that the peer-to-peer system incurs (1) an extra timeout for block recovery if data is lost on the uplink or a poor set of peers was selected, and (2) slightly higher network overhead for state communication. This is the price paid for the extra system scalability; distributed hash tables have similar characteristics but have unnecessarily high complexity and overhead for this type of problem.

More rigorously, let

N	= Number of terrestrial nodes
BS	= Block size
$Pr(L)$	= Probability of loss on a satellite link
BW_s	= Bandwidth of satellite bottleneck
BW_t	= Bandwidth of terrestrial bottleneck
T_{send}	= Time that a block is sent (absolute)
D_{sat}	= Max Delay on satellite (uplink↔nodes)
D_{ter}	= Max Delay between terrestrial nodes
D_{s_xmit}	= Max Delay to send a satellite block (BS/BW_s)
D_{t_xmit}	= Max Delay to send a terrestrial block (BS/BW_t)
D_{tick}	= Delay between clock ticks (timeouts)
T_{peer}	= Timeout for peer/server response

Using this notation we can calculate aspects of the system behavior: maximal block delay, termination time, and so forth.

With uniform independent losses, the chance that a node immediately gets any particular packet is simply $(1 - Pr(L))^2$ since packets may be lost on the uplink or downlink. This is the probability of “ideal” reception at time $T_{ideal} = T_{send} + D_{s_xmit} + D_{sat}$. If only one copy of the packet is lost, it will be detected in at worst one timeout, then metadata requested from the server/peers, then the

²Well-known techniques can be applied to avoid the “ack implosion” problem.

With Scheduler	Without Scheduler
Node detects data block loss	
Send Scheduler NACK	Request block from k peers
Scheduler picks best peer	Peers say if they have block
Request block from peer	Request block from first peer
Peer provides block	
Provide Scheduler info: load, cumulative acks	Request block from source Source re-broadcasts/replies

Figure 2. Error Recovery Algorithms

packet requested from a peer and received at time at most $T_{ideal} + D_{tick} + 4 \cdot D_{ter} + D_{t_xmit}$. When multiple copies of a block are lost, at most $l = \lfloor (D_{tick} * BW_t) / BS \rfloor$ losses can be recovered per clock tick by each node that has received the block. Then, after another (at most) D_{ter} those nodes can provide the block to others.

In the worst case, (1) a block is lost on the uplink so only the source has it and (2) the source is on the terrestrial bottleneck. In this case, after one clock tick (plus network delay) $l + 1$ nodes will have the block, after two ticks approximately l^2 have it, and so on. At worst, the last node will receive the block at time $T_{ideal} + (D_{tick} + D_{ter}) \cdot \lceil \log_l N \rceil + 4 \cdot D_{ter} + D_{t_xmit}$. This is a desirable bound as it grows very slowly, but it is subject to a few constraints.

The first constraint is that D_{tick} is large compared to D_{t_xmit} and D_{ter} . If not, the recovery algorithm breaks down – to make progress, replies to requests must arrive before the next timeout.

The second is that the satellite loss rate must be “streaming-recoverable”, that is, the terrestrial links must be fast enough to fix the losses on the satellite links within a few RTTs of when they occur. This is only true when $(1 - (1 - Pr(L))^2) \cdot BW_s < BW_t$ or equivalently when $Pr(L) < 1 - \sqrt{1 - (BW_t/BW_s)}$. This is why it is the ratio between satellite and terrestrial speeds, rather than their absolute values, that is most important. In practice, the maximum loss rate recoverable at streaming rates is slightly lower due to congestion losses and transient load.

To make this concrete, when losses are streaming-recoverable, the maximum latency to receive a block in our simulations is less than half a second. If the satellite loss is *not* streaming-recoverable, then losses will accrue and the delay to replace a lost packet will grow without bound. Eventually, if the satellite transmission completes and there is no subsequent transmission, the errors can be recovered at the speed of the terrestrial network.

3.3. Peer Selection

The peer selection algorithm is the core of block recovery. With a scheduler, or other metadata, we can globally optimize this step. Without it, we cannot improve upon random selection. The best peer is selected as follows:

Select a random peer from
the minimal cost peers from
all peers which have the block.

The first selection is easy; any reasonable pseudo-random number generator suffices. Similarly, the third selection is easy when information is centralized or other infrastructure exists to maintain it. The second selection depends on “cost.” This can be any formula, but for our tests it is simply load: lower load, lower cost. This selection is a trivial instance of the job scheduling or bin-packing problem where all jobs are the same size (transferring one block). In such a case, minimizing global cost resolves to globally balancing cost, which is exactly what this algorithm attempts to do. Thus, its optimality is limited only by metadata accuracy.

The delay in metadata propagation means the scheduler may err by up to the number of requests that arrive in the time it takes a packet to traverse the network.³ The expected difference is only a few blocks (below D_{tick}/D_{t_xmit}) and will be self-correcting over time: the percent difference from optimal falls as the transmission size increases. Our empirical results show this behavior. Randomization avoids overloading individual hosts during the update period.

4. Evaluation

The system was implemented using the ns-2 network simulator. Due to the expense of satellite transponders, it is quite difficult to perform this type of large scale exploratory work in any other way. This section discusses the design and validation of the simulations, the loss model we use, and the performance metrics with which we test the system.

4.1. Simulation Design and Validation

Our implementation includes the algorithms discussed in Section 3 as well as a fair amount of infrastructure including code for peer connection handling, block tracking, message passing, and so forth.⁴ It comprises about 1600 lines of C++ code adding to the simulator itself, 1000 lines of TCL code to set up and test various configurations, and some shell programming to assist in analyzing results.

The correctness of the implementation was validated in three ways beyond the standard ns validation suite; these measures make us confident that the simulations accurately capture the salient features of this problem. The first is by comparing results to the theoretical predictions from the

³Up to $D_{t_xmit} + 4 * D_{ter}$: D_{ter} to return metadata, D_{ter} to request the block, D_{t_xmit} to send it, D_{ter} to traverse the network, and D_{ter} to return an ACK to the Scheduler.

⁴Ironically, the lack of some desired features in the simulator means that much of the work to create a “real-world” implementation is complete.

earlier analysis. The second is by running against a set of small contrived configurations and manually reproducing all output. Finally, the third is by visualizing results in the network animation tool nam and looking for unexpected behavior. Over the course of development, we also repeated prior tests to verify consistency in results.

We use ns to perform tests varying different aspects of the algorithm, error rates, error distribution (correlation across hosts), link speeds, link costs, network topology, block size, and transfer sizes. Some of these parameters turn out to be insignificant, and thus we present only the interesting data. We tested both the scheduler-based algorithm and the fully distributed P2P algorithm (with k set to 3). Our results validate the six claims made in Section 1.

Initial tests are performed on a relatively simple topology to provide the equivalent of micro-benchmarks. This topology connects access links to a single backbone router, creating a terrestrial star topology. The satellite links are 23Mbps (effective data bandwidth) and terrestrial links are DSL or T1 speeds (768Kbps down/128Kbps up, 1.5Mbps down/768Kbps up, 1.5Mbps down/128Kbps up, or 1.5Mbps both ways).

Other tests are on a larger, realistic topology using the AT&T core US network [3] (covering a significant portion of the core Internet in the United States), and the information from the PBS/PlanetLab Grand Challenge [5] correlated with station/location information from the PBS web site [13]. This is a concrete, realistic topology and shows that our approach provides an alternative solution to many issues in the grand challenge problem.

4.2. Performance Metrics

Our evaluation primarily concerns our six claims in the Introduction. Metrics include:

- global transfer termination time— simulated seconds; when the last peer receives the last block required
- fairness— how evenly distributed cost is among peers
- streaming latency— percent completion (in aggregate, total blocks) as a function of time; a measure of how good the system is at streaming
- raw bandwidth— megabits/second; a simple measure of system speed
- link utilization— percent of aggregate or individual links; a simple measure of efficiency.

Fairness is evaluated using Jain’s measure from [11]. In this case, x_i is the cost incurred on the uplink by node i ; the number of blocks it has provided to others. Put another way, this is the amount of work done that is of no direct benefit to a given node:

$$\text{fairness} = \frac{(\sum x_i)^2}{(n \cdot \sum x_i^2)}$$

We want this value to be close to 1 (completely fair) both to provide incentive for users to participate and because fairness is a good measure of global performance: spreading load evenly leads to the fastest termination. We exclude the initial source node from this calculation, as it will by definition be transmitting an order of magnitude more data.

Together, our system attempts to maximize broadcast link utilization, minimize access link utilization, minimize latency to receive each block, and maximize global aggregate goodput in the system.

5. Results

We present results for both the micro-benchmark topology and the more realistic telecommunication company core network topology. First, we demonstrate the system performance as a function of satellite link loss. Second, we demonstrate scalability in terms of file size and number of receiver nodes. Third, we show block latency is low. Fourth, we demonstrate the access links' utilization efficiency. Finally, we show that the protocol as a whole is very fair. Together these justify our original claims.

5.1. Loss Recovery

The first question concerns how much loss we can recover using this system. Figure 3 shows a graph of loss rate over the satellite links and completion time for a 100MB broadcast to 10 nodes.

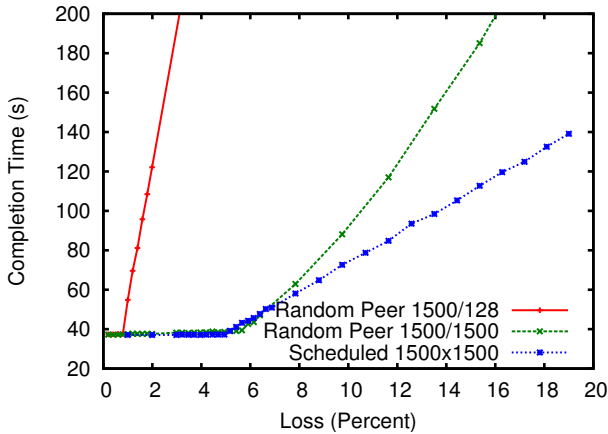


Figure 3. Loss Rate vs. Completion Time

As we expected, there is a sharp knee showing the point at which streaming recovery is no longer possible. This knee is at the same point for both schemes given the same access link speeds. With slower peer links, less loss can be recovered and knee moves lower.

According to our calculations in Section 3.2, the maximal theoretical streaming-recoverable loss rate is 6.5%, but we see the actual maximum at 5%. This is about 75% of ideal recovery efficiency. The difference is primarily due to the application's in-order block semantics; if some blocks in the recovery window cannot be retrieved, recovery may temporarily stall waiting for them. This may occur when multiple blocks are lost on the uplink and exist only on the source; it requires multiple iterations to propagate them throughout the network.

The above graph shows the aggregate loss when loss rates are equal on uplink and downlink. That is, a unidirectional loss probability of $p\%$ leads to a loss rate above of $1 - (1 - p/100)^2$. There is no reason to expect the loss rates to be equal; in fact, uplink losses will tend to be lower as described above. It can be shown that system performance is only weakly tied to loss correlation; in the worst case, all nodes can recover from the source node in $\log(N)$ iterations of the algorithm. Further discussion has been removed due to length limitations.

5.2. Scalability

The second question concerns scalability—both in terms of file sizes and in terms of system size (number of nodes). Ideally, completion time should be linear in the file size. Similarly, as the number of nodes grows, the completion time should stay constant (below the streaming point) and grow slowly above it (as nodes become overloaded).

To test the former, we broadcast files of sizes ranging from 1MB to 1GB under various amounts of loss (above and below the knee in the prior graph) using T1-speed peers, and present the results normalized to the percent of ideal time (zero loss on the satellite link).

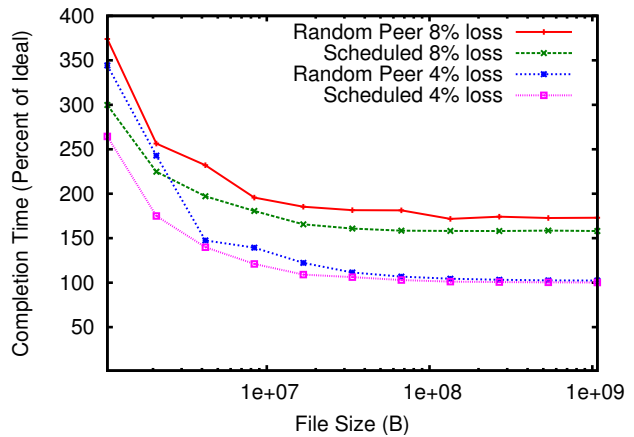


Figure 4. Completion Time vs. File Size

For 1-8MB transfers, network latency and other overheads dominate. Above that size, we can effectively meet the ideal transfer time for moderately high loss rates. The peer-to-peer system matches the scheduled scheme well but is slightly less efficient for high loss rates—the randomized block location scheme has higher overhead.

To test scalability in terms of nodes, we test a 100MB broadcast to varied number of nodes. Figure 5 shows completion time for increasing numbers of nodes under 2% loss.

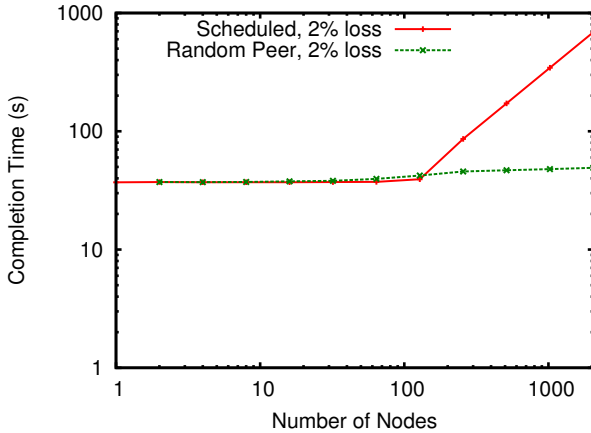


Figure 5. Completion Time vs Node Count

A 2% loss is streaming recoverable with 1.5Mbps peers, but at around 128 nodes, the capacity of the centralized Scheduler to track losses and reply with metadata is exhausted. Moving the Scheduler to a 100Mbps core link shifts that knee to around 1,800 nodes. The peer-to-peer system is slightly less efficient, but scales better.

The peer-to-peer curve grows approximately with the log of the number of nodes, as expected—this is due to block propagation requiring an additional iteration with each doubling of the number of nodes. Larger numbers of nodes were not simulated due to time constraints and other limitations with the simulations/simulator; however, we reiterate that these results satisfactorily meet the goals of our application.

5.3. Intra-Transfer Performance

The next logical question is how the system is performing within a transfer, i.e., how the different parts of the transfer progress and whether we meet our latency goals. Figure 6 shows the aggregate blocks received via the satellite and terrestrial networks over time for all 10 nodes under high-loss (10%) conditions.

The ideal curve transfers all data over the satellite link with no loss. Our actual performance is effectively a linear combination of the degraded satellite signal and terres-

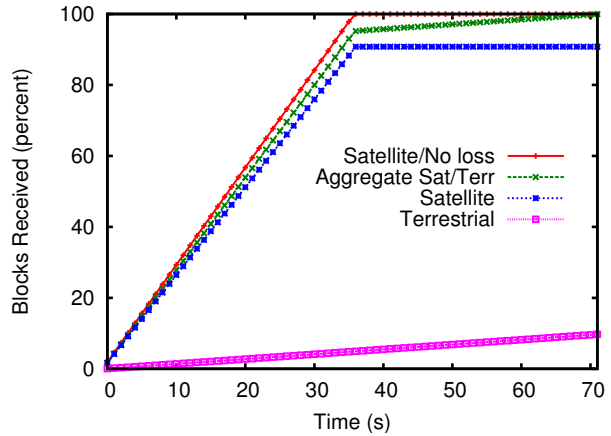


Figure 6. Initial Transfer and Recovery Phases

trial recovery transmissions. For low-loss cases, both terminate at same time. For higher loss cases such as this, we spend time after the satellite transmission has completed to recover the errors.

This graph shows that while the system is able to recover higher losses at low cost, the satellite link may be underutilized. In such a case, we need to increase the forward error correction slightly to bring the net satellite errors down; otherwise the per-block latency grows unacceptably (up to 32s for the last block lost in this test).

The normal latency for blocks is captured in Figure 7, which shows the difference from the expected time of arrival for the blocks in a 100MB broadcast to 10 hosts with 2% satellite loss.

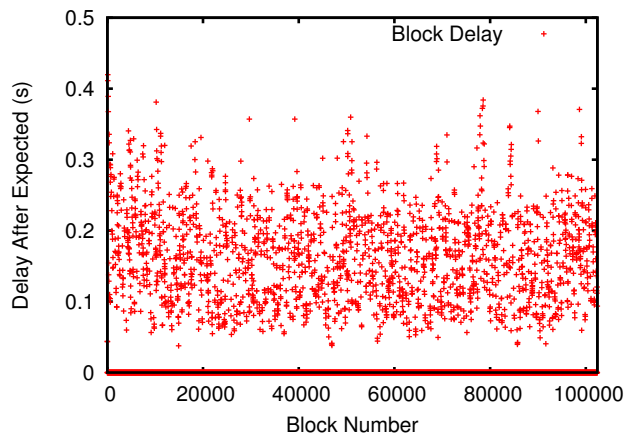


Figure 7. Block Delay Due to Loss

The vast majority of the blocks (98%) are received when expected from the satellite. The 2% lost are recovered via the peer-to-peer mechanisms, with latency up to 450ms but on average about 175ms. Each doubling of the number of hosts increases the worst-case latency we observe by ap-

proximately 50ms. The total latency incurred is of the same order as the baseline latency to reach and return to a geosynchronous satellite.

5.4. Network Efficiency

To show system efficiency we must have high access link utilization over the duration of the transmission (the core network is lightly loaded relative to its total capacity). Figure 8 shows the percent of access link capacity utilized by data; that is, not including packets corrupted, metadata, or packet header overhead using the the large core (Internet-like) network configuration. We perform a 1GB transfer under 2% loss rate on this network, parameters representing a realistic streaming-recoverable scenario. This test uses the centralized Scheduler. Full P2P results are qualitatively the same but have higher variability, making for messier illustrations.

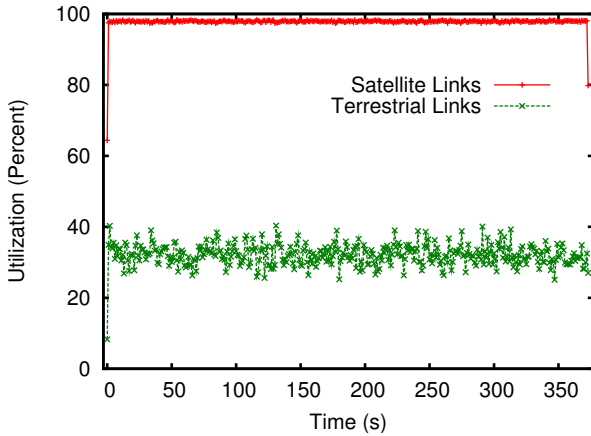


Figure 8. Percent Utilization of Core Access Links with 2% Satellite Loss

The first thing to notice is that with streaming-recoverable losses, there is no phase where the satellite is idle. Both networks are utilized for the whole transfer. Second, the losses only require about 30% of the data capacity of the terrestrial network. Third, the results are not qualitatively different than those on simpler configurations (not shown due to space limitations). This has held true for all our tests. The primary difference is that the backbone structure creates a higher variation in the data than a simple benchmark topology.

In general, demand for access link bandwidth is directly proportional to satellite loss, and satellite utilization (goodput) is inversely proportional to loss. The 2% loss rate in Figure 8 shows high satellite utilization and low access link utilization. The point at which transient access link utilization reaches 100% is the point at which streaming recoverability becomes impossible.

Furthermore, the data to be sent by each node, and hence the link utilization, is controlled internally by a token-bucket mechanism. Currently, it allows 100% of the link to be used for recovery, but the protocol is user-tunable. Users can limit this to any desired proportion to avoid competing with other traffic.

5.5. Fairness

Finally, fairness under Jain’s measure has been very high in all cases. With the Scheduler, for the simple topologies and blocks sent it was uniformly over 0.999, meaning all nodes sent out almost the same number of blocks. Similarly, for recovery blocks received it was over 0.999, in this case due to the global termination constraints and uniform loss behavior. Under nonuniform loss, by definition, some nodes will unfairly load the system to recover their data, but the blocks sent will still be evenly allocated (i.e., no tit-for-tat behavior).

The more realistic core topology had fairness values consistently around 0.987, also very high but somewhat lower than the simpler topology. The difference was again due to the structure in the core network; some nodes were closer to the Scheduler. This implies the Scheduler data was consistently more fresh for those nodes, and they would tend to have slightly higher load.

Without a centralized scheduler, 65% of our tests showed fairness over .99, 94% over .95, and 100% were over .88 fair. In general, the smaller the transfer the greater the chance that the random peer query will create an unfair work allocation; the last 6% above were transfers where peers uploaded less than 20 blocks/node.

In sum, these results coincide with our prior analysis and support our claims as to the system’s performance under a variety of conditions.

6. Discussion and Related Work

There is a large body of work on content distribution, multicast tree construction, data distribution over unreliable transports, and P2P file transfer. It falls into three categories: terrestrial only, based on a set of point-to-point transfers; satellite only, based on a global broadcast mechanism; and other hybrid solutions based on the use of both satellite and terrestrial systems.

Content distribution networks such as Akamai [2] fall into the first category. They have many of the same goals as this work, but focus on putting smarts “in the network” and the use of careful engineering to get caches close to the expected user base. For example, if one places a very powerful source node on a very fast link, there is no need for P2P transfers: the source can provide all error-correction information.

Multicast trees or meshes (such as Bullet [12] or SRM [10]) and current P2P solutions (such as BitTorrent [8]) also fit in the first category. In the prior case, they solve the problem of scalable one-to-many transfer, but induce problems with ACK implosion, node churn, or group membership/agreement. The latter are pull-based mechanisms for relatively small files (<4GB) with high delay tolerance, due to the use of randomization, rarest-first, erasure coding, and other mechanisms. Neither can exploit hybrid networks. We make weaker assumptions about internal network structure, use different internal mechanisms, provide better scalability and lower latency, and avoid most of these issues entirely.

Commercial media distribution networks (ClearChannel, CNN, etc) fall into the second category. They are proprietary systems which involve custom hardware and software at all levels, specialized and dedicated to whatever they transmit (e.g. AM or FM radio signals, PAL or NTSC TV signals, HDTV, etc).

Much work in this category has centered around mechanisms for error detection and correction, focusing on FEC and TCP's well-known performance problems with loss being treated as congestion. Similarly, error detection after-the-fact is trivial via checksums, CRCs, or stronger cryptographic hashes. Again, these approaches can not take advantage of a side channel (the terrestrial network) for error recovery information. Block level Forward Error Correction (FEC) such as Tornado or Luby erasure codes [6, 7] or other network coding approaches have problems with high-latency encoding or decoding. Another drawback is that fixed FEC consumes satellite bandwidth—reducing user-available bandwidth—even if error rates are low.

The third category has one closely related project, a peer-based recovery mechanism for satellite transmission by Awal et al. [4]. Differences from our work include an assumption of no terrestrial link to the source, meaning there is no way to communicate with the original source and hence no guarantee that nodes will receive the complete file. Also, in this work, error recovery is only done when the satellite broadcast is completed; thus, the concepts of block latency or streaming-recoverability do not apply. Finally, they do not address the optimizations of link utilization or system cost as we have.

7. Summary and Future Work

We have presented a simple, flexible, robust, fair, and minimal cost mechanism for low-latency data transmission over hybrid satellite/terrestrial networks. Analysis and simulations have shown excellent performance in both specially contrived and realistic environments under a variety of conditions. One can easily include arbitrary link cost models and still achieve globally maximal performance.

The performance of this approach is sensitive to the ratio between satellite and terrestrial link speeds (fewer losses can be recovered at speed when the ratio is high) and requires good connectivity to the scheduler node. On the other hand, performance is insensitive to (is robust under changes in) network topology, absolute link speeds, link latency, and error correlation. It is particularly useful when satellite errors are low; it requires no complicated infrastructure or large memory footprint (e.g., multicast trees or erasure coding), and recovers the original data stream in almost real time.

Future directions for this work include integration of an automatic congestion control mechanism based on feedback, and, if possible, experiments on real-world hybrid networks.

References

- [1] PlanetLab Consortium. <http://www.planet-lab.org/>.
- [2] Akamai Technologies Inc. Akamai content distribution system. <http://www.akamai.com/>.
- [3] AT&T. Global IP network. <http://ipnetwork.bgtmo.ip.att-net/pws/index.html>.
- [4] M. Awal, Y. Tsuchimoto, and K. Kanchanasut. An approach of peer-based packet recovery using edbit for unidirectional satellite environment. In *Proceedings of the Workshop on Asia-Pacific Networking Technology*, Jan 2005.
- [5] M. Bowman, J. Sedayao, and R. McGeer. Bakeoffs. <http://www.planet-lab.org/Talks/2005-05-01/Bakeoffs-final.ppt>, Sep 2005.
- [6] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege. A digital fountain approach to reliable distribution of bulk data. In *Proceedings of ACM SIGCOMM*, pages 56–67, 1998.
- [7] J. W. Byers, M. Luby, and M. Mitzenmacher. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In *Proceedings of INFOCOM*, pages 275–283. IEEE, March 1999.
- [8] B. Cohen. The BitTorrent file sharing protocol. <http://bittorrent.com/>.
- [9] S. Floyd. Measurement studies of end-to-end congestion control in the internet, 2002. <http://www.icir.org/floyd/ccmeasure.html>.
- [10] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Transactions on Networking*, 5(6):784–803, Dec 1997.
- [11] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared systems. Technical Report TR-301, Digital Equipment Corporation, Littleton, MA, 1984.
- [12] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat. Bullet: High bandwidth data dissemination using an overlay mesh. In *Proceedings of ACM SOSIP*, volume 37, pages 282–297, October 2003.
- [13] Public Broadcasting Service (PBS). PBS — Station Finder. <http://www.pbs.org/stationfinder/index.html>.