TECHNICAL RESEARCH REPORT

A Reliable Multicast Transport Protocol for Satellite Communication Systems

by Gun Akkor, John S. Baras, and Michael Hadjithedosiou

CSHCN TR 2003-17
(ISR TR 2003-32)

A Reliable Multicast Transport Protocol for Satellite Communication Systems

Gün Akkor, John S. Baras, Michael Hadjitheodosiou

In this paper, we propose a reliable multicast transport protocol for satellite communication systems. Many of the emerging applications in the Internet would benefit from reliable multicast services, and broadband satellite communication systems have attractive characteristics for supporting such services. However, many of the protocols designed primarily for terrestrial networks do not perform well over satellite networks. Therefore, it is necessary to look at the problem of reliable multicast in the solution space of satellite communications. Our protocol makes use of a special form of forward error correcting codes and couples it with an adaptive window based control mechanism to dynamically adjust the number of encoding packets forwarded to the users. Protocol makes very good use of the broadcast nature of the satellite channel and attempts to minimize the feedback from users. We evaluate the protocol performance by extensive computer simulations.

I. INTRODUCTION

Many of the emerging applications in the Internet, such as distributed computing, software updates, distance-learning, and Internet gaming would benefit from reliable multicast services. These applications are distributed in nature and require concurrent transmission of the same content to multiple users. Broadband satellite communication systems offer wide-area coverage and ubiquitous access to a potentially large number of users. Therefore, they are a natural technology option for carrying multicast services.

However, many of the protocols designed primarily for terrestrial networks do not perform well over satellite networks. TCP, which is the dominant protocol in the Internet for reliable delivery of data, suffers from performance degradation over satellite links due to long propagation delays and high loss rates [1]. It also does not scale well to concurrent transmission of the same content to multiple users because of *feedback implosion* [2], and *loss path multiplicity* [3] problems. Therefore, it is necessary to look at the problem of reliable multicast in the solution space of satellite communications.

In this paper, we propose a reliable multicast transport protocol for a satellite-based communication system. Our protocol makes use of a special form of forward error correcting codes and couples it with an adaptive window based control mechanism to dynamically adjust the number of encoding packets forwarded to the users. Protocol makes very good use of the broadcast nature of the satellite channel and attempts to minimize the feedback from users.

The rest of the paper is organized as follows. In the next section, we discuss related work on this topic. In Section III, we describe our network scenario and our assumptions on the operation of the communication system. Section IV outlines the goals of the protocol and introduces the use of LT codes. In Section V, we present the details of our protocol. Sections VI and VII discuss the simulation environment and present performance results. Section VIII concludes the paper.

II. RELATED WORK

Primary focus on this topic has been on providing some form of forward error protection by transmitting redundant information along with the forwarded data. Forward error protection helps recover corrupted data and thus minimizes the need for retransmissions over the link. For multicast services, it also helps solve the feedback implosion problem and improves the scalability. Forward error correction (FEC) coding can be applied at the network layer or above, to generate (n-k) parity packets for a group of k data packets [2][4]. In this case, a single parity packet can be used to repair the loss of any one of the n transmitted packets. This means that a single parity packet can repair the loss of different data packets at different receivers, improving transmission efficiency and scalability. In [5], author shows the potential benefit of packet level FEC coding on a generic reliable multicast protocol. In [6], author discusses the possibility of transmitting redundant packets using a separate channel along side the original data packets. In [7] and [8], authors propose schemes for adaptively adjusting the number of parity packets transmitted to the receivers. Implementation of packet level FEC coding is difficult for large packet sizes [9]. Also, number of parity packets that can be transmitted against packet losses is limited to the block size of the code. Therefore, authors in [10] propose a scheme using Tornado codes, which are capable of working over large block sizes.

Our protocol favors integration of packet level FEC coding at the transport layer. We use a special form of packet level FEC codes, namely the LT codes, which are capable of generating a large number of encoding packets for the same group of input packets. This flexibility in the number of encoding packets improves the efficiency of the protocol considerably. We improve the ideas in [5] and [6], and build a novel control mechanism that dynamically adjusts the number of encoding packets transmitted for each input packet group, as well as which packets are to be transmitted in response to receiver requests.

III. NETWORK DESCRIPTION

In this paper, we consider a satellite communication system, where a Ka-band satellite provides broadband services to a large number of users located inside its footprint. In this scenario, users that are equipped with two-way direct communication terminals, access the terrestrial network through a *gateway* node referred to as the network operations center (NOC). A user terminal may serve a small home network with only a few user machines or may act as a gateway for a local area network. In general, we assume that there are as many terminals as there are users, and from our point of view, user terminals are the end-points. The structure of the network is shown in Figure 1.

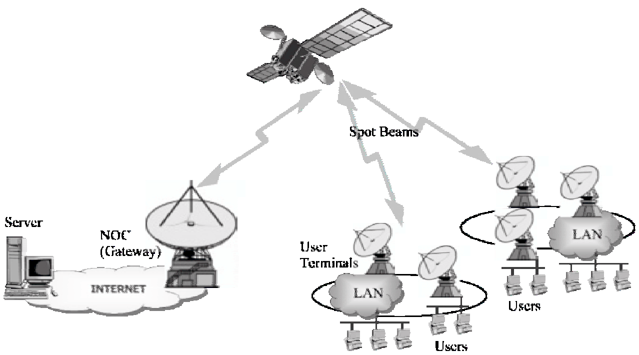

Figure 1 An overview of the network architecture

We assume that the data (software download, web content, file transfer) is to be distributed reliably to many end-users (one-to-many communication) and it is located at a server (sender) on an IP based terrestrial network. We assume that the connection is established using the idea of *TCP splitting* [11] that has recently been of much interest to the research community. In this scenario, the connection is split into three components: (a) a TCP/IP connection is established between the server and the satellite gateway, (b) over the satellite portion of the network, the DVB data broadcasting standard [12] is used for transporting packets, and (c) between the user terminal and the user machines another TCP/IP connection is established. We assume that data reliability is ensured natively by the TCP protocol in sub-connections (a) and (c). Therefore, we are primarily interested in providing reliability over the satellite portion (Figure 2) of the network.

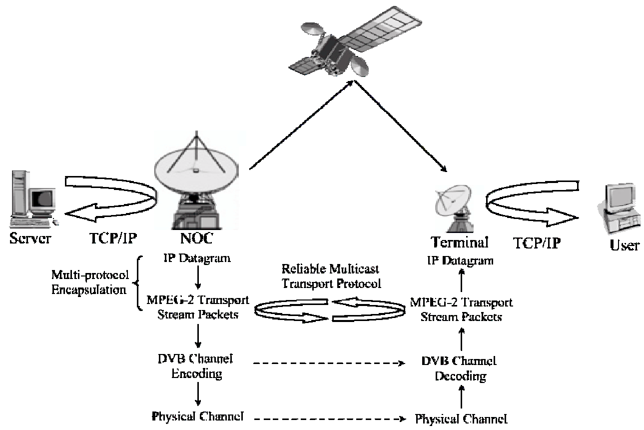

Figure 2 Proposed breakup of the communication session

DVB standard for data broadcasting allows encapsulation of IP datagrams into MPEG-2 Transport Stream Packets using the Multiple Protocol Encapsulation (MPE) standard [12]. The incoming stream of IP datagrams are segmented into 188 Bytes transport packets at the satellite gateway. The DVB channel encoder encodes each packet first using a shortened (204,188) RS code and then passing the packet bit stream through a rate punctured convolutional encoder. This concatenated channel encoding is effective against detecting and correcting random errors and recovering from packet erasures up to a certain level. However, it is not sufficient against communication outages that may occur due to severe rain and atmospheric attenuation. For reliability, in this case, packets have to be retransmitted from the satellite gateway.

Retransmission of individual packets over a high latency and error prone satellite connection is not desirable. Moreover, in one-to-many communication, individual end-users will likely have different packet requirements and therefore accommodating all the requests would cause waste of network resources. We present an efficient reliable transport mechanism that operates on top of the DVB coding, at the MPEG-2 layer between the satellite gateway and each of the end-user terminals.

IV. PROTOCOL OUTLINE

Some of the properties that the transport protocol should satisfy are:

i. Minimize the number of user requests per packet loss.
ii. Make good use of the broadcast capability of the satellite connection.
iii. Adapt to the changing channel conditions and to the perceived channel quality of the users.
iv. Be robust against different application requirements.

In order to achieve some of these goals, we make use of a special class of FEC codes, namely the *LT codes* [13]. This coding is applied at the packet level and it assists in packet retransmissions rather than guarding against channel errors. The coding is performed at the MPEG-2 level, and is applied to the MPEG-2 transport stream packets. Let k denote the number of MPEG-2 transport packets currently buffered at the satellite gateway. The LT encoder outputs individual encoding packets using this batch of k MPEG-2 packets by: (i) randomly choosing the degree d of the encoding packet from a degree distribution, (ii) choosing, uniformly at random d distinct input packets as neighbors of the encoding packet, and (iii) calculating the value of the encoding packet as the exclusive-or (XOR) of the d neighbors. Each encoding packet is uniquely identified by its header and has the same packet size. The encoding packets generated in this manner are further encoded by the DVB channel encoder as described in the previous section and are transmitted over the channel.

The clever construction of the LT codes allows end-users to recover the original batch of k transport packets, if they manage to accumulate a slightly larger set of $K = (1+\varepsilon)\cdot k$ of the encoding packets that are generated from the same input packet batch. Additionally, the number of encoding packets that can be generated from the same input packet batch is typically much larger than the size of the batch. The benefit of this construction is two fold. First, it allows the satellite gateway to transmit additional encoding packets as long as there are users that have not yet accumulated enough packets for the batch. Secondly, it simplifies the user request for additional packets, since now only information that has to be forwarded to the satellite gateway is comprised of the number of additional encoding packets (from the same batch) that is required by each receiver until all receivers complete the reception. This construction also makes very good use of the broadcast nature of the satellite channel since every transmitted encoding packet benefits all the users equally.

The transport protocol, in the first round, generates a set of $K = (1+\varepsilon)\cdot k$ encoding packets for an input batch of k MPEG-2 transport packets and transports them over the satellite channel. Each user, at the end of this initial round, evaluates its success and reports the number of additional encoding packets required to complete its reception. The satellite gateway collects these reports and in the second round, transmits enough encoding packets to accommodate the need of the worst-case user (the one that requested the most packets in the first round). In the subsequent rounds, the process continues in a similar manner until all receivers successfully complete their reception of the batch. In the next section, we will describe the behavior of our reliable transport protocol.

V. PROTOCOL DETAILS

We assume that data can be organized into B batches of k input packets. Set K as $K = (1+\varepsilon)\cdot k$.

A. Protocol Behavior at the Satellite Gateway

End-users have to accumulate at least K encoding packets in order to successfully recover any input batch. Therefore, for each batch B_i, the protocol has to transmit *at least* K encoding packets in the first round. The number of encoding packets transmitted in the first round of each batch is controlled by the *transmission window* (W_1). W_1 encoding packets from the batch is stored in an outgoing queue Q1. Only after the transmission of W_1 encoding packets for an input batch, the end-users can start the recovery process. Therefore, the minimum value of the W_1 is always equals to $W_1^{min} = K$.

A user may fail to complete the recovery process of a batch after the first round, if the packet loss over the channel is too severe, and not enough packets have been accumulated. In this case, additional encoding packets are transmitted in the subsequent rounds. A second window, W_2, controls the transmission of additional packets for failed batches. Additional encoding packets for failed batches are stored in an outgoing queue Q2. In the best case, all batches are recovered at the end of the first round. Therefore, W_2 has a minimum value $W_2^{min} = 0$.

The transmission of the packets of windows W_1 and W_2 follow a time division, where transmission of packets of window W_1 is followed by the transmission of the packets of window W_2. Therefore, two window sizes may be viewed as the time-share of the two queues (Q1 and Q2) that are served by a processor with rate $W = W_1 + W_2$ packets per unit time. However, if one of the queues is empty, then the processor does not remain idle and immediately moves to serve the next queue in a round robin fashion. One of the goals of the protocol is to minimize the value of W per transmitted input batch over the lifetime of the session.

At the start of the transmission, the protocol has no information on the channel conditions of end-users. Therefore, the transmission window (W_1) is initialized to its minimum value $W_1(0) = W_1^{min} = K$. Similarly, W_2 is initialized to its minimum value $W_2(0) = W_2^{min} = 0$. The protocol starts transmitting $W_1(0)$ encoding packets for the first input batch and immediately moves to the transmission of $W_1(0)$ encoding packets for the subsequent batches until the first feedback reports from the users arrive (since there are no packets to transmit in Q2).

When the protocol receives feedback reports from the users, it gets two pieces of information from every report: (i) a request for additional encoding packets for a batch that previously failed in the recovery process, and (ii) a weight value ω, which is the average of additional encoding packet requirements for all failed batches at the user, at the time of the transmission of the report. The weight value of an end-user is an indicator of how well the receiver succeeds in the recovery process and is used at the gateway side of the protocol to adjust the size of the transmission windows (W_1 and W_2).

Every request is queued in a request queue (RQ). If a request for that particular batch has already been queued by another end-user, then the additional encoding packet requirement for the queued request is updated as the maximum of the current value and the newly requested value. The protocol generates additional encoding packets for the requested batches and stores them in queue Q2. The weight value of the user (as stored at the gateway) is updated to the value last reported.

From user perspective, the protocol must make sure that every batch can be recovered at the end of the first round of transmission, because additional rounds add to the overall delay. At the same time, protocol should transmit as few encoding packets as possible (ideally completing the recovery after transmitting K encoding packets). Therefore, the protocol has to adjust its transmission windows such that it transmits enough encoding packets for every batch in the first round for high recovery probability while avoiding transmitting more than necessary by the user group.

The windows are adjusted every time feedback is collected at the satellite gateway, based on the following rules:

i. At instance n, $W_2(n)$ is set to the sum of the additional packet requirements of all requests that are in the request queue (RQ), i.e. all requests that are in the queue are served at the next transmission instance.

ii. If W_2 is increasing, i.e. $W_2(n) > W_2(n\text{-}1)$, it indicates that the current value of W_1 is not large enough, i.e. not enough encoding packets are accumulated after the first round under current channel conditions.

iii. The weight of an end-user is the average number of packets it is going to request, in the future, for a typical batch. Therefore, the maximum of the weights of the end-users can be used as an estimate of the value of a future request entry for a typical batch. Define $S_\omega(n)$ as:

$$S_\omega(n) = \max_{r \in R}\{\omega_r\} \cdot \text{size(RQ)}.$$

$S_\omega(n)$ is an indicator of the trend in the size of W_2 for a queue size comparable to current request queue.

iv. If $S_\omega(n) > W_2(n)$, then W_1 is incremented by one, $W_1(n) = W_1(n\text{-}1)+1$. If $S_\omega(n) < W_2(n)$, W_1 is decreased by one, $W_1(n) = W_1(n\text{-}1)\text{-}1$. Otherwise, it remains unchanged.

The new window sizes are used when transmitting the encoding packets for the next available batch.

B. *Protocol Behavior at the End-Users*

At the beginning of the transmission, user protocol initializes the weight value of the user to 0. As encoding packets for the input batches are received, they are stored in temporary buffers awaiting the start of the recovery process. Whenever the user protocol detects the start of encoding packets for a new batch, it starts the recovery process for the last transmitted batch. The user protocol polls the temporary buffers for all batches that are now in the recovery phase. If the recovery process of any batch is successful, the input packets are forwarded to upper layer for reassembly. For all batches for which the recovery process has failed, the protocol files requests in the users' request buffer (RQ_U) for additional encoding packets or updates a previously filed requests to reflect the new packet requirements.

After temporary buffers are polled and completed batches are removed from them, the user protocol calculates the new weight value of the user by calculating the average of the packet requirements of all batches that have failed the recovery process. The user protocol then transmits the first request in the request buffer together with the new weight value of the user. The session continues as described until all batches complete the recovery process. At this time, the user protocol transmits a *terminate message* to the gateway protocol and terminates the connection. In the next section, we describe our simulation environment and present simulation results on the performance of the algorithm.

VI. SIMULATION ENVIRONMENT

A. *Channel Model*

We model the satellite channel by a threshold-based 2-state Markov Chain. In this model the channel is either in *GOOD* state, if the transmitted signal experiences less than Γ dB. attenuation, or it is in *BAD* state, if the signal fade is more than Γ dB., where Γ is the fade attenuation threshold [14]. We assume that if the channel is in *GOOD* state, channel coding is capable of correcting all bit errors and for simulation purposes the probability of bit error at the output of the channel decoder is zero. In the *BAD* state, the channel behaves as a Binary Symmetric Channel (BSC) with bit error probability equal to p_b at the output of the channel decoder. The transition probability matrix M characterizes the channel state:

$$M = \begin{bmatrix} r & 1-r \\ 1-s & s \end{bmatrix}, \tag{1}$$

where r and $(1\text{-}s)$ are the probabilities that the channel will be in the *GOOD* state at the k[th] symbol duration, given that the channel was in *GOOD* or *BAD* states, respectively at the (k-1)[th] symbol duration. Using the results of the ACTS Propagation Experiments [15], for a fade attenuation threshold of Γ = 10 dB., we have r = 0.9999813, and (1-s) = 0.00172.

In order to evaluate the performance of the transport protocol, we need to calculate the bit error probability at the output of the channel decoder. In order to estimate the value of p_b, we use the link budget calculations of a commercial satellite system proposed in [16]. The calculations confirm that for signal attenuation of less than 10 dB., link budget margins and channel coding are capable of keeping the bit error probability around 10^{-9} at the output of the channel decoder. For signal attenuation of more than 10 dB., on the other hand, bit errors occur with probability $p_b \geq 0.1$ at the decoder output [17]. Using this result, in our simulations, we evaluate the time progress of the channel state for every bit transmission and assume that a bit is in error with probability $p_b = 0.1$ when the channel is in *BAD* state and no errors occur when the channel is in *GOOD* state. Since these errors are at the output of the channel decoder, we further assume that a packet is corrupted if at least one bit is in error. We use this packet loss pattern in our numerical results.

B. Simulation Environment

In our simulations, we assume that the satellite gateway has a fixed number, B = 100 input batches, each with k = 220 MPEG-2 transport stream packets. The LT coding instance requires K = 235 encoding packets to be transmitted for a successful recovery of an input batch. Therefore, W_1^{min} is set to 235. The protocol performs between the satellite gateway (NOC) and the user terminals. We assume that the one-way propagation delay between the satellite and the user terminals is 112 milliseconds. The effective link rate at the MPEG-2 transport layer is 4 Mbps.

VII. RESULTS

In this section, we provide simulation results on the performance of our protocol under different scenarios. In Table 1, we present the numerical figures on the per receiver performance of the protocol as a function of the receiver group size.

RECEIVER PERFORMANCE PER BATCH

# RECV	# RNDS	RECP. EFF. (%)	DELAY (MSEC)
10	1.2660	0.9457	247.5460
100	1.2153	0.9343	225.5252
1000	1.2026	0.9193	220.8909
10000	1.2131	0.9098	230.7485

Table 1 Receiver performance per transmission of a batch.

Table 1 shows that a receiver successfully completes the transmission of a batch, on the average, in 1.2 rounds with a delay of approximately 220-250 milliseconds. These figures remain almost constant over the changing group size. The reception efficiency, which is the ratio of the batch size to the total number of packets received for the batch over the lifetime of the session, remains over %90 for the group size range. The reception efficiency decreases with increasing group size, because the variation in the reception quality of the users forces the satellite gateway to transmit more additional encoding packets.

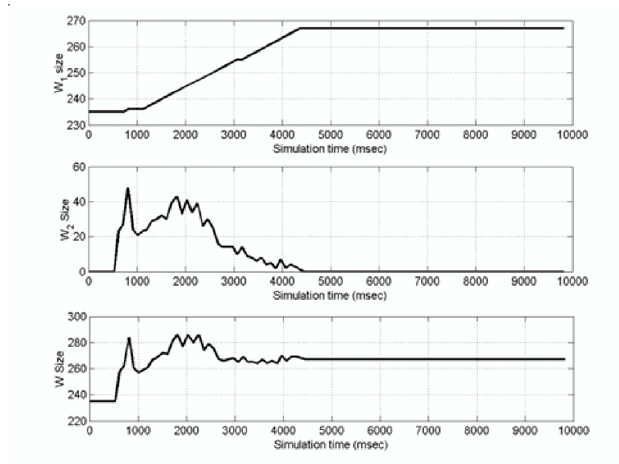

Figure 3 Progress of the session windows

In Figure 3, we plot the size of the session windows for a typical case of 1000 users. We observe that at the beginning of the session, W_2 increases, which indicates that users are not accumulating enough encoding packets at the end of the first round of transmission and asking for more packets. In response to this, W_1 size is dynamically adjusted and at time $t = 4500$ milliseconds, W_2 decays down to zero, i.e. users are accumulating enough packets at the end of the first round of transmission and are not requesting additional packets. After this point, all batch transmissions are completed in one round. The session lifetime of this particular example is t = 10000 milliseconds. This is an important property because it minimizes the number of feedback reports transmitted by the users in response to losses.

Table 2 gives the numerical figures on the maximum value that the W_1 settles at, and the variation in its size from the start of the session. Similarly, Table 3 gives numerical figures on the maximum value reached by W_2 before decaying back to zero and the variation in its size from the start of the session.

SIZE OF PROTOCOL WINDOW W_1

# RECV	MAX	MEAN	STD
10	259.0	252.3	9.4146
100	262.0	255.0	10.3445
1000	267.0	258.1	12.2085
10000	269.0	258.5	12.9432

Table 2 Size of transmission window W_1 over the lifetime of the session.

SIZE OF PROTOCOL WINDOW W_2

# RECV	MAX	MEAN	STD
10	38.0	6.25	10.5499
100	43.0	6.47	11.4622
1000	48.0	7.62	12.7404
10000	54.0	10.01	15.6660

Table 3 Size of transmission window W_2 over the lifetime of the session.

OVERALL PROTOCOL BANDWIDTH

# RECV	MAX	MEAN	STD
10	278.0	258.5	8.4453
100	288.0	261.5	8.9120
1000	286.0	265.7	9.8297
10000	295.0	268.5	11.2712

Table 4 Overall transmission bandwidth over the lifetime of the session.

Table 4 shows the maximum value that the total session bandwidth (W_1+W_2) settles at, and the variation from the beginning of the session. We observe that as the number of users increases, the optimal size of window W_1 settles at a higher value and W_2 reaches a higher value before decaying down to zero, because of the variation in the loss patterns of the receivers. However, the protocol scales well, with an overall increase of approximately %4 in total protocol bandwidth in response to a 1000-fold increase in user set.

VIII. CONCLUSION

In this paper, we have introduced a reliable multicast transport protocol that operates at the satellite-only tier of a hybrid satellite-terrestrial communication system for reliable delivery of data to end-users. The protocol effectively uses a special class of FEC codes and complements it with a dynamic window management scheme to adjust number of encoding packets forwarded to the end-users. Protocol scales well for a range of receiver group sizes. Protocol is lightweight, and uses only a single type of feedback information from the end-users for both reliability purposes and dynamic window adjustment. Simulation results are currently preliminary and more detailed analysis is currently in progress.

REFERENCES

[1] X. Zhou and J. S. Baras, "TCP over satellite hybrid networks: a survey," CSHCN Technical Report TR2002-15, University of Maryland, College Park, MD, 2002.

[2] J. Nonnenmacher, E. W. Biersack, and D. Towsley, "Parity-based loss recovery for reliable multicast transmission," *IEEE/ACM Trans. Networking*, vol. 6, no. 4, pp. 349-361, August 1998.

[3] S. Bhattacharyya, D. Towsley, and J. Kurose, "The loss multiplicity problem in multicast congestion control," in *Proc. of IEEE INFOCOM*, 1999.

[4] L. Rizzo and L. Vicisano, "A reliable multicast distribution protocol based on software FEC techniques," in *Proc. of HPCS*, 1997.

[5] S. Payne, "Reliable multicast via satellite," Master Thesis, University of Maryland, College Park, MD, 1999.

[6] K. Manousakis, "Reliable multicasting based on air-caching for flat hierarchy networks," Master's Thesis, University of Maryland, College Park, MD, 2002.

[7] J. Yoon, A. Bestavroz, and I. Matta, "Adaptive reliable multicast," in *Proc. of IEEE ICC*, September 1999.

[8] D. E. Gossink and J. P. Macker, "Reliable multicast and integrated parity transmission with channel estimation considerations," in *Proc. of IEEE GLOBECOM*, 1998.

[9] L. Rizzo, "On the feasibility of software FEC," DEIT, Tech. Rep., January 1997, http://www.iet.unipi.it/luigi/softfec.ps.

[10] J. W. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A digital fountain approach to reliable distribution of bulk data," Tech. Report, UC Berkeley, TR-98-013, 1998.

[11] V. G. Bharadwaj, J. S. Baras, and N. P. Butts, "Internet service via broadband satellite networks," CSHCN Technical Report TR1999-11, University of Maryland, College Park, MD, 1999.

[12] European Telecommunication Standards Institute, "Digital Video Broadcasting (DVB): DVB specification for data broadcasting," ETSI EN 301 192, v.1.3.1, 2003.

[13] M. Luby, "LT Codes," in *Proc. of IEEE Symposium on Foundation of Computer Science*, 2002.

[14] E. O. Elliot, "Estimates of error rates for codes on burst-noise channels," The Bell Systems, Technical Report, 1963.

[15] J. Slack and M. Rice, "Finite state Markov models for error bursts on the ACTS land mobile satellite channel," NASA ACTS Experiment Paper ACTS-96-046, 1996.

[16] E. J. Fitzpatrick, "SPACEWAY systems summary," *Space Communications*, vol. 13, pp. 7-23, 1995.

[17] J. Foerster and J. Liebetreu, "FEC performance of concatenated Reed-Solomon and convolutional coding with interleaving," IEEE 802.16 Broadband Wireless Access Working Group Report, June 2000.