# TCP-Aware Forward Error Correction for Wireless Networks

Dzmitry Kliazovich, Magda Bendazzoli, and Fabrizio Granelli

DISI - University of Trento
Via Sommarive 14, I-38050 Trento, Italy
{kliazovich,granelli}@disi.unitn.it,
bendazzoli@telefin.it

**Abstract.** This paper studies TCP performance improvement in wireless and heterogeneous networks using Forward Error Correction (FEC) technique driven by TCP semantics. In the proposed scheme, called TCP-aware FEC, the amount of redundancy added to a packet at the sender node corresponds to the level of error protection and is computed as a function of TCP congestion window. The TCP-aware FEC becomes stronger for low congestion window values (experienced after a packet loss is detected), while the amount of added redundancy is reduced for the large window values approaching the capacity of the end-to-end link.

The design of TCP-aware FEC adaptability is driven by the mechanics of TCP congestion and flow control mechanisms, and it is based on the notion that link losses become more dangerous when congestion window and sending rate are low.

The proposed FEC scheme can be implemented either end-to-end as a part of TCP sender and receiver protocol stacks, or locally, covering the wireless part of the connection only. The performance results obtained through simulations confirm the design assumptions and underline the benefits of the proposed approach with respect to traditional FEC schemes[1].

**Keywords:** Adaptive FEC, TCP-aware.

## 1 Introduction

TCP is de facto standard protocol for communications in Internet [1]. Most of today's applications (including web access, file transfer, email, and even video streaming) demand TCP, mainly for its two major properties: reliable data delivery and congestion control. However, being designed almost 40 years ago for wired networks, TCP/IP reference model fails to deliver sustainable performance in heterogeneous network environments, which often includes wireless links [2].

Wireless links suffer from limited capacity, time-varying behavior, interference, etc. As a result, Bit Error Rate (BER) of the wireless medium ($10^{-3}$ – $10^{-1}$) is several

[1] This work is supported by Italian ministry in the framework of WORLD (Wireless multiplatfOrm mimo active access netwoRks for QoS-demanding muLtimedia Delivery) project.

P. Chatzimisios et al. (Eds.): MobiLight 2010, LNICST 45, pp. 68–77, 2010.

magnitudes higher than that of the wired links ($10^{-8}$ – $10^{-6}$) [3]. TCP interprets each loss as congestion-related and reacts by reducing its sending rate to a half of its size. The phase of congestion window increase is linear and takes considerable amount of time to compensate even a single such reduction of the sending rate.

The problems of TCP performance over wireless links are well-known [4]. The Forward Error Correction (FEC) technique applied to the TCP stream is one of the main solutions compensating the main problem, i.e. high error rate. FEC can be applied locally at the wireless link [4, 5] or at the end-to-end basis [6].

The FEC technique reduces packet loss probability at the expense of the bandwidth available for data transmission. While static FEC strength is commonly tuned to the worst error rate evidenced at the wireless link, dynamic control of the FEC strength, i.e. the amount of added redundancy, is required to maintain a reasonable TCP performance over heterogeneous networking scenarios.

Adding dynamic redundancy as a function of network state is proposed in [7-10]. Such schemes try to keep the amount of the redundancy added at the sender in accordance with error rate level estimated at the receiver side. As a result, the performance of dynamic FEC schemes depends on three components: wireless link error rate and accuracy of its estimation, end-to-end delay as measurements should be fed back from the receiver to the sender node, and bandwidth of the connection. These measurements are often inaccurate.

The design of forward error correction for TCP is shown to be dependent on the underlying random packet loss and distribution process. In [11], Loss-Tolerant TCP (LT-TCP) is proposed for extreme environments. It estimates the distribution of link errors and tunes its hybrid ARQ/FEC component accordingly. The authors of [12] propose a dynamic FEC assisted with cross-layer interaction from the link layer reporting the type of loss occurred. The approach proposed in [13] adapts channel estimation for the vehicular speeds, while the authors of [14] improve channel estimation by enabling Explicit Loss Notification (ELN) from network routers.

Summarizing, current proposals for dynamic FEC try to follow highly dynamic wireless channel state and, thus, struggle with inaccuracy of channel, end-to-end delay, and capacity estimation.

In this paper, we propose an FEC technique which adaptation is not a function of a network (or link) state, but a function of TCP mechanics. The proposed technique, called TCP-aware FEC, can be either end-to-end if implemented at the TCP sender or a local solution if it operates at the wireless link only.

The main design principle of TCP-aware FEC is that the amount of added redundancy is defined as a function of TCP window (or sending rate). This ensures tighter integration of FEC technology into TCP congestion control and helps to further improve the performance by increasing redundancy and protecting from error propagation into data segments with higher impact on TCP flow performance.

The rest of the paper is organized as follows: Section II provides the background on FEC and TCP flow and congestion controls, presents the core of the proposed approach and discusses its possible operation scenarios; Section III aims at presenting details of the simulation scenario and discussing the obtained results; Section IV concludes the paper outlining several points for the discussion and defining directions for future work on the topic.

## 2 TCP-Aware Forward Error Correction

Before presenting the core functionality of the proposed approach, it is important to understand the details of TCP congestion control algorithm and basic FEC operation.

### 2.1 Forward Error Correction

Forward Error Correction (FEC) is a technique allowing the data sender to add redundant data to the messages in order to help the receiver to detect and correct a limited number of errors in the received data flow without requesting for retransmission. At the packet level, FEC encoder operates with block codes, adding R redundancy packets to the block of K data packet and sending resulting N=R+K packets. One of the most widely used block error correction codes in wireless systems is Reed-Solomon code [16]. It allows correction of errors until their number does not exceed a half of the added redundancy code. This defines the tradeoff between the portion of bandwidth used for data transmission and the level of error protection offered.

For TCP, it is always desirable leaving the most of the bandwidth for data transmission until this level of bandwidth can be reached by TCP window evolution algorithm (under a given error rate on the channel). The dependence between TCP sending rate and channel errors are defined by the square root formula in [15].

However, it appears that the impact of transmission errors is not uniform and it depends on the position of erroneous packets within the TCP flow.

### 2.2 TCP Flow and Congestion Control

The most widely diffused version of TCP protocol in the Internet is TCP NewReno [17]. It employs Additive Increase Multiplicative Decrease (AIMD) as congestion window evolution strategy. TCP connection is initiated with congestion window ($W$) equal to one packet and Slow Start Threshold (*ssthresh*) set to a maximum possible value. Then, $W$ is increased by one for every received non-duplicate ACK, until the *ssthresh* is reached. This represents the slow start phase and its basic idea is to provide a fast (roughly exponential over time) window increase till the capacity of the transmission pipe is reached.

Once the capacity is reached and the packet loss is detected by three consecutive duplicate ACKs, $W$ is halved and TCP enters congestion avoidance phase. In this phase, the TCP sender gently probes the network for available bandwidth with a linear increasing window by $1/W$ for every TCP ACK received.

### 2.3 TCP-Aware FEC

Fig. 1 illustrates AIMD window evolution strategy and the proposed FEC strength protection. Stronger FEC is added for low TCP window values ($W_{min}$), which happen after congestion-related drop occurred, while weaker FEC is employed as the window approaches $W_{max}$ (which corresponds to the congestion window value before the last packet loss).

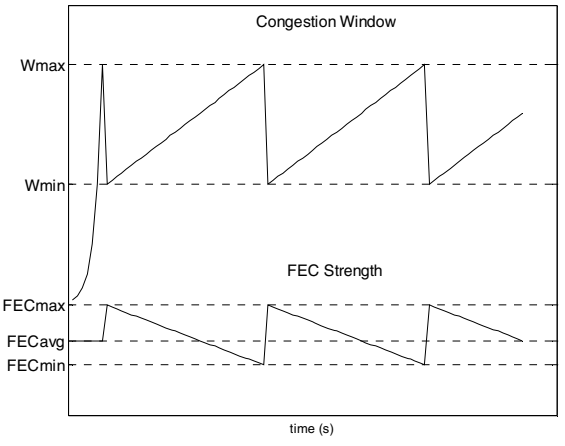


**Fig. 1.** TCP-aware FEC strength evolution

Any link-related packet loss will trigger unnecessary window reduction. However, for small window values, it will further decrease the throughput getting the flow away from available capacity. The actual value of the available end-to-end bandwidth is located between $W_{min}$ and $W_{max}$. As a result, when current window value is close to $W_{min}$, the number of link errors should be minimized by adding stronger FEC – allowing the flow to increase throughput faster.

On the other hand, when congestion window is close to $W_{max}$, TCP sending rate is higher than end-to-end link capacity. The exceeding portion starts filling up the buffers and the probability of experiencing congestion loss increases. In this situation, the level of FEC protection can be reduced, since a congestion loss (and not a link error) will be likely to occur.

Based on such reasoning, the proposed FEC linearly varies the amount of added redundancy between $FEC_{max}$ and $FEC_{min}$ values, i.e. reversely proportional to the TCP window evolution. The linear function is chosen to be in line with AIMD window evolution strategy. However, for other TCP implementations like BIC TCP [18] or Compound TCP (CTCP) [19], it could be adapted accordingly.

### 2.4 Implementation and Deployment Scenarios

Fig. 2 presents two possible scenarios for TCP-aware FEC implementation: end-to-end and local.

**End-to-end TCP-aware FEC** is implemented either as a part of transport layer or directly below it, both at the TCP sender and the receiver side. This is the most suitable scenario for TCP-aware FEC implementation. The FEC coder can easily obtain information about current value of the congestion window and have instant signaling notification in case of congestion-related loss is occurred from TCP layer.

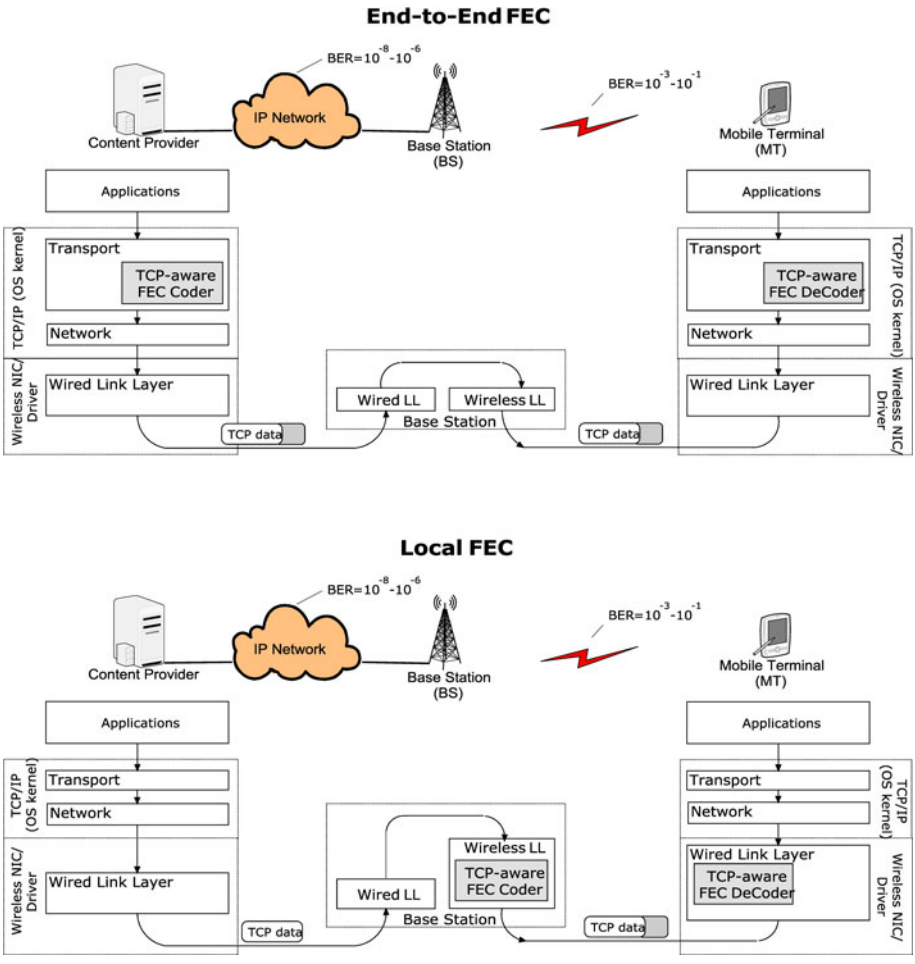


**Fig. 2.** TCP-aware FEC implementation scenarios

In this scenario, once the redundancy is added to the packet, it propagates to the receiver end without any modification – traversing fixed and wireless parts of the connection. At a first sight, this could be considered as a disadvantage, since no FEC protection is needed for the wired network (where error rate is low and added redundancy consumes unnecessary bandwidth). However, it appears that in most scenarios, the wireless last mile link represents the bottleneck of the end-to-end connection, while the bandwidth of the wired part remains underutilized. In fact, most of the proposals for coupling FEC and TCP operate in such end-to-end scenario.

**Local TCP-aware FEC** is implemented at the link layer and covers the wireless part of the end-to-end connection only. From the deployment point of view, this solution is more flexible, as it can be implemented within the wireless equipment (inside wireless cards or an OS driver) and be deployed and used where/when needed. In contrast,

end-to-end implementation would require insertion of TCP-aware FEC module into the part of the protocol stack (TCP and IP layers) implemented inside OS kernel.

A drawback of local implementation lies in the requirement for the base station to track TCP flow related parameters, such as current congestion window, which is not always trivial and may require availability of additional computational and storage resources at the base station.

Taking into account the above mentioned disadvantages, the end-to-end scenario is chosen as preferred for TCP-aware FEC implementation over the "local" scenario. Nevertheless, local implementation is also possible in several operating scenarios.

## 3 Performance Evaluation

In order to perform performance evaluation of the proposed approach, we extended the Network Simulator (ns-2) [20] with the required functionalities. TCP-aware FEC was implemented at the end-to-end basis and redundancy information was added to every TCP packet leaving the source node.

TCP NewReno [17] is selected to operate at the transport layer, and its goodput (i.e. the amount of data successfully delivered to the destination) is chosen as the main performance metric.

### 3.1 Simulation Scenario

Fig. 3 presents the details of the simulation scenario. It is composed of two content provider nodes $CP_1$ and $CP_2$, one router $R_1$, two base stations $BS_1$ and $BS_2$, and two mobile terminal nodes $MT_1$ and $MT_2$. The network nodes located in the wired part of the network are interconnected with 10 Mb/s, 2 ms links. The wireless links are configured according to the IEEE 802.11b standard, with the configuration parameters reported in Table 1. Buffers at the base stations $BS_1$ and $BS_2$ are limited in size to 500 packets.

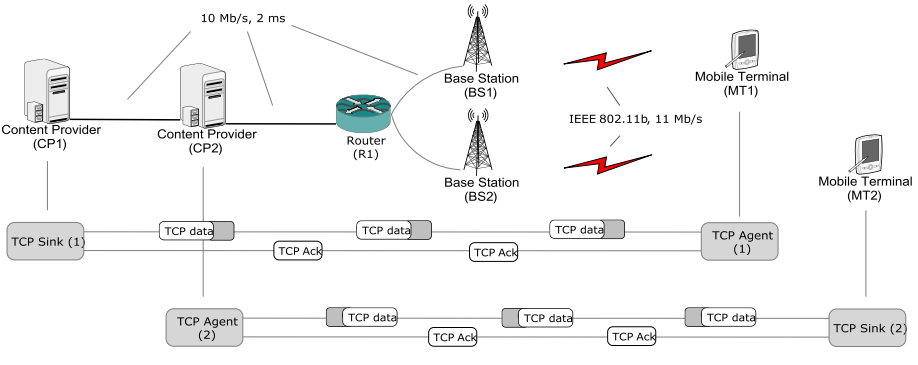


**Fig. 3.** Simulation scenario

**Table 1.** Simulation Parameters

| Parameter Name | Value |
|---|---|
| Slot | 20 μs |
| SIFS | 10 μs |
| DIFS | 50 μs |
| PLCP preamble + header | 192 μs |
| Data Rate | 11 Mbps |
| Basic Data Rate | 1 Mbps |
| Propagation Model | two-ray ground |

The wireless link error model follows uniform distribution, with BER in the range of $10^{-5}$ to $10^{-1}$. FEC strength is expressed as the number of bits that can be corrected per packet (which was in range between 0 and 6 bits).

The duration of each simulation is 250 seconds, and the results reported are averaged from 10 runs with 95% confidence interval.

Two configurations for TCP flow are considered: one originated at $CP_1$ and a destination $MT_1$, another between $MT_2$ and $CP_2$. Only one flow is active at any time during the simulation. The main idea behind having flows in two different directions is to test the proposed approach for different locations of bottleneck wireless link. In the first case, it is located at the end of the connection (last mile); while in the second case it is the link connecting TCP sender (first mile).

The results obtained for both flows are essentially similar and in line with the expectation. For that reason, only those obtained for the connection between $CP_1$ and $MT_1$ are reported.

### 3.2 Results

Fig. 4 presents the results of a TCP connection throughput for different FEC strengths (from 0 to 6 bits), when the correction is applied constantly (and the amount of added redundancy is stable for the duration of the experiment) as well as for the dynamic TCP-aware FEC.

For low error rates (BERs smaller than $10^{-5}$), all the schemes behave equally well with a minor difference in TCP throughput accounting for the overhead of added redundancy. However, when BER becomes higher than $10^{-5}$, the flows with low FEC strength start to rapidly decrease their throughput. As expected, the flow with the strongest FEC of 6 bits remains insensitive to errors longer than other flows.

The curve corresponding to the TCP-aware FEC lies in the middle. It outperforms fixed FEC solutions when FEC strength is low (less than 4 bits). However, falls short when the error rate becomes large.

The main advantage of using TCP-aware FEC is in its capability to preserve excessive usage of bandwidth for transmitting redundancy information. In TCP-aware FEC, the amount of added redundancy is the function of congestion window and thus it depends on the BER of the wireless channels.

Fig. 5 presents the amount of total redundancy added during the simulation time as percentage of the actual data flow for a wireless channel with BER equal to $10^{-3}$. It can be seen that TCP-aware FEC outperforms approaches with fixed FEC strength for all the evaluated values.

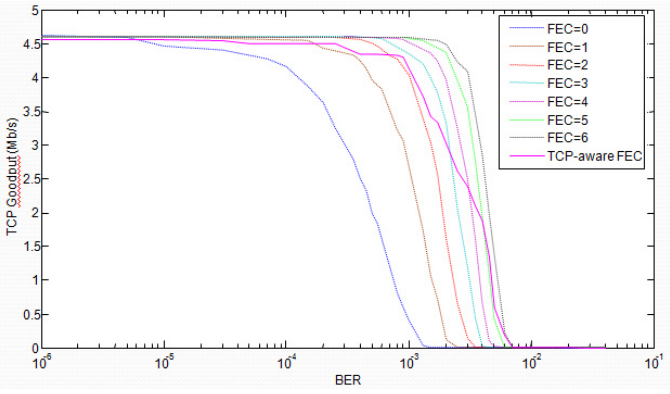


**Fig. 4.** TCP goodput performance for different FEC schemes in erroneous channels

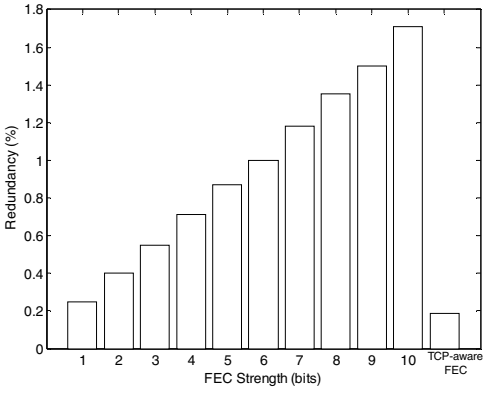


**Fig. 5.** Comparison of total redundancy added by different approaches

## 4 Conclusions and Future Works

This paper presents a novel approach for error correction TCP-aware FEC. The amount of redundancy added at the sender node to TCP packets is computed as a function of TCP window size. TCP-aware FEC becomes stronger for low window values experienced after a packet loss is detected and the window is reduced to its half, while for large congestion windows approaching end-to-end capacities the amount of added redundancy is reduced.

Driven by the TCP congestion and flow control mechanics, the proposed solution varies the level of error protection protecting TCP from incorrect (due to link losses) decisions for window reductions.

The proposed FEC scheme can be implemented either end-to-end as a part of TCP sender and receiver protocol stacks, or locally covering wireless part of the connection only.

Simulation results confirm an excellent tradeoff between the level of offered error protection and the amount of total redundancy added to the TCP flow for TCP-aware FEC.

Future directions for the approach will deal with implementation of the proposed technique as a patch for OS Linux kernel and extensive wide-area network experiments.

## References

1. Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T., Diot, S.C.: Packet-level traffic measurements from the Sprint IP backbone. IEEE Network 17(6), 6–16 (2003)
2. Bakin, D.S., Joa-Ng, M., McAuley, A.J.: Quantifying TCP Performance Improvements in Noisy Environments Using Rotocol Boosters. In: Fifth IEEE Symposium on Computers and Communications (2000)
3. Pentikousis, K.: TCP in wired-cum-wireless environments. IEEE Communications Surveys 3, 2–14 (2000)
4. Balakrishnan, H., Padmanabhan, V.N., Seshan, S., Katz, R.H.: A comparison of mechanisms for improving TCP performance over wireless links. IEEE/ACM Transactions on Networking (TON) 5(6), 756–769 (1997)
5. Barakat, C., Altman, E.: Bandwidth tradeoff between TCP and link-level FEC. Computer Networks 39(2), 133–150 (2002)
6. Lundqvist, H., Karlsson, G.: TCP with end-to-end FEC. In: International Zurich Seminar on Communications, pp. 152–155 (2004)
7. Baldantoni, L., Lundqvist, H., Karlsson, G.: Adaptive end-to-end FEC for improving TCP performance over wireless links. In: IEEE International Conference on Communications, vol. 7, pp. 20–24 (June 2004)
8. Park, K., Wang, W.: AFEC: an adaptive forward error correction protocol for end-to-end transport of real-time traffic. In: International Conference on Computer Communications and Networks (1998)
9. Tsugawa, T., Fujita, N., Hama, T., Shimonishi, H., Murase, T.: TCP-AFEC: An adaptive FEC code control for end-to-end bandwidth guarantee. Packet Video, 294–301 (November 2007)
10. Liu, B., Goeckel, D.L., Towsley, D.: TCP-cognizant adaptive forward error correction in wireless networks. In: IEEE Global Telecommunications Conference (GLOBECOM), vol. 3, pp. 2128–2132 (November 2002)
11. Subramanian, V., Kalyanaraman, S., Ramakrishnan, K.K.: An End-to-End Transport Protocol for Extreme Wireless Network Environments. In: Military Communications Conference, MILCOM (2006)
12. Baroudi, U., Abu Qadous, B.: An efficient adaptive Cross-Layer interaction mechanism for TCP traffic over heterogeneous networks. In: IEEE Symposium on Computers and Communications, pp. 118–123 (2008)
13. Ahmad, I., Habibi, D., Rahman, Z.: An Improved FEC Scheme for Mobile Wireless Communication at Vehicular Speeds. In: Telecommunication Networks and Applications Conference, pp. 312–316 (December 2008)
14. Miyoshi, M., Sugano, M., Murata, M.: Performance improvement of TCP on wireless cellular networks by adaptive FEC combined with explicit loss notification. IEEE Vehicular Technology Conference (VTC Spring) 2, 982–986 (2002)

15. Mathis, M., Semke, J., Mahdavi, J., Ott, T.: The macroscopic behavior of the TCP congestion avoidance algorithm. ACM SIGCOMM Computer Communication Review 27(3), 67–82 (1997)
16. Immink, K.A.S.: Reed–Solomon Codes and the Compact Disc. In: Wicker, S.B., Bhargava, V.K. (eds.) Reed–Solomon Codes and Their Applications. IEEE Press, Los Alamitos (1994)
17. Floyd, S., Henderson, T.: The NewReno Modification to TCP's Fast Recovery Algorithm. Request for Comments 2582, ETFC (April 1999)
18. Xu, L., Harfoush, K., Rhee, I.: Binary increase congestion control (BIC) for fast long-distance networks. INFOCOM 4, 2514–2524 (2004)
19. Tan, K., Song, J., Zhang, Q., Sridharan, M.: A compound TCP approach for high-speed and long distance networks. Microsoft Press, MSR-TR-2005-86 (July 2005)
20. The network simulator ns2, `http://www.isi.edu/nsnam/ns`