A Cross-layer Approach for Throughput Enhancement in Unsteady Satellite Networks

Lin Cui · Xin Cui · Seok J. Koh

As a feasible mechanism against link-layer errors, Forward Error Correction (FEC) scheme has been suggested for satellite channel. However, the existing FEC scheme cannot lead to the optimal throughput gain of TCP when a satellite link is in an unsteady state. In particular, if FEC applies a larger amount of redundancy to recover the most corrupted packets, too much bandwidth has to be consumed to send the overlarge redundant information. On the contrary, if FEC uses a lower amount of redundancy in order to fully utilize the available bandwidth to send the more user data, a higher residual packet loss rate will be incurred. This paper introduces an effective cross-layer approach to overcome the drawbacks of the existing scheme. In the scheme, a maximum header scope is defined to cover all protocols' headers of every MAC frame, and the FEC decoder only identifies the integrity of the header scope and leaves the validation of frame payload to TCP. Thus a lower amount of redundancy can be used to significantly reduce the residual packet loss rate as well as the unnecessary consumption of the available bandwidth. Simulation result shows that the proposed scheme can significantly reduce the throughput jitter of TCP while the satellite link is in an unsteady state or when a non-optimal amount of redundancy is applied by FEC, compared to the existing approaches.

Keywords: FEC, TCP, BER, satellite link, corruption

I. Introduction

Satellite link is generally featured by long propagation delay, large bandwidth-delay product and high Bit Error Rate (BER). At present, the two types of satellites, Geostationary Earth Orbit (GEO) and Low Earth Orbit (LEO), are mostly used in the satellite networks. The one-way propagation delay is about 250 ms for the GEO satellite and 10~100 ms for the LEO satellite [1].

It is noted that the average bit error rate of satellite link ranges from 10^{-8} to 10^{-5}, and higher 10^{-6} to 10^{-2}[1], which are much higher than those of terrestrial links. Thus Forward Error Correction (FEC) scheme has been suggested for error recovery over satellite channels [3].

The main drawback of FEC is that it consumes some extra bandwidth to transmit the redundant information. Therefore, there is an optimal amount of redundancy to add for a certain level of BER, above which the end-to-end throughput of TCP will degrade [7],[8]. However, the spatial BER is easily influenced by random factors. It means that the BER sometimes varies at a wide range randomly, and the optimal amount of FEC redundancy cannot be a static value as well. This implies that we cannot tightly track the variation of BER in time, unless the satellite link is in a very steady state. Thus the bit errors cannot be completely recovered by FEC redundancy if a higher throughput is expected.

On the other hand, the conventional Transmission Control Protocol (TCP) tends to suffer from performance degradation in wireless networks, since any corrupted packets are also regarded as the indication of network congestion. Hence some corruption-aware TCP variants,

Lin Cui,Seok J. Koh: Kyungpook National University
Xin Cui: Shandong University

such as [10],[11], have been presented for this issue.

Based on above analysis, we argue that it may be preferred to apply a corruption-aware TCP variant over the FEC scheme with a lower amount of redundancy rather than the existing approaches (e.g., TCP Westwood+ [4], TCP SACK [5] , TCP Peach [6], etc., over the FEC scheme with a higher amount of redundancy) for the satellite network with a high BER. The core problem is that those corrupted packets will be directly discarded by FEC decoder before they are delivered to the TCP receiver, even if the corruption-aware TCP is used.

This paper introduces an effective variant of FEC and presents a cross-layer approach between the revised FEC and corruption-aware TCP to further improve the throughput performance of satellite network and reduce the throughput jitter resulted from the variable residual packet loss rate of the existing scheme. In the proposed scheme, a maximum header scope is defined to cover all protocols' headers of every MAC frame, and the FEC decoder only identifies the integrity of the header scope and leaves the validation of frame payload to TCP. Thus the frames with valid header scope can be still passed to a corruption-aware TCP variant of the destination without checking the field of Frame Check Sequence (FCS). Therefore, the corruption-aware TCP receiver can feed corruption information back to the sender so as to trigger the prompt retransmissions with unchanged congestion window (cwnd) at sender side.

With this scheme, even if the carried redundancy is not the optimal amount for every FEC codeword, the enhanced throughput and decreased throughput jitter could be expected because the residual packet loss rate incurred by corruption can be significantly decreased.

The rest of this paper is organized as follows. Section II briefly represents the related works. In Section III, we describe the proposed scheme. Section IV shows some simulation results and section V concludes this paper.

II. Related Works

Generally, FEC encoder fragments a frame packet into several code words and sends redundancy along with the original data, so that the corrupted original data can be recovered from the redundancy without retransmission if only the redundancy amount is enough.

The significant FEC example for satellite networks is BCH code. BCH encoder encodes k data bits into an n-bit code word by adding n-k parity checking bits for the purpose of recovering some errors. Given the length of a BCH code word, say $n = 2^m-1$ for any integer $m \geq 3$, we will have t (where $t<2^m-1$ and $n-k \leq mt$) that is the bound of the error correction. That is, BCH decoder can correct any error bits (burst or separate) less than t within an n-bit BCH codeword; otherwise, the whole frame packet, which might have been separated into several code words, won't be reassembled by FEC decoder. In general, we denote a specific BCH code as BCH (n, k, t) and denote its code rate as k/n.

In recent years, FEC has attracted the much attention due to substantial underlying error conditions of wireless/satellite channel. However, increasing the level of FEC redundancy has dual effects on end-to-end throughput performance. On one hand, it increases the achievable throughput since the throughput is a monotonic increasing function of the level of FEC redundancy. On the other hand, it decreases the effective channel bandwidth because the effective channel bandwidth is opposite to that level. Hence end-to-end throughput is maximized when the effective channel bandwidth becomes equal to the maximum achievable throughput.

To mitigate the throughput degradation due to overhead FEC redundancy, some studies have investigated the performances of cross-layer approaches to optimize the end-to-end throughput over wireless links with high BER.

LT-TCP [13] presents an end-end mechanism to separate the congestion indications from the wireless packet erasures, by exploiting Explicit Congestion Notification (ECN) [14]. This scheme is built on top of TCP-SACK and depends on SACK, and a dynamically adaptive FEC scheme is used so that redundancy can be tuned based on the measured error rate. In the scheme, the residual packet errors are tackled by an enhanced retransmission scheme using reactive FEC repair packets to complement proactive FEC and SACK retransmission. In addition, the scheme dynamically changes the MSS to tailor the number of segments in the window for optimal performance.

The study in [15] introduces a hybrid model between FEC and ARQ-SR (Automatic Repeat Request with Selective Repeat) at link level in order to counteract the drawbacks of both FEC and ARQ, which usually retransmits a complete packet to correct a small piece of erased data so as to increase the RTT (but consumes extra bandwidth only when packets are lost) and deteriorate the performance of TCP.

The hybrid scheme of FEC and ARQ-SR can (i) reduce the retransmission number of erased packets over wireless link, (ii) shorten the RTT and end-to-end delay, and (iii)

reduce the bandwidth wasted due to retransmissions as well as the amount of FEC redundancy, compared to the case when either of FEC or ARQ is used alone. This is done by tuning the parameters of a link-level hybrid FEC/ARQ-SR model so as to maximize the performance of TCP. A typical example of parameters to should be tuned is the amount of FEC redundancy and the maximum number of trials to allow ARQ to do before a packet is discarded in link layer.

L. Baldantoni, et al. [16] investigated the performance of an adaptive end-to-end packet-level FEC for recovering end-to-end losses. This scheme calculates the optimum ratio of redundancy according to the state of the connection, thus it can avoid the TCP back-off behavior.

On the other hand, from the perspective of transport layer some studies have been done against the corrupted packets, among which TCP HACK [10] is the original version of the corruption-aware TCP variants. TCP HACK uses two additional TCP options for data segment and duplicate acknowledgement (DUPACK). In particular, on reception of a data segment, TCP receiver first verifies the integrity of the whole segment by checking the inherent overall checksum. In case that the segment is corrupted, the receiver then identifies the integrity of the segment's header by verifying the additional header checksum contained in the TCP option field. In this way, once corruption only occurs in a packet's payload, the receiver can recover the available sequence numbers from the valid TCP header, and timely report it to the sender. Therefore, in wireless environment with a high BER, TCP HACK can prevent shrinking *cwnd* from corruptions.

However, TCP HACK [10] takes a real-time retransmission policy only, and cannot distinguish congestion from corruption by itself while the former occurs behind the later since all DUPACKs have the same cumulative ACK value.

TCP CAIAD [11] overcomes the problem of TCP HACK and introduces a new error and congestion control scheme using corruption-aware adaptive increase and adaptive decrease algorithm. In [11], the corrupted segments will be further processed by the transport layer of the receiver, and a duplicate ACK is used to explicitly indicate both a real-time corruption event and the congestion state of the link, by which the sender estimates the current corruption strength and increases the *cwnd* as per the different increments instead of entering fast recovery phase if only there is no unrecovered lost segments until then. Meanwhile, the slow start threshold of TCP CAIAD is estimated not only based on the amounts of the received integral packets but also based on the corrupted packets as per the interval of round trip time and only updated when the lost but not the corrupted segment is detected by the sender (since if the corrupted packets can arrive at the receiver side, they should reflect some available bandwidth at a certain extent as well). Therefore, TCP CAIAD [11] can estimate the network bandwidth more precisely and can significantly improve TCP performance over networks with wireless single hop.

III. Proposed Scheme

1. New protocol type value for the corruption-aware TCP variants

At present, any variant of FEC mechanism does not support the corruption-aware transport protocols, since all corrupted packets will be directly discarded by FEC decoder before they are delivered to transport layer.

To let FEC decoder cognize the type of the corruption-aware transport protocols, a currently unused type value can be assigned in related field of every frame header, e.g. "protocol type" field of the 802.3 MAC frame, while the frame is constructed at the sender side so as to show all intermediate nodes as well as link layer's mechanisms (including FEC decoder) that the payload of this frame eventually will be processed by a corruption-aware TCP and ask to skip the validating of FCS field. For this purpose, link layer's mechanisms may need a slight revision in order to cognize the new protocol type and extend the corresponding procedure to skip the CRC checksum for FCS. As a reference, the detailed structure of 802.3 MAC frame is shown in Figure 1.

2. Maximum header scope identified by FEC decoder

When a MAC frame arrives at the input of FEC mechanism, FEC encoder fragments the frame packet into several code words and sends redundancy along with the original data, so that the erased original data can be recovered from the redundancy without retransmission if only the redundancy amount is enough.

Therefore, we present here to classify the FEC code words into two classes, one is header class and the other is data class. All code words that compose of the frame header are classified into the header class, and the others are classified into the data class. For the code words of the header class, the ones with the erased bits more than error bound will be directly discarded, together with the

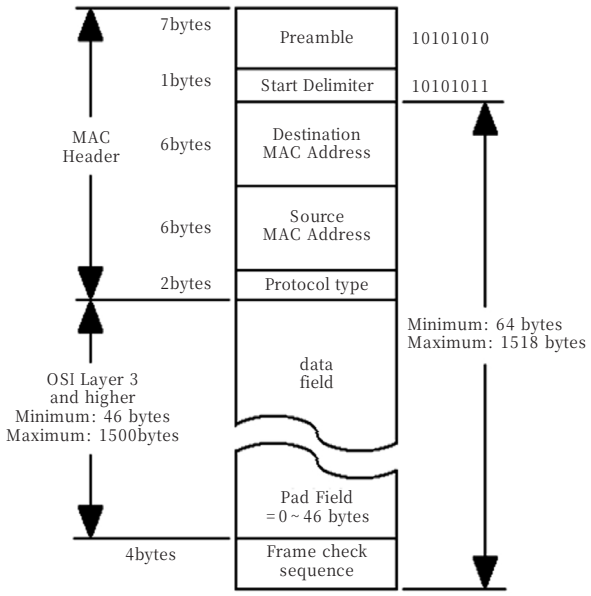

Figure1. The structure of 802.3 MAC frame

other code words belonging to the same packet (as does in existing FEC), since the packet header cannot be correctly recovered and may contain some wrong information. As for those code words belonging to the data class, even if they cannot be correctly recovered, the FEC decoder should still encapsulate them into the right packets that have the valid headers, and then still deliver the corrupted packets to the destination.

Nevertheless, since some IP headers may contain various option fields, the FEC decoder has no the capacity to cognize the detailed IP structure of every frame. Thus we have to define a possibly maximum header scope to cover all protocols' headers of each frame, and all code words that are fragmented from the header scope will be classified into the header class.

Further, we argue in this paper that the possibly maximum header scope should cover frame header, IP base header as well as the maximum length of IP option field, and TCP base header (where we do not take care of TCP option field, since the useful information, e.g. sequence number as well as checksum field, is contained in TCP base header).

In particular, if the FEC decoder lies on a wired network, the length of the possibly maximum header scope should be the sum of 14 bytes (frame header), 20 bytes (IP base header), 40 bytes (the maximum length of IP option field) and 20 bytes (TCP base header), which is equal to 94 bytes in total.

That is, as long as the FEC decoder correctly recovers the first 94 bytes' original data, the whole frame packet should be reassembled no matter whether the later part of this packet is corrupted or not. In this way, this scheme can significantly reduce the residual packet loss rate incurred by corruption, and those corrupted packets with the valid headers can be handled by the upper layer's corruption-aware TCP so as to feed the corruption information back to the sender.

3. Residual packet loss rates at the output of FEC

Typically a BCH code of FEC is denoted as (n, k, t), where n indicates the length of each code word in bit, k denotes the bit number of original data contained in a code word and t expresses the upper threshold of the fault-tolerance (in bit) for a code word.

Let T is the actual number of the erased bits occurred in a code word, and then the code word's recovery rate can be found as:

$$P[T \le t] = \sum_{i=0}^{t} \binom{n}{i} P^i (1-p)^{n-i} = \sum_{i=0}^{t} \binom{n}{i} \left(\frac{p}{1-p}\right)^i (1-p)^n \qquad (1)$$

where p is BER. Let $\alpha = (1-p)^n$, $\beta = p/(1-p)$, we can yield:

Table 1. The different packet loss rates with 10^{-3} BER

	BCH(255,215,5)	BCH(255,223,4)	BCH(255,231,3)	BCH(255,239,2)	BCH(255,247,1)
P_L	1.66E-5	3.86E-4	0.74%	10.91%	75.08%
P_H	1.16E-6	2.81E-5	5.63E-4	0.9%	10.52%

$$P[T \le t] = \alpha \sum_{i=0}^{t} \binom{n}{i} \beta^i \tag{2}$$

Given the packet size, say L, then the packet can be divided into $\lceil L/k \rceil$ code words, and the residual packet loss rate at the output of the existing FEC scheme can be written as:

$$P_L = 1 - P[T \le t]^{\lceil L/k \rceil} = 1 - \alpha^{\lceil L/k \rceil} \left[\sum_{i=0}^{t} \binom{n}{i} \beta^i \right]^{\lceil L/k \rceil} \tag{3}$$

On the other hand, if we apply the FEC with a lower amount of redundancy and leave the most corrupted packets with the valid headers for the corruption-aware TCP to handle, then only the least corrupted packets with the corrupted headers will result in losses.

In particular, given a BCH code of (n, m, t-λ), because $n = 2^x - 1$ and $x = \log_2(n+1)$, thus a BCH code word usually has to carry every $\log_2(n+1)$ redundant bits to recover each error bit in most cases. Therefore, we can have $m \approx k + \lambda \log_2(n+1)$.

In the same way, for a packet which has a total size L and a header scope H, the code word's recovery rate $p[T \le t\text{-}\lambda]$ as well as the packet loss rate P_H can be computed as follows:

$$P[T \le t-\lambda] = \alpha \sum_{j=0}^{t-\lambda} \binom{n}{j} \beta^j \tag{4}$$

$$P_H = 1 - \alpha^{\lceil H/(k+\lambda \log_2(n+1)) \rceil} \left[\sum_{j=0}^{t-\lambda} \binom{n}{j} \beta^j \right]^{\lceil H/(k+\lambda \log_2(n+1)) \rceil} \tag{5}$$

144-bit (1500 bytes' IP packet plus 14 bytes' frame header as well as 4 bytes' FCS field, that is, L value is equal to 1518 bytes) and the maximum header scope has a length of 752-bit (94bytes, H value), we can get the particular packet loss rates as per the formulas from (1) to (5). Table 1 lists the specific packet loss rates caused by the existing FEC (P_L) as well as the proposed scheme (P_H) under the environment with 10^{-3} BER:

Recall the earlier studies [2],[11] that when the packet loss rates are less than or close to the order of magnitude of 10^{-4}, the end-to-end throughputs are almost the same. In other words, it cannot significantly improve the end-to-end throughput to reduce packet loss rate much less than 10^{-4}, compared to the case of which the packet loss rate is close to 10^{-4}. As a result, all BCH codes listed in Table 1, except BCH (255, 247, 1), can be available for the proposed scheme while t must be larger than 3 for the conventional FEC mechanism.

That is, under the same utilized ratio of bandwidth resources, the proposed scheme can produce the almost same throughput, compared to those of the existing approaches. However, since a higher code rate's BCH code can improve the utilized ratio of bandwidth up to $\lambda \log_2(n+1)/k$, we can argue that the corruption-aware TCP with a higher code rate's BCH code (just with a lower amount of redundancy) could be a better choice for satellite network.

IV. Simulation Results

We use TCP CAIAD [11] as the corruption-aware transport protocol in our experiments and compare its performance with those of TCP SACK [5] and TCP Westwood+ [4] over the various BCH codes, using the ns-2 network simulator [9]. For each experiment, three connections are established through the same edge routers, terminals and GEO satellite, as shown in Figure 2.

where satellite link has the capacity of 2Mbps and the default propagation delay of GEO satellite link in ns2 simulator, while the terrestrial links have the bandwidth of 10Mbps and the propagation delay of 2ms. In addition, queue size of each intermediate node is set to 512 packets and each IP packet has a fixed size of 1500-byte. Also, we apply the associate packet loss rate over the satellite link while a particular BCH code is employed, based on Table 1.

In experiments, we perform the file transfer

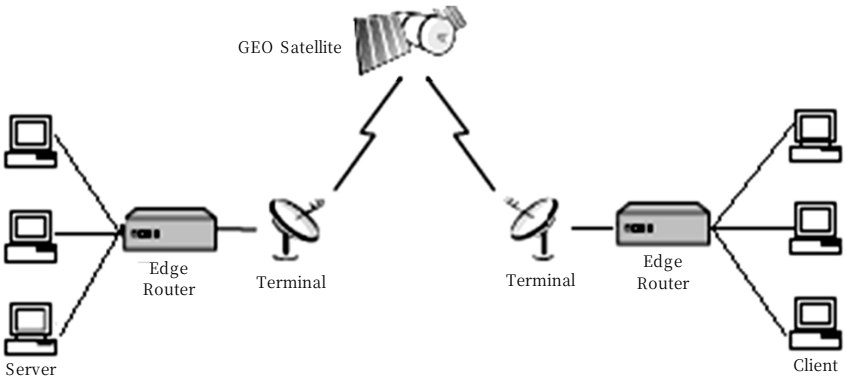

Figure 2. Simulation topology

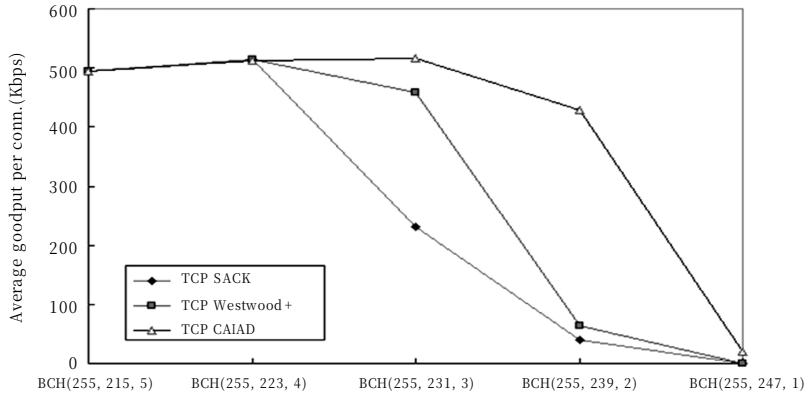

Figure 3. Traces of available goodputs with BCH code

application for 100 seconds and compare their average goodputs under the following test scenarios:

- 3 connections of TCP SACK with the traditional FEC;
- 3 connections of TCP Westwood+ with the traditional FEC;
- 3 connections of TCP CAIAD with the proposed FEC.
- Respective connections of the various TCP variants with the different FEC schemes through the same bottleneck satellite link.

1. Goodput performance with packet loss rate incurred by corruption

We first compare the average goodputs of TCP SACK, TCP Westwood+ and TCP CAIAD over 100 seconds with the different packet loss rate (P_H for TCP CAIAD and P_L for other TCP variants) given in Table 1. Figure 3 depicts the traces of the average goodputs as well as the corresponding BCH codes, based on the scenarios from 1) to 3), where three connections of the same TCP variant are established over the bottleneck satellite link, respectively.

From the figure we can see that the performance of the traditional approach highly depends on the amount of FEC redundancy, and drastically fluctuates as long as the level of FEC redundancy is changed a little. However, in practice it is very difficult to accurately tune the FEC redundancy to meet the level of BER in time since the spatial environment is easily influenced by random factors. Thus, we argue that the throughput jitter cannot be avoided effectively by the traditional approach which runs non-corruption-aware TCP over the standard FEC scheme, unless overmuch FEC redundancy is used (but this will lead to decrease of effective bandwidth).

In contrast, except BCH (255, 247, 1) code, all other BCH codes are available for the proposed cross-layer

Table 2. Fairness performance

TCP	BCH(255,215,5)	BCH(255,223,4)	BCH(255,231,3)	BCH(255,239,2)	BCH(255,247,1)
SACK	1.0	0.99	0.96	1.0	0.67
Westwood+	1.0	0.91	1.0	0.99	0.67
CAIAD	1.0	1.0	1.0	0.97	0.80

approach which runs corruption-aware TCP variant over the revised FEC scheme. In other words, the goodput fluctuation of the proposed scheme is much less than that of the conventional approach and the proposed scheme is much resilient to BER's fluctuation as well. This benefit may significantly improve the performance stability of the proposed scheme over an unsteady satellite link, compared to those of the existing approaches. Moreover, simulation results show that the proposed scheme can achieve the best end-to-end goodput with BCH (255, 231, 3) code in the simulations.

Such performance gain can be anticipated since ① the proposed FEC decoder only discards those MAC frames where the frame headers contain some unrecovered bit errors in order to significantly reduce the residual packet loss rate, and ② the proposed FEC decoder leaves the most packets which carry the valid header to be processed by the corruption-aware TCP so as to alleviate throughput jitter due to frequent deflation of *cwnd*. Thus, the proposed scheme can lead to the highly stable throughput performance even if the used t value is not optimal.

Besides, simulation results also show that in the case of a very high BER, e.g., 10^{-3}, the corruption-aware TCP alone cannot significantly improve the performance of TCP because of overmany lost packets caused by header corruption (see the goodput of the proposed scheme with BCH (255, 247, 1) code).

Finally, the goodput evolution curves demonstrate the conclusions of [7],[8]. That is, there is an optimal amount of redundancy to add, above which the end-to-end performance degrades instead of improving. This is because FEC has dual effects on TCP throughput performance. On one hand, it increases the achievable end-to-end throughput since FEC can recover a certain level of random bit errors; On the other hand, FEC decreases the effective channel bandwidth because FEC carries some redundant information.

2. Fairness performance

Fairness is an important evaluation of TCP performance, by which we can know whether a set of connections of the same TCP variant can share a bottleneck link fairly. In this paper, we evaluate each TCP variant's fairness performance using the fairness index function defined in [12] which is given in formula [6]:

$$F(x) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n\left(\sum_{i=1}^{n} x_i^2\right)} \tag{6}$$

where x_i is the goodput of the *i*-th TCP connection, n is the total number of TCP connections over the bottleneck link. The fairness index ranges from 1/n to 1.0, and 1.0 indicates a perfectly fair allocation of bandwidth.

We estimate the fairness indexes of TCP SACK, TCP Westwood+ and TCP CAIAD based on the same simulations as those of section IV.1. The detailed fairness comparison is given in Table 2.

From the table, it is noted that all TCP variants can achieve rather satisfactory fairness index in most cases, except the case of applying BCH (255, 247, 1) code, and the fairness performance is best while BCH (255, 215, 5) code is used over link layer.

Such phenomenon can be explained since when BCH (255, 247, 1) code is applied over the satellite link with 10^{-3} BER, FEC cannot recover the most corrupted packets due to the lower amount of redundancy. This can result in an overlarge end-to-end residual packet loss rate that will eventually lead to traffic starvation within all connections. Thus fairness evaluation can be highly influenced by any slight goodput change among the connections because the denominator value of formula (6) is too small in this case. In contrast, when a lower code rate's BCH code is employed, the end-to-end residual packet loss rate can be significantly decreased by FEC decoder in link layer, thus fairness performance can be evaluated properly.

Therefore, based on the simulation results, we believe that the proposed cross-layer scheme does not impact the fairness performance of any transport protocol.

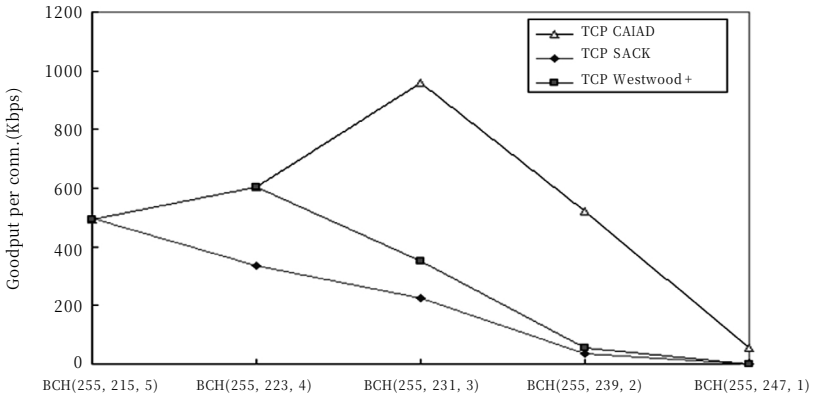

Figure 4. Traces of the goodputs of all TCP variants

3. Friendliness performance

Friendliness is another important evaluation of TCP performance. A friendly TCP scheme should be able to coexist with other TCP variants and does not cause traffic starvation of other connections or itself. To verify the friendliness performance of the proposed scheme, we establish the respective connection for the various TCP variants over the same satellite bottleneck link, and depict the traces of their goodputs in Figure 4.

By comparing Figure 4 with Figure 3, we can see that the goodputs of the proposed scheme are significantly improved in most cases in Figure 4, except the case of BCH (255, 247, 1) code. This is because the goodputs of the proposed scheme shown in Figure 3 are the average value of the same three connections, but they are not in Figure 4.

From the figure, we can also see that while a higher code rate's BCH code is employed, e.g. either BCH (255, 231, 3) or BCH (255, 239, 2), the goodput of the proposed scheme can get the mostly significant improvement, compared to the cases shown in Figure 3. This is because a higher code rate's BCH code, which carries a lower amount of redundancy, can result in the obviously raised packet loss rates in the existing schemes. Thus non-corruption-aware transport protocol cannot effectively utilize the bandwidth resources due to repeated deflation of congestion window. On the contrary, the proposed scheme can be able to fully utilize the spare bandwidth that is backed off by both TCP SACK and TCP Westwood+.

On the other hand, it is noted in Figure 4 that the goodputs of both TCP Westwood+ and TCP SACK are not significantly decreased compared to the cases shown in Figure 3, except the case of BCH (255, 223, 4) code (the reason will be explained later). This implies that the performance gain of the proposed scheme mainly comes from the spare bandwidth that cannot be used by the non-corruption-aware TCP variants due to the corrupted packets which are discarded by the conventional FEC scheme. In other words, the poor goodputs of TCP SACK and TCP Westwood+ are not caused by the proposed scheme, but by the higher corruption rates.

As an exception, while BCH (255, 223, 4) code is applied, the goodput of TCP SACK drastically decreases, compared to the case of section IV.1. However, it is noted that TCP Westwood+ has the same goodput compared to that of TCP CAIAD. Thus we can affirm that the performance degradation of TCP SACK is not caused by the aggressive policy of the proposed scheme, but by the adaptive decrease congestion control algorithm adopted by both TCP Westwood+ and TCP CAIAD. In particular, this is because TCP SACK halves its congestion window for any loss events, while both TCP Westwood+ and TCP CAIAD adjust their cwnds based on the measured rate of returning ACKs.

V. Conclusions

In this paper, we propose to enable the corruption-aware TCP over the BCH code of FEC so that there is no necessary to find out the optimal amount of FEC redundancy. Our first contribution lies in classifying the FEC code words into header class and data class, and differently processing them. Our second contribution lies in defining a maximum header scope, by which the intermediate nodes have no the necessary to cognize the detailed IP structure of every MAC frame passing through.

From simulation results, we can see that the proposed scheme not only can significantly reduce the throughput jitter but also can highly improve the goodput of TCP over the satellite link with a high BER, compared to the existing approach. Such performance gain is anticipated since the proposed FEC decoder only discards the frame packets of which the headers contain some unrecovered bit errors. Otherwise, the other corrupted packets with valid headers can be passed to a corruption-aware TCP variant, which can eventually trigger the prompt retransmissions with unchanged *cwnd* at the sender side.

Acknowledgement

This research was partially supported by the MKE (Ministry of Knowledge Economy) of Korea, under the ITRC support program supervised by the IITA (IITA-2008-C1090-0804-0004).

[References]

[1] Zhu, J, Roy, S, Kim, JH, "Performance modelling of TCP enhancements in terrestrial-satellite hybrid networks," *IEEE/ACM Transactions on Networking*, Vol. 14, No. 4, Aug. 2006, pp. 753-766.

[2] Xu K, Tian Y, Ansari N, "TCP-Jersey for wireless IP communications," *IEEE Journal on Selected Areas in Communications*, 2004, 22(4):747-756.

[3] M. Allman, D. Glover, and L. Sanchez, "Enhancing TCP Over Satellite Channels using Standard Mechanisms," RFC 2488, Jan. 1999.

[4] R. Ferorelli, L. A. Grieco, S. Mascolo, G. Piscitelli, P. Camarda, "Live Internet Measure-ments Using Westwood+ TCP Congestion Control," in *proceeding of GLOBECOM '02*. IEEE, Vol. 3, Nov. 2002, pp. 2583-2587.

[5] M. Mathis, J. Mahdavi, S. Floyd, A. Romanow, "TCP selective acknowledgment and options," *RFC 2018*, IETF, Oct. 1996.

[6] I. F. Akyildiz, G. Morabito, and S. Palazzo, "TCP-Peach: A new congestion control scheme for satellite IP networks," *IEEE/ACM Trans. Netw.*, Jun. 2001, pp. 307-321.

[7] C. Barakat and E. Altman, "Bandwidth tradeoff between TCP and link-level FEC," *Computer Networks*, Vol. 39, No. 5, Jun. 2002, pp. 133-150.

[8] B. Liu, D. Goeckel, and D. Towsley, "TCP-cognizant adaptive forward error correction in wireless networks," in *proceeding of IEEE GLOBECOM '02*, Vol. 3, Nov. 17-21, 2002, pp. 2128-2132.

[9] Network Simulator (ns-2), available from http://www. isi.edu/nsnam/ns/

[10] Balan, R.K.; Lee, B.P.; Kumar, K.R.R.; Jacob, L.; Seah, W.K.G.; Ananda, A.L. "TCP HACK: TCP Header Checksum Option to Improve Performance over Lossy Links," *Proc. IEEE INFOCOM 2001*, Vol. 1, Apr. 2001, pp. 309-318.

[11] L. Cui, S. J. Koh, X. Cui and Y. J. Kim, "Adaptive Increase and Adaptive Decrease Algorithm for Wireless TCP," *The 3rd International Conference on Natural Computation (ICNC'07)*, Vol. 2, Aug. 24-27, 2007, Haikou, China, pp. 392-398.

[12] R. Jain, D. Chiu, and W. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," *DEC, Res. Rep.TR-301*, 1984.

[13] K. K. Ramakrishnan, et al, "LT-TCP: End-to-End Framework to Improve TCP Performance over Networks with Lossy Channels," *In proceedings of IEEE 13th International Workshop on Quality of service (IWQoS)*, Passau, Jun. , 2005, pp. 21-23.

[14] K.K. Ramakrishnan, S. Floyd, and S. Black, "The Addition of Explicit Congestion Notification (ECN) to IP," *IETF RFC 3168*, Sep. 2001.

[15] C. Barakat and A. A. Fawal, "Analysis of link-level hybrid FEC/ARQ-SR for wireless links and long-lived TCP traffic," *Performance Evaluation Journal*, Vol. 57, No. 4, Aug. 2004, pp. 423-500.

[16] L. Baldantoni, H. Lundqvist and Gunnar Karlsson, "Adaptive End-to-End FEC for Improving TCP Performance over Wireless Links," in *proceeding of ICC 2004*, Jun. 2004.

Lin Cui

Lin Cui received B.S. degree in Electronic Engineering from Tianjin University, China, and M.S. degree in Computer Engineering from Kyunghee University, Korea, in 1989 and 2005, respectively. He is now as a Ph.D candidate with theDepartment of Computer Science in Kyungpook National University, Korea. His current research interests include Transport Layer Protocols, Wireless Communication and Internet Mobility.
E-mail: cuilin@cs.knu.ac.kr
Tel:+82-53-940-8608

Xin Cui

Xin Cui received B.S. degree in Materials Science and Engineering from Northwestern Polytechnical University, China, and M.S. degree in Interdisciplinary Program of Electronic Commerce from Chonnam National University, Korea, in 1986 and 2002, respectively. He has been an instructor in Shandong University at Weihai, China, since January 2003. His current research interests include development and application of Electronic Commerce, Networks Security, Wireless Networks and Transport Control Protocol.
E-mail: chlgms@sdu.edu.cn

Seok Joo Koh

Seok Joo Koh received B.S. and M.S. degrees in Management Science from KAIST in 1992 and 1994, respectively. He also received Ph.D. degree in Industrial Engineering from KAIST 1998. From August 1998 to February 2004, he worked for Protocol Engineering Center in ETRI. He is now an Associate Professor at Electrical Engineering and Computer Science in the Kyungpook National University since March 2004. His current research interests include Mobility Management for NGN, Internet Mobility and Transport Layer Protocols. He has so far participated in the International Standardization as an editor in ITU-T SG19, SG17, SG13, ISO/IEC JTC1/SC6 and IETF.
E-mail: sjkoh@knu.ac.kr