Modifications for TCP Improvements over Wireless Networks

Toni Janevski, *Senior Member, IEEE,* and Strahil Panev

Abstract — **In this paper we provide modifications of TCP for better performances in wireless networks. The improvements can be classified in several categories according to the problems they address. In the main section, a proxy based solution called Radio Network Feedback - RNF is implemented in NS-2 and its characteristics are compared to a standard TCP Reno. The control structure in the proxy sets the window size according to the information for the momentary radio bandwidth and the queue size. The results of the simulations show that there are several key areas where the RNF solution clearly outperforms widely used TCP Reno and increases the radio link utilization.**

Keywords — **Performance, Radio Network Feedback, TCP, Wireless network**

I. Introduction

THE popularity of Internet in recent years has driven the focus of mobile operators towards never-ending expansion of services and performance enhancements in the packet domain. The primary concern of the users is to gain maximal download speed and to minimize the response time. From operator's perspective there is one additional very important target: efficient utilization of their network infrastructure and the sacred bandwidth.

The connection between wireless access and Transmission Control Protocol is unique and full with challenges. TCP was initially developed for wired networks and links with fixed bandwidth and delays. The main problem of TCP in wireless networks is the wrong interpretation of the wireless losses as network congestion. This is the reason why TCP regularly and unnecessary enters Congestion Avoidance phase. The bandwidth adaptation is typically slow and the delay variations in the radio link trigger the TCP time-out mechanisms that lead to retransmissions and non-efficient usage of the available bandwidth. The usage of available resources is tightly coupled with "Bandwidth-Delay Product (BDP)" and TCP Sending Window (wnd). If wnd is lower than BDP there are moments when the window is full and the sender can not transmit more data although the available capacity allows at least one more packet to be transmitted. The combination of large wnds and long delays can have a negative impact on the retransmission mechanism. TCP needs at least one Round Trip Time (RTT) to detect a packet loss, but is unable to determine if the packets sent in the meantime are correctly received. Long RTTs slow the growth of wnd during the Slow Start phase. Connection outages are especially unfavorable and cause many time-outs because TCP does not have an algorithm for detection of the availability of the link. The only way for the sender to detect the availability of the receiver after an outage is to send probes which are exponentially divided and significantly increase the resume of the connection.

There are several approaches on how to address TCP problems in wireless environment. They all focus on a specific area and try to improve different parts of TCP characteristics. [1] - [3] propose an elegant Radio Network Feedback solution that addresses most of the challenges of TCP over wireless. This solution is implemented by adding a Proxy Server between the Radio Network Controller (RNC) and the remote server in operator's Core Network. The transport agent in the Proxy is aware of the varying conditions of the wireless channel and modifies its Congestion Window (cwnd) to adapt the transmission rate according to the available bandwidth. Also time-triggered information about the queue size in RNC is sent and the Proxy uses it to keep the desired queue size in the RNC constant. In this paper the parameters used in [1] are updated to match the existing HSDPA commercial networks. Also, different aspects are considered and parameters are plotted in the connection outage scenario and when transferring small files.

The rest of the paper is organized as follows. In Section II the existing TCP improvement solutions are presented together with RNF. Section III describes the simulation scenario. In Section IV the simulation results comparing RNF and TCP Reno are shown. Finally, Section 5 gives the conclusion and overview.

II. TCP improvements and RNF

All the propositions for TCP improvements can be divided in 3 basic groups: Link Layer, End-to-End and Split solutions. Link Layer solutions tend to correct the wireless link errors without involving the TCP layer. In other words, these solutions try to hide the error corrections from TCP. Typical example for this group is Snoop protocol. The basic idea is to implement an agent in the Base Station that "snoops" inside the TCP packets in order to get the sequence number, caches the unacknowledged

Dr. Toni Janevski is Professor at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Karpos 2 bb,1000 Skopje, Macedonia, (e-mail: tonij@feit.ukim.edu.mk).

Strahil Panev is MSc student at the Faculty of Electrical Engineering and Information Technologies, Ss. Cyril and Methodius University, Karpos 2 bb, 1000 Skopje, Macedonia, (e-mail: strahil_panev@yahoo.com).

segments and masks the wireless errors from the TCP sender. The great advantage of the Snoop protocol is the handover efficiency, but the shortcomings are higher processing power and memory usage. End-to-end solutions implement higher level of cooperation between the sender and the receiver to be able to distinguish wireless losses and network congestion. TCP Eifel is proposed to handle the problem of transmission ambiguity. After spurious time-out, the TCP sender continues to retransmit the segment considered to be lost. When ACK arrives at the sender, it does not know if this ACK refers to the retransmitted or the original segment. This is avoided by adding extra information in the ACK called TCP timestamp. Split solutions recommend splitting of TCP connection in two parts at the contact point between the wired and the wireless part because they have different characteristics and bandwidth. In UMTS this point would typically be situated in the Gateway GPRS Support Node (GGSN) in the operator's network. Software agent is implemented in the GGSN that uses standard TCP towards the remote server and different connection oriented transport protocol to communicate with the User Equipment (UE). As a result of this, the TCP connection towards the remote server is hidden from the changes in the wireless link.

The control structure proposed in [1] belongs to the group of Split TCP solutions. The functionality is achieved by adding a Proxy server situated between the RNC and the remote server. The proxy controls the transmission rate by accommodating its wnd, based on the information received from RNC. Proxy incorporates event-triggered feedforward mechanism based on bandwidth changes and time-triggered feedback mechanism based on the queue size in RNC. The aim is to achieve higher usage of the radio link and to keep the queue size in the RNC close to a wanted q_{ref} value. To address the problems that might appear in proxy buffer, due to the variations of the available bandwidth, a simple proxy–server control algorithm is implemented. This control is used to speed up or slow down the TCP sender depending on the length of current buffer queue in the proxy by using the receive window (rwnd) parameter in the TCP of the proxy. The RNF solution does not require any changes in the mobile terminal or the remote server.

III. Simulation Scenario

Fig. 1 shows the simulation scenario used in this paper. The mobile terminal UE from the right is downloading a file from the remote server on the left. The proxy resumes the role of a gateway towards the external world. Two TCP connections exist: the first is between the UE and the proxy and the second one between the proxy and the server. The communications of the end elements towards the proxy are using standard TCP Reno protocol and the proxy has implemented functionality to adapt its rate according to the information provided by the RNC. The RNC sends the RNF message every time the bandwidth changes (bandwidth information) and periodically every one second (queueinformation).

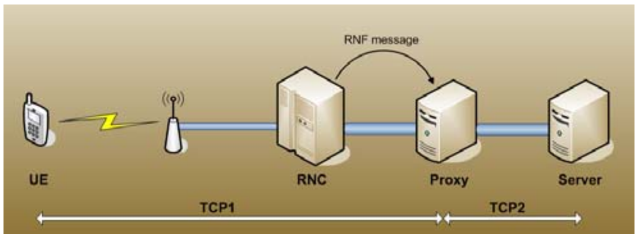

Fig. 1. System architecture

In a typical scenario, the UE requests a file from a server. First, the initial TCP packet (SYN) for establishing a connection is received in the proxy and response (SYACK) is sent back to the mobile terminal (TCP1). The proxy then establishes a connection with the remote server and transfers the file request from the client (TCP2). After receiving this message, the remote server starts sending the file following a traditional TCP algorithm. The whole transferred information from the server is buffered in the proxy and then the proxy adopts its parameters to make the TCP1 connection with maximized throughput according to the momentary bandwidth conditions in the wireless link.

IV. Simulation Results

Network Simulator 2 (NS-2) is used to test the performance of the RNF mechanism. The radio access network of the operator is represented only by the RNC which is completely modeled in Tool Command Language (Tcl). In the simulations the proxy solution is compared to a standard TCP Reno and the improvements are analyzed. The simulated scenario is given in Fig 1. Several C++ modules are added to model the proxy behavior in NS-2.30. RNFProxy agent in the proxy is the modified TCP and it is responsible for adapting the parameters of the TCP connection between the proxy and the terminal. Two sets of simulation scenarios are implemented with Tcl: transfer of large file and transfer of small files. The most important parameters of interest are the transmission rate and the queue length in the RNC. Table 1 gives the link bandwidth and the delays between the nodes in the scenario. The initial processing delay in the terminal is 15 ms and the desired queue length in the RNC is 15 packets. The Slow Start Threshold (ssthresh) for TCP Reno is 50 packets. In both cases buffer size is not limited in the RNC.

Table 1: Link bandwidth and delay

Node	**BW (Mbps)**	**Delay (ms)**
Terminal –RNC	2	25
RNC–Proxy	10	6
Proxy–Server	10	15

TABLE 2: BANDWIDTH CHANGE

Time (s)	BW (Mbps)
1	3,5
3,5	5,25
5	3,5
7,5	0
22	4,5

A. Transfer of large files

In the first scenario RNF and Reno are compared for transfer of 15 MB file while the radio link bandwidth changes. The changes are given in Table 2. The focus of the analysis is at the beginning of the connection, during the bandwidth change and after connection outage.

Fig. 2 gives the result of this simulation. There are several phases that contribute for the lesser time that RNF needs to download the file in comparison with Reno. First, at the beginning of the connection TCP Reno does not have the information for the available bandwidth so it has to start with the lowest rate possible. Proxy receives the bandwidth message and adapts its sending rate more quickly. Also it has to be noted that at the very beginning of every connection RNF also needs some time to begin to follow the maximal available bandwidth. The reason is that proxy-terminal connection is initially limited by the proxy-server connection that remains in slow start until the server's sending rate becomes greater than the proxy-terminal bandwidth. Secondly, during the bandwidth change RNF has slightly better performance than Reno. The biggest advantage of RNF is in the response time after connection outage. In the RNF case a simple outage does not affect the performance at all. The behavior of Reno is seriously degraded because of the Exponential Backoff algorithm. After the outage probes are sent to the terminals, which are exponentially separated. Fig. 3 shows linear dependency of the download time from the outage time for RNF, but for Reno the time needed for resuming the connection after an outage follows an unfavorable non-linear curve.

Fig. 4 represents the queue length in the RNC for RNF and Reno. The RNF mechanism has built-in capability to maintain an almost fixed buffer length at RNC. In the first part of the connection, Reno is unable to cope with the variable bandwidth and buffer needs are varying and they are huge. In the second part when the bandwidth is constant, the characteristic saw-tooth behavior of TCP can be seen. TCP Reno has higher memory demands which can make the system more expensive. Even if the buffer size is not an issue, longer queues increase the response time and the handovers are more difficult. Fig. 5 shows the download time when the buffer size at RNC is limited. Desired queue length coded in the RNF agent is 5 packets. For buffer lengths higher than 5, which is the valid area for analysis, RNF has lower buffer needs. For buffer size higher than 15, the download time for RNF is constant and in the Reno case further increase of the buffer reduces the download time.

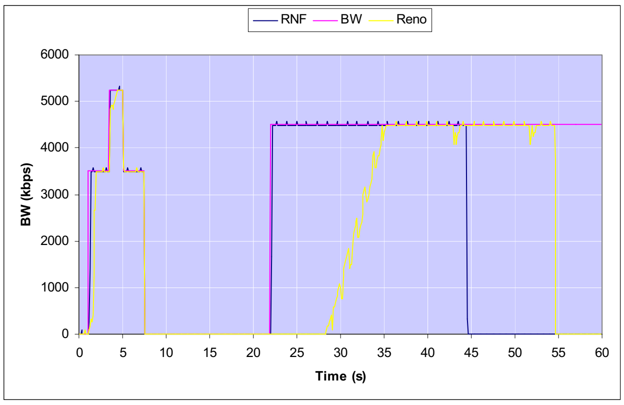

Fig.2 Bandwidth change

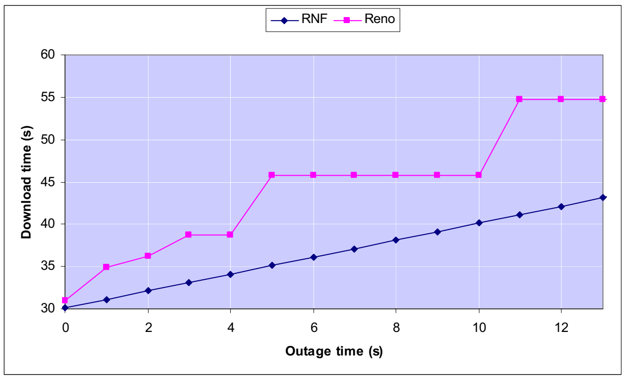

Fig.3 Download time when the outage varies between 1s and 13s.

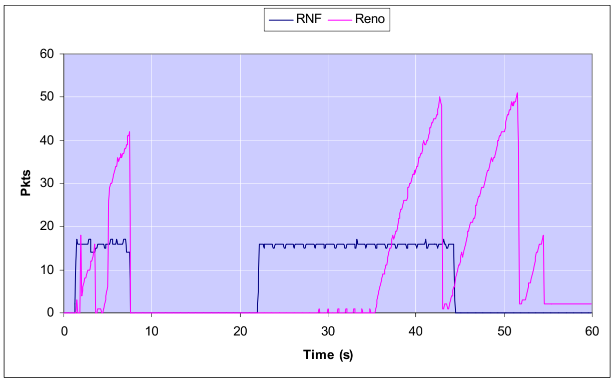

Fig. 4 Queue length in RNC

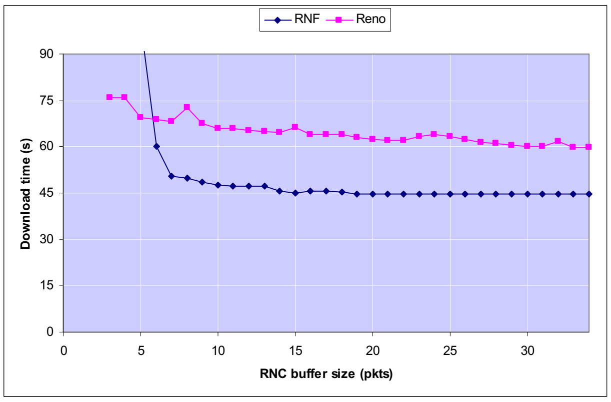

Fig 5. Download time for buffer size in RNC

B. Transfer of small files

The transfer of small files can be equated to web browsing. In our scenario the UE downloads 3 files with 500kB, 320 kB and 270 kB. The time between user clicks is 28s. Also, at the moments of transfer, a variable bandwidth is introduced for both RNF and Reno. The total transfer-time for RNF is 57s and for Reno is 58.3s which means an increase of 2.23% for RNF. But, this number includes the time the user spends viewing the page. If we exclude this time and measure only the transfer time, then for RNF it is 3.6s and for Reno it is 4.9s. The improvement is 26.53%.

From operator's point of view this makes considerable difference. The main contributing factor for the improvement is the fact that Reno remains in slow start for most of the connection.

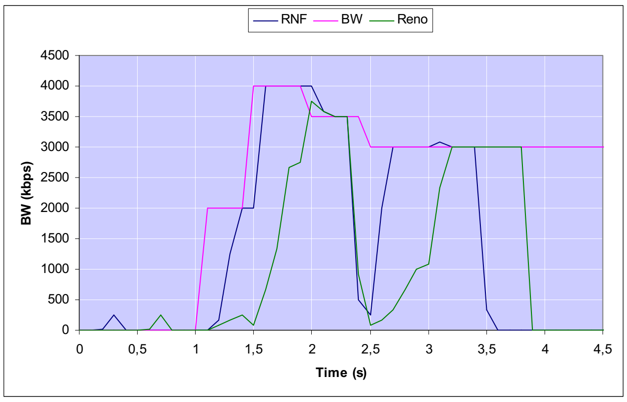

Fig. 6 User requests second file, before the first file transfer is finished for RNF and Reno, respectively

Especially interesting case for study is requesting a second web page before the first one is completely downloaded. This is shown on Fig. 6. The focus of this analysis is the measured reaction time after the user clicks on another link after 1.25s after the opening of the first link which is not enough for complete transfer of the first file. The total transfer time is 2.5s for RNF and 2.9s for Reno. RNF manages to avoid the two slow start phases which results in 0.4s faster download time. But if we consider the response time only, it is almost the same for RNF and Reno. The reason for this is the length of the queue in the RNC. RNF has the ability to keep the buffer length constant, but this is the stale data length at the moment of the second page request. In Reno's case, the TCP sender transmits with lower rates than the available bandwidth and this implies lower queue length in the RNC. If the connection runs for some time, then there is a high probability that the buffer at the RNC is full. During the small file transfer, Reno limits the sending rate because of the slow start algorithm. This results in lower usage of the available bandwidth, but also lower response times. If the user clicks on a different link when the connection is in more advanced state, longer RNC queue lengths are expected and also longer reaction times.

V. Conclusion

A TCP modified version called Radio Network Feedback was benchmarked with standard TCP Reno for transmission rate and queue length at RNC. The solution in [1] was updated with the latest parameter values of the existing commercial HSDPA networks. Also, other interesting aspects were analyzed with NS-2 that further proof the dominance of RNF over other transport protocols.

Although RNF shows improvement in almost all areas, the greatest advantage is in the case of connection outage. The information for the current bandwidth helps RNF to avoid the Exponential Backoff algorithm that TCP Reno enters. The buffer needs for the proxy solution are much lower than TCP Reno. But, contrary to expectations, the proxy setup does not seem to improve significantly the response time when a new page is requested in the middle of a download. The reason is the queue length at the moment of the second user click (stale data), which in the case of Reno is lower, consequently leading to lower response time.

From user's point of view, RNF reduces the time to serve the user and decreases the response time. Operators have the benefit of higher bandwidth usage and lower buffer needs at RNC. The proxy server is easily implemented in the operator's network and the operator has full control over it. All the necessary changes are done in the Core Network, and UE and the remote server are unaware of the modification. This kind of solution needs simple operation and maintenance and supports higher number of users at approximately highest transmission rates.

References

[1] N. Möller, I. C. Molero, K. H. Johansson, J. Petersson, R. Skog, and Å. Arvidsson, "Using Radio Network Feedback to Improve TCP Performance over Cellular Networks". Proc. of the 44th IEEE Conference on Decision and Control, 2005.
[2] M. Fiorenzi, D. Girella, N. Möller, Å. Arvidssony, R. Skogy, J. Peterssony, P. Karlssony, C. Fischionez, and K. H. Johansson, "Enhancing TCP over HSDPA by Cross-Layer Signalling", GLOBECOM, 2007
[3] Niels Möller, Åke Arvidsson, Justus Petersson, Carlo Fischione, Robert Skog, Patrik Karlsson, and Karl H. Johansson "Supporting end-to-end applications over HSDPA by cross-layer signaling", 2008
[4] Mohamad Assaad, Djamal Zeghlache, "TCP Performance over UMTS-HSDPA systems", 2006
[5] Liang Hu, "Evaluation of End-to-End TCP performance over WCDMA", Technical University of Denmark, Lyngby, Denmark, 2004
[6] M. Assaad, D. Zeghlache, "Cross-Layer design in HSDPA system to reduce the TCP effect", 2006.
[7] Möller, K. H. Johansson, and H. Hjalmarsson, "Making retransmission delays in wireless links friendlier to TCP", 2004.
[8] G. Xylomenos, G. C. Polyzos, P. Mähöonen, and M. Saaranen, "TCP Performance Issues over Wireless Links", 2001.
[9] 3GPP, available at http://www.3gpp.org.
[10] "The ns Manual" - The VINT Project - May, 2010