Andreadis et al. EURASIP Journal on Wireless Communications and Networking (2016) 2016:191
DOI 10.1186/s13638-016-0695-0

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

# A cross-layer jitter-based TCP for wireless networks

Alessandro Andreadis*, Sandro Rizzuto and Riccardo Zambon

**Abstract**

The Transmission Control Protocol (TCP) is one of the main communication protocols in the Internet, and it has been designed to provide an efficient reaction to packet loss events which are due to network congestion. Congestion is the main cause of losses in wired networks, but in today heterogeneous networks, loss events can also be introduced due to higher error rates on wireless channels, host mobility, and frequent handovers. Unfortunately, all packet losses are interpreted by TCP as a sign of congestion, triggering an inappropriate reaction which reduces its transmission rate and leads to performance degradation. In order to avoid this problem, it is important for TCP to correctly understand whether the reason of a packet loss is due to congestion or to a problem in the wireless link. This paper presents an innovative jitter-based cross-layer TCP algorithm, named XJTCP. It adopts the jitter ratio as loss predictor, joined with a layer two notification, in order to correctly infer the nature of a loss event. Performance evaluation and comparison with other common TCP implementations shows how XJTCP can be an interesting solution in the presence of wireless environments.

**Keywords:** Wireless TCP, Fairness, QoS, Performance evaluation

## 1 Introduction

The Transmission Control Protocol (TCP) is a reliable transport protocol widely adopted in today Internet. It adopts a suitable end-to-end congestion control scheme, providing high robustness against congestion losses, which constitute the main problem during packet transmission through wired networks [1]. In presence of heterogeneous networks including wireless connections, packet losses can be caused not only by congestion events but also by random or burst errors on the wireless channel. Nevertheless, TCP always reacts in the same way: it halves down its congestion window and triggers a backoff of its retransmission timeout (RTO). Such a reaction can lead to performance degradation when wireless links are present in the end-to-end path. This is the main reason why researchers have proposed several modified TCP protocols, at the aim to drive appropriate reactions when different types of losses are experienced [2, 3] and, consequently, to improve TCP performance.

The different solutions proposed in literature can be categorized according to three broad approaches: split-connection, link-layer, and end-to-end approach [4, 5]. Split-connection protocols break TCP connection into two parts, at the base station; link-layer variants provide reliability through local retransmissions; finally, in end-to-end approaches, the sender performs loss recovery without breaking the semantics of TCP.

Even if each methodology has several advantages and disadvantages, this work focuses on the end-to-end approach because it does not involve any congestion explicit signaling from the network elements; therefore, it is highly deployable, with easy installation and utilization, and it requires modifications only in the terminal devices, without direct involvement of intermediate nodes.

This paper introduces an innovative modification to the TCP protocol, named XJTCP, which is based on a cross-layer variant of the jitter-based TCP (JTCP) [6]. In particular, XJTCP adopts the inter-arrival jitter as loss predictor in order to infer whether a TCP packet loss is due to congestion or to random error. Moreover, it introduces a cross-layer mechanism for taking into account the discarded medium access control (MAC)

* Correspondence: andreadis@unisi.it
Department of Information Engineering and Mathematics, University of Siena, Via Roma 56, 53100 Siena, Italy

frames, with the aim to enhance TCP performance over wireless networks.

The paper is organized as follows: Section 2 introduces the main problems affecting TCP protocol on wireless networks. Section 3 provides a detailed description of the proposed XJTCP algorithm and of its key features. Section 4 describes the simulation environment. Section 5 reports performance comparisons with popular solutions such as Reno [1], JTCP [6], and Venoplus [7], under different loss typologies and congestion states. Finally, the last section concludes the paper with final remarks.

## 2 TCP issues over wireless network

TCP was originally designed for wired networks, where congestion is the main cause of losses and non-congestion loss events are usually negligible. Consequently, when a packet loss occurs, TCP efficiently reacts by reducing its congestion window size and retransmitting the lost data segment [1]. However, the Internet of today is characterized by end-to-end paths which often include wireless links affected by higher error rates and variable delays [8]; the wireless channel can generate non-congestion losses, due to either random bit errors or burst errors [2].

Random bit errors are often due to unpredictable effects such as shadowing, fast fading, or interference, which can cause bit flipping in the transmitted data. When the MAC layer is not able to recover such errors, the received erroneous frame is discarded and TCP congestion control mechanism comes into play. Moreover, wireless channel condition is highly variable by nature, and this provokes delay fluctuations of the round-trip time (RTT), intended as the time between one packet is sent and its ACK is received. Since the TCP RTO is estimated on the basis of previous samples of the RTT, when such fluctuations exceed the estimated RTO, the in-flight packet is prematurely considered as it was lost, even if it is correctly received after the RTO deadline (i.e., spurious timeout), and an unnecessary retransmission is triggered (i.e., spurious retransmission). Spurious retransmissions have two negative effects: first, they cause waste of bandwidth and secondly, they do not minimize energy consumption and that is an important issue due to limited power availability in wireless and mobile devices [9].

Burst errors refer to a contiguous sequence of erroneous bits, which are caused by persistent channel problems, such as prolonged interference, long fading events, handovers and disconnections often due to user mobility, and correlated packet losses are experienced. Consequently, TCP congestion control throttles its sending rate and reduces significantly its throughput.

In this environment, TCP inability to distinguish the actual reason (i.e., congestion or not) for a packet loss can lead to inappropriate reaction to a false congestion state, which results in wasteful behavior and unnecessary throughput decrease. Another key issue contributing to the degradation of TCP performance is packet reordering, which is not such a rare event in wireless networks [3]. It happens when the receiving order of TCP packets differs from the sending order. High channel error rates in wireless links activate link-layer retransmissions which can lead to out-of-order delivery. The packet reordering problem violates the near in-order assumption made in the design of many traffic control mechanisms (i.e., duplicate acknowledgments), introducing several unnecessary retransmissions.

Given these issues, several TCP solutions have been proposed by researchers in order to improve TCP performance over heterogeneous networks [10]. At the time of writing, cross-layer approach can provide interesting solutions for TCP over wireless, since this approach does not violate the end-to-end semantics of TCP, by embedding all the modifications inside the sender and/or the receiver stack. On the other hand, it introduces more computational complexity, due to the interaction between MAC and transport layers.

Nowadays, one of the most diffused TCP implementations is TCP Reno, which introduces fast recovery in conjunction with fast retransmit in order to avoid slow start phase after a single segment loss. Fast recovery is performed after receiving three duplicate acknowledgments (DACKs). Furthermore, the sender uses additional incoming DACKs to increase the congestion window, exiting the fast recovery phase when a unique ACK related to new data is received. Then the congestion window is set to half of the current congestion threshold and the DACK counter is set to zero. Reno's fast recovery is optimized for the case when there is a single segment loss within a congestion window [1]. Hence, it suffers from performance problems when multiple segments are lost in the same congestion window.

Jitter-based TCP (JTCP) adopts the jitter ratio "Jr" as a loss ratio predictor to determine the congestion level of the end-to-end path [6]. The jitter ratio is derived from the inter-arrival jitter between segments, as defined in real-time protocol (RTP). This information is deployed to investigate the link status, so as to take appropriate action on congestion window when a loss is detected. If the sender receives three DACKs or a RTO occurs, the value of Jr is compared with the inverse value of current congestion window size to distinguish between congestion and non-congestion losses. Then, if the loss is inferred as congestion loss, a Reno response is performed; otherwise, the congestion window is slightly reduced in order to maintain high throughput.

TCP Veno [11] joints the Reno mechanism with the channel estimation introduced in TCP Vegas, so as to distinguish between congestive and non-congestive losses. It adjusts the slow-start threshold according to the perceived network-congestion level. Specifically, it adopts a parameter $\beta$ in order to infer congestion in the network: if the estimated congestion level exceeds the value of $\beta$, the loss is considered due to congestion and Reno additive-increase multiplicative-decrease (AIMD) scheme is applied. If the loss is assumed as due to random errors, Veno reduces the congestion window of 1/5.

TCP Venoplus [7] derives from Veno algorithm and it introduces two important refinements in order to distinguish between congestion and random losses. Specifically, two new variables, congestion loss window, and random loss rate are adopted. Congestion loss window is incremented at each triple DACKs or timeout. Venoplus incorporates cross-layer functionality: it retrieves the received signal strength information (RSSI) and calculates BER for every packet received by the MAC layer, in order to evaluate the number of packet losses due to random errors in the wireless channel [7]. At each TCP segment loss, congestion loss window and random loss rate variables are used to opportunely reduce the congestion window (Cwnd). It infers segment loss due to congestion and random error (i.e., halving Cwnd value), due to random errors (i.e., Cwnd × 4/5), or due to congestion (i.e., halving Cwnd value). With respect to TCP Veno, TCP Venoplus improves the accuracy of congestion loss identification, providing significant enhancement on performance.

The described end-to-end solutions try to distinguish congestion from random losses on the basis of some indications inferred at the transport layer, such as DACKs, timeouts and delay, or jitter, and they regulate their reaction by opportunely dimensioning the TCP congestion window. The algorithm here proposed still represents an end-to-end approach, thus not breaking the semantics of original TCP, but it also considers feedback which originates from the MAC layer and is communicated to the transport layer. Due to this cross-layer signaling, the new TCP scheme can constantly monitor the channel status and it is aware of channel errors and related frame loss rates; consequently, it tries to gather the reason of packet loss, being it due to congested network or to channel impairments. The knowledge of the status of the wireless link at the moment in which the loss event is detected allows to better adjust the degree of TCP reaction and to better tune its congestion parameters. For this reason, the choice of a suitable algorithm capable of distinguishing whether network congestion or channel corruption is predominant in a wireless network appears of paramount importance, in order to achieve good performance in terms of throughput, delay, and jitter.

## 3 XJTCP algorithm

The key idea of the proposed algorithm is to integrate the jitter-based error estimation introduced in [6] with a cross-layer approach. Specifically, the XJTCP algorithm continuously monitors the wireless channel error status, by counting the number of inferred congestion and non-congestion loss events.

According to this monitoring activity, the congestion control scheme takes an appropriate decision about TCP window size reduction and retransmissions. In particular, when three DACKs (TDACKs) or a RTO are detected, the algorithm evaluates whether it is a congestion or non-congestion loss and it adjusts the congestion window opportunely.

### 3.1 Channel monitoring

In order to monitor the wireless channel status and the nature of loss events, two variables are introduced, namely, $C_i$ and $N_i$. Specifically, $C_i$ is the congestion loss counter related to the *i-th* monitoring window, and it is incremented each time a loss occurrence is inferred as associated to a congestion event. $N_i$ is the non-congestion loss counter, and it is incremented each time a packet loss is not inferred as associated to a congestion event (Fig. 1).

$N_i$ increase can be induced either by a cross-layer signaling within the receiver stack or by a transport layer event.

In the first case (i.e., cross layer signaling), when a frame is discarded at the receiving MAC layer, such event is notified to the receiving TCP and, through the TCP option field, this information is piggybacked to the sender; consequently, the sender increases its $N_i$ counter by one.

In fact, IEEE 802.11 medium access control (MAC) layer [12] provides a checksum to prevent forwarding of erroneous frames, and a frame is discarded by the MAC layer according to one of the following reasons: the received signal strength indication (RSSI) is below a given threshold, frame check sequence (FCS) detects corruption, a frame (i.e., MAC) collision occurs. All such events denote that the loss event is due to channel problems and not to network congestion.

Each transmitted MAC frame in wireless LANs should be promptly acknowledged, after a short interframe space (SIFS) time interval, if it has been correctly delivered to the receiving station (only unicast frames are acknowledged). Nonetheless, random errors and collisions due to the channel contention mechanism can corrupt the transmission of the original frame or of its returning ACK, and consequently, the ACK frame might not be received; so, the sender has to enter a new contention phase of the wireless channel for gaining the opportunity to

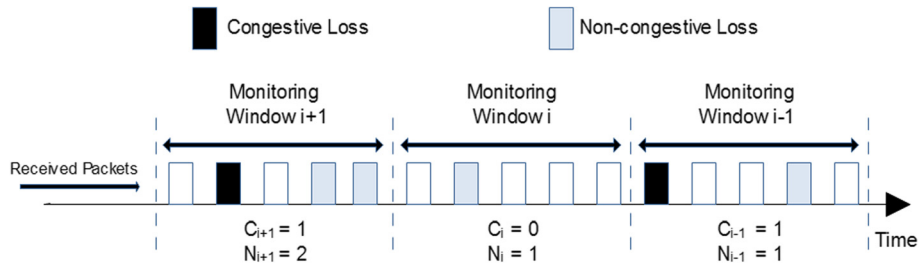


**Fig. 1** Congestion loss and non-congestion loss counters

retransmit the unacknowledged packet, until a maximum number of attempts (i.e., retry limit parameter) is achieved.

Cross-layer signaling between MAC and TCP layers can be accomplished by interacting with the management information base (MIB) standard of IEEE 802.11 [12]. In particular, the indications about channel failures can be retrieved by looking inside the "dot11QosCounterTable" defined in this standard, as well as the number of lost packets in the downlink direction. As uplink is concerned, lost packets can be estimated from the number of received MAC PDUs inside "dot11QosCountersTable" and the mean data rate field of the traffic specification (TSPEC) element. All such indications can be retrieved (e.g., through SNMP queries) at layer two and used at the TCP layer for increasing the non-congestion loss counter $N_i$.

In the second case (i.e., transport layer event), when TDACKs or RTO occur, the TCP algorithm tries to infer whether such loss event has been caused by a congestion or by a channel problem. In detail, as in JTCP [6] and [13], the congestion state is deduced through a loss ratio predictor called jitter ratio (Jr), defined as:

$$J_r = \frac{D(i,j)}{R_j - R_i} = 1 - \frac{S_j - S_i}{R_j - R_i} \quad (1)$$

where $S_i$, $R_i$ and $S_j$, $R_j$ are defined as the sending and receiving times of segments $i$ and $j$, respectively, and $D(i,j)$ represents the inter-arrival jitter for the pair of packets $(i, j)$ defined as the difference between the inter-packet times at the receiver and the inter-packet times at the sender [14].

$D(i,j)$ can be computed as

$$D(i,j) = (R_j - R_i) - (S_j - S_i) = (R_j - S_j) - (R_i - S_i) \quad (2)$$

For each packet loss, XJTCP updates the inter-arrival jitter through Eq. (2) and calculates the $Jr$ value through (1). If the jitter ratio is less than or equal to a given threshold Th, the segment loss is not considered congestion-driven; otherwise, it is due to congestion problems. Choosing the right threshold is very important because it determines how TCP should react to packet losses; as suggested in [6] for JTCP, the threshold is related to the value, in segments, of the congestion window size (cwnd) when the loss occurs and can be given by

$$\mathrm{Th} = {}^{1}/_{\mathrm{cwnd}} \quad (3)$$

From the above considerations, if a loss event has been provoked by channel problems (i.e., non-congestion, Jr ≤ Th), $N_i$ counter is increased. Otherwise, if Jr > Th, segment loss is inferred as due to congestion and the $C_i$ counter is incremented.

Finally, in order to obtain a smoothed value of $C_i$ and $N_i$, a low-pass filter is introduced at each monitoring window, by adopting the exponential weighted moving average (EWMA) [15], as shown in the following Eqs. (4) and (5):

$$\mathrm{Cewma}_i = \alpha C_i + (1-\alpha) C_{i-1} \quad (4)$$

$$\mathrm{Newma}_i = \alpha N_i + (1-\alpha) N_{i-1} \quad (5)$$

where $\alpha$ is a smoothing factor, which denotes how recent and old samples of $C_i$ and $N_i$ influence the EWMA estimates. As an example, $\alpha = 0{,}8$ means that 80 % of each new estimate comes from the last sampled value and only 20 % comes from the past sample (i.e., recent samples are given more importance).

### 3.2 Regulation of the congestion window size

A non-congestive loss can be detected at the MAC level or at the transport level. In the first case, the loss derives from a channel problem and this event is notified by the MAC layer through a cross-layer signaling to the receiving TCP; such information can be used to increase $N_i$.

In the second case, the JTCP scheme infers whether the loss is due to congestion or not, and, consequently, the TCP sender decides which counter (i.e. $C_i$ or $N_i$) should be increased.

In order to better adjust the congestion window size and hence the aggressiveness of TCP reaction, the following parameter $\theta$ has been introduced as follows:

$$\theta = \frac{Newma_i + 1}{Cewma_i + 1} \quad (6)$$

$\theta$ represents the ratio between non-congestive loss rate and congestive loss rate. In (6), the unitary value is added in order to avoid the presence of poles in the function.

The parameter $\theta$ helps to understand whether corruption or congestion is the predominant condition in a certain time window. Consequently, the TCP congestion parameters (e.g. Cwnd) can be suitably adjusted, in order to regulate the aggressiveness of TCP scheme, thus increasing its throughput and reducing packet losses (e.g., if congestion is predominant, it might be better to trigger slow start phase rather than injecting further packets in a congested environment).

So, $\theta > 1$ indicates that non-congestive losses are predominant in the wireless channel; thus, a loss event is expected to occur mainly due to a loss prone channel. On the other hand, if $\theta < 1$, congestive losses are predominant in the monitored window, and the TCP congestion control action is needed. Finally, if $\theta = 1$, no clear indications can be obtained, and the reason of a loss event is inferred through the adoption of Jr, as for native JTCP algorithm.

When three duplicate ACKs are received, if $\theta > 1$, the event is considered caused by a lossy link and immediate recovery is performed, without any modification of congestion window and slow start threshold values. Immediate recovery is also performed if $\theta = 1$ and Jr ≤ Th.

Otherwise, if $\theta < 1$, TDACKs are considered caused by a congestion, and fast recovery action is performed, by halving the congestion window and setting the slow start threshold to this new value. Fast recovery is also performed if $\theta = 1$ and Jr > Th.

After a RTO, if $\theta > 1$ or $\theta = 1$ and Jr ≤ Th, fast recovery is performed, since loss is inferred due to non-congestive event.

On the other hand, if $\theta < 1$ or $\theta = 1$ and Jr > Th, the occurred loss is inferred as a congestive event, thus triggering a slow start phase. Figure 2 shows the pseudo-code of the described algorithm after three duplicate ACKs and retransmission timer expiration events.

```
IF (TDACKs are received)
        IF (θ>1)                        //random loss
                //immediate recovery
                ssthreshold = cwnd
                cwnd = ssthreshold

        IF (θ==1)                       //undefined, JTCP actions
                IF (Jr<=1/cwnd)     //random loss
                        //immediate recovery
                        ssthreshold = cwnd
                        cwnd = ssthreshold

                IF (Jr>1/cwnd)      //congestive loss
                        //fast recovery
                        ssthreshold = ½ cwnd
                        cwnd = ssthreshold

        IF (θ<1)                        //congestive loss
                //fast recovery
                ssthreshold = ½ cwnd
                cwnd = ssthreshold

IF (RTO expires)
        IF (θ>1)                //random loss
                //fast recovery
                ssthreshold = ½ cwnd
                cwnd = ssthreshold

        IF (θ==1)                       //undefined, JTCP actions
                IF (Jr<=1/cwnd)     //random loss
                        //fast recovery
                        ssthreshold = ½ cwnd
                        cwnd = ssthreshold

                IF (Jr>1/cwnd)      //congestive loss
                        //slow start
                        ssthreshold = ½ cwnd
                        cwnd = 1

        IF (θ<1)                //congestive loss
                        //slow start
                        ssthreshold = ½ cwnd
                        cwnd = 1
```

**Fig. 2** Pseudo-code of XJTCP after TDACKs and RTO expiration

## 4 Simulation scenario

To investigate XJTCP performance over wireless environments, several performance metrics (goodput, fairness, delay, jitter, and packet losses) have been examined through extensive simulations which have been carried out through the popular NS-2 tool [16] (specifically, version 2.33).

The simulation scenario, depicted in Fig. 3, consists of an IEEE 802.11b network in infrastructure mode, composed by five wireless stations (STAs) and an access point (AP); the AP communicates with a fixed host (i.e., the server) through two routers with wired links at 100 Mb/s.

We assume that all STAs are positioned close to the AP (5–10 m distance), so as to constantly work at the maximum physical bit rate of 11 Mbit/s, and they are equipped with omni-directional antennas. The communication radius of the wireless area is about 250 m and the two-ray ground reflection model implemented in NS-2 is adopted. This radio propagation model calculates the path loss when the received signal is composed by a direct line of sight path and a multi-path component which is mainly constituted by a single ground-reflected wave. It is shown [17] that this model gives more accurate prediction at a long distance than the free space model.

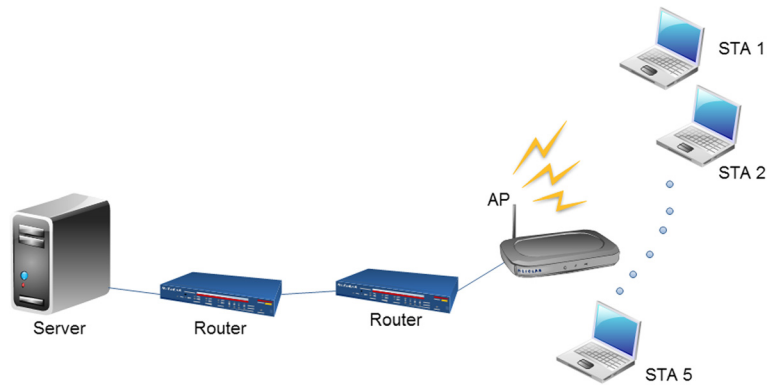


**Fig. 3** Simulation scenario

All devices have the same buffer capacity of 1000 segments, and the buffers are managed in a drop-tail policy. Each STA represents a single traffic source, and it can generate TCP or UDP packets, according to the following characteristics:

- The TCP traffic reproduces an uplink File Transfer Protocol (FTP), with a fixed packet size of 1024 byte (40 bytes header, 984 bytes payload) and an inter-packet generation time of 5 ms. Therefore, FTP can reach a maximum throughput of about 1.6 Mbit/s.
- The UDP traffic reproduces an uplink high quality video streaming, modeled as an on/off exponential traffic at variable bit rate (VBR). Traffic is generated according to an exponential distribution with an average rate of 1 Mb/s, burst time ("on" period) 1.2 s, idle time ("off" period) 1.8 s, and fixed packet size 512 bytes (8 bytes header, 504 bytes payload). During the on period, packets are generated at a constant rate of 2.5 Mbit/s, and during the off period, no packet is generated.

We have adopted TCP/FTP and UDP/VBR exponential traffic models as implemented in NS-2 [16]. Three traffic configurations have been tested, according to the desired level of congestion:

- No congestion: only one TCP traffic source (one STA) is active for the whole simulation time
- UDP congestion: one TCP traffic competes with four UDP streaming video connections, for the entire simulation time (five STAs are involved)
- TCP congestion: one TCP traffic under study (which can be Reno, JTCP, Venoplus, or XJTCP) competes with other four TCP Reno traffics (five STAs are involved), for the entire simulation. The choice of Reno version as a reference TCP is due to its popularity in almost all implementations

The above configurations can be affected by different losses in the wireless link:

- Random losses, obtained by applying a uniform error model to the wireless channel
- Bursty losses (i.e., correlated losses), generated according to an exponential probability distribution: errors occur at time intervals which follow an exponential distribution function with a specific average error rate

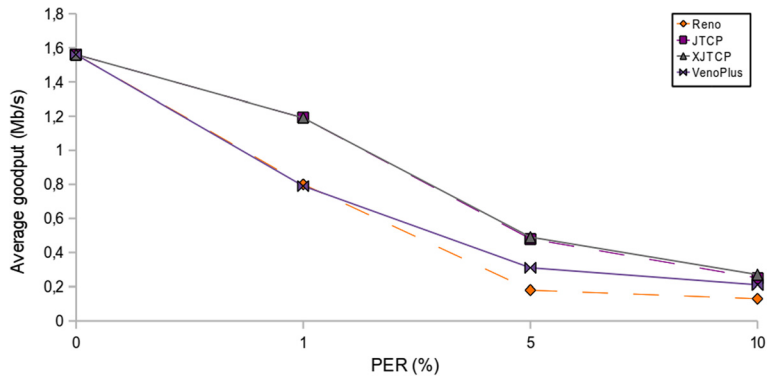


**Fig. 4** Average goodput—random loss, no congestion

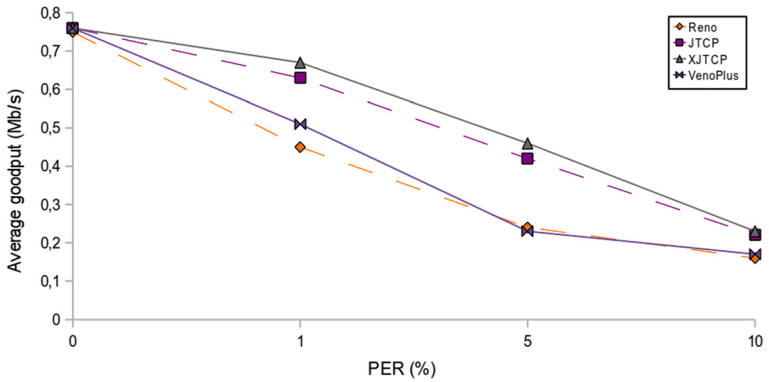


**Fig. 5** Average goodput - random loss, UDP congestion

The packet error rate (PER) is different in all simulation runs (PER is set to 0, 1, 5, or 10 %).

The one-way time delay between the sender (wireless station) and the receiver (fixed host) is set to about 50 ms, and the duration of the simulations is set to 300 s.

Different TCP variants, namely TCP Reno, JTCP, VenoPlus, and XJTCP, have been compared under the described configuration sets. In order to study TCP behavior under different traffic loads, the average TCP data transfer rate, named as *average goodput*, has been adopted as a reference parameter; its value has been obtained through repeated simulation runs (at least 10 independent runs were made for each configuration).

## 5 Performance evaluation

The first series of simulations were performed in a wireless channel affected by random losses; hence, a uniform error model has been implemented.

In the first traffic configuration (no congestion), only one TCP traffic is active during the whole simulation. Figure 4 reports average goodput values obtained under different PER levels for each different TCP algorithm.

Results show that TCP versions based on jitter-ratio (XJTCP and JTCP) perform significantly better than Reno and Venoplus. This result can be justified mainly by the ability of distinguishing the cause of packet losses between congestion and corruption. In particular, with zero value of PER (ideal channel condition), the several versions of TCP exhibit equal performance, since the new mechanisms introduced by JCTP, XJTCP, and Venoplus do not enter into play. This starting point is common to all simulation configurations (i.e., no congestion and TCP/UDP congested network), and this is always true when PER = 0 %.

When PER grows, the performance difference is more evident, especially for the two jitter-based versions, since they are able to recognize that congestion is not present (due to a constantly low jitter) and thus their congestion window keeps higher than other TCPs. In fact, looking at the XJTCP pseudo-code of Fig. 2, no congestion condition always falls in the case $\theta > 1$ and $Jr \leq Th$; therefore, XJTCP has the same reaction of JTCP (to random loss).

When congestion is introduced (UDP congestion) through the activation of four UDP flows (Fig. 5), XJTCP achieves the maximum goodput under all PER levels,

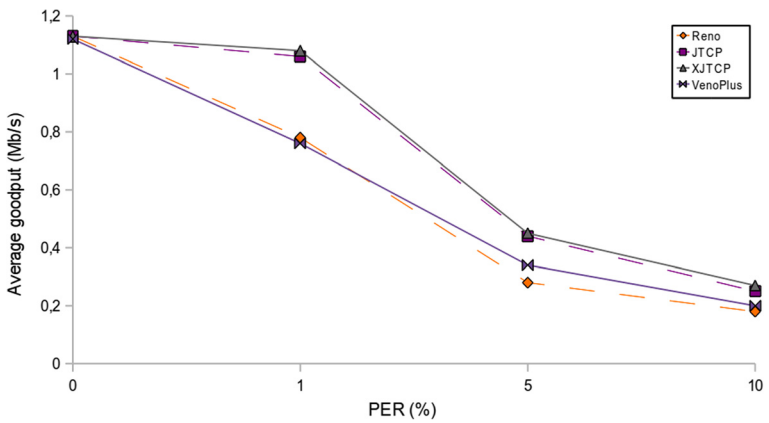


**Fig. 6** Average goodput—random loss, TCP congestion

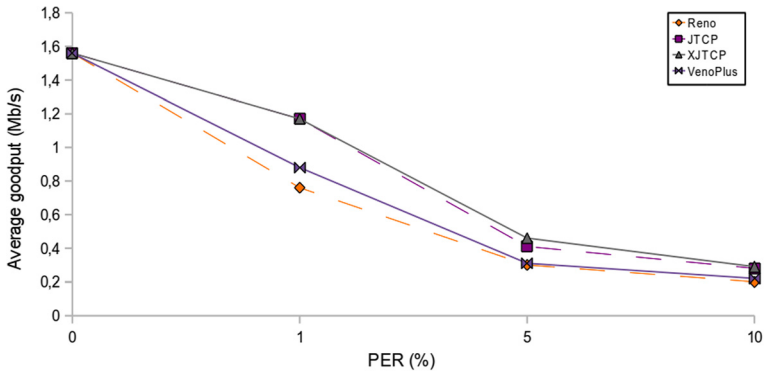


**Fig. 7** Average goodput—bursty loss, no congestion

and this is due to its capacity of reacting differently in case of congestion and random losses. The difference between JTCP and XJTCP is now due to the introduction of the $\theta$ parameter, which is now conditioned by both random losses and congested environment, thus ranging lower and above the unitary value. Consequently, XJTCP reaction can be different from JTCP and this affects its goodput positively.

Under TCP congestion (Fig. 6), the above considerations are still valid and XJTCP performs slightly better (about 5 %) than JTCP.

When burst losses are introduced in the wireless channel, under no critical conditions (no congestion) XJTCP (Fig. 7) performs similarly to the case of random losses, and jitter-based TCP versions perform better than Reno and Venoplus.

In the second configuration (UDP congestion), when the ability to distinguish between non-congestive and congestive losses becomes more important, XJTCP shows a substantial improvement with respect to its closest competitor (JTCP). In particular, its goodput performance (Fig. 8) performs better with non-zero PER values, up to 10 % more than JTCP. Even in this case, similarly to previous figures, jitter-based versions behave better in all configurations, but XJTCP, thanks to the $\theta$ indicator, achieves the best performance. In fact, in case of consecutive losses due to a burst channel, other TCP schemes react mainly to congestion (due to the fact that they can rarely receive TDACKs and their RTO expires), while XJTCP can still recognize a corrupted network and can enter into fast recovery mode (see Fig. 2, RTO expiration and $\theta > 1$).

Concerning the TCP congested scenario, XJTCP achieves again the best performance (Fig. 9), with a goodput increase of about 10 %, as before.

Even if the described simulations provided good results in terms of goodput, it is fundamental for new versions of TCP algorithm not to behave unfairly when sharing connections with other TCP implementations. Therefore, fairness investigation related to XJTCP has been carried out in this work, by adopting the *Jain's fairness* Index [18] as a benchmark.

For this test, a TCP congested scenario has been adopted, where a XJTCP-based traffic competes with four TCP Reno traffics. Figure 10 shows the Jain's fairness index trend for the XJTCP connection, under uniform and bursty losses at different PER levels.

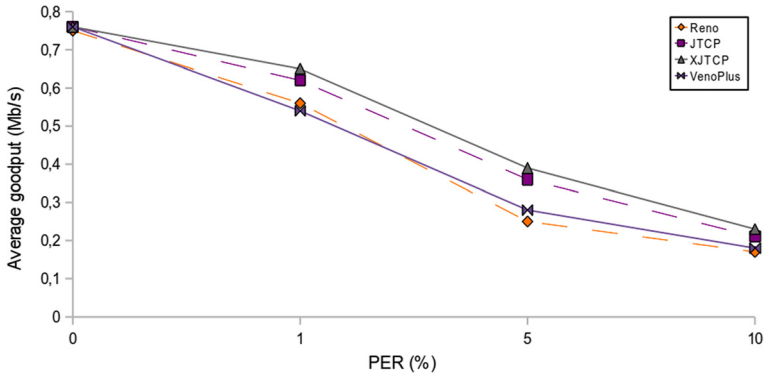


**Fig. 8** Average goodput—bursty loss, UDP congestion

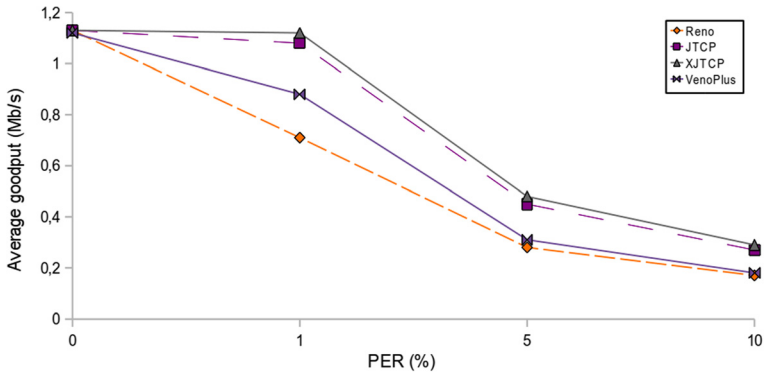


**Fig. 9** Average goodput—bursty loss, TCP congestion

Results show that XJTCP can coexist with traditional TCP implementations, showing a fair behavior that is more evident under random loss events.

However, in some cases, XJTCP can take advantage of the conservative approach of traditional TCP versions like Reno in case of non-congestive events. In fact, TCP Reno decision about the congestion level of the network is based only on TDACKs or timeout events. So, when non-congestive events increase (i.e., higher channel error rates), Reno algorithm can lead to an excessive reaction which causes repeated reductions of the Cwnd, with consequent release of network resources due to multiple slow start phases. The concurrent XJTCP can take advantage of this fact, since it is designed to better interpret loss events, consequently it tends to exploit the released resources.

This is the reason why XJTCP fairness decreases at higher PER. Moreover, this phenomenon is more evident in case of burst errors, when consecutive packet losses hinder receiving TDACKs and cause RTO expiration. Hence, this condition is misinterpreted by TCP Reno as a congestion problem, with unnecessary consecutive reductions of the Cwnd value and to drastic reduction of its packet injection rate. This fact can favor XJTCP, which can better distinguish the reason of loss event. However, this phenomenon is still acceptable, and fairness keeps always above 94 % even under high PER values.

Finally, further simulations have been carried out in order to compare the performance of the two jitter-based TCP algorithms (i.e., JTCP and XJTCP); lost packet, jitter, and delay metrics have been taken into account for this comparison and results are shown in the Figs. 11, 12 and 13. The scenario of interest reproduces the worst network conditions, characterized by a bursty channel and UDP congestion configuration.

Figure 11 shows the results related to lost packets for JTCP and XJTCP schemes. PER is considered at the link layer of 802.11. According to this standard, several corrupted frames can be recovered by link layer retransmissions; hence, TCP layer sees a more reliable channel, with a lower error rate.

JTCP exhibits a worse behavior (with a higher number of lost packets) than XJTCP. In fact, when PER augments, JTCP is less capable than XJTCP to distinguish between congestion and corruption problems which caused the loss event. Therefore, TCP connections undergo many consecutive timeouts under a high jitter

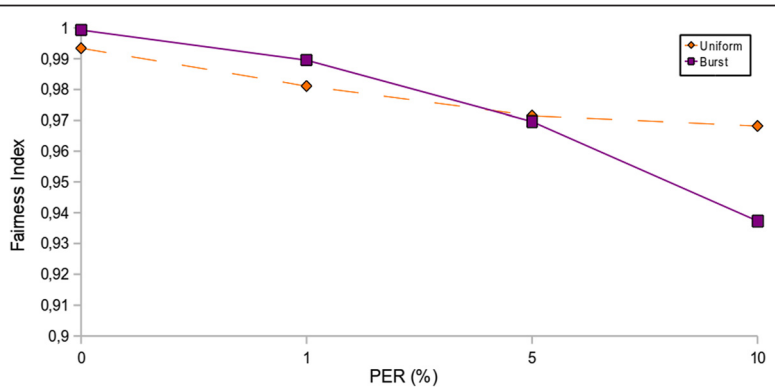


**Fig. 10** Fairness

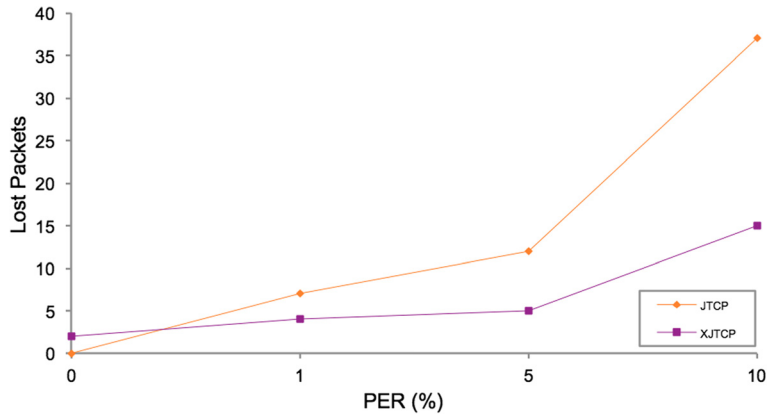


**Fig. 11** Lost packets—bursty loss, UDP congestion

value (due to congestion). This fact is interpreted by JTCP as pure congestion, and hence, it causes repetitive falls to the slow start phase. For this reason, the amount of packets to be retransmitted increases.

On the contrary, XJTCP, thanks to the $\theta$ parameter, can better distinguish between congestion and corruption and can react more suitably with fast recovery, thus experimenting an inferior number of slow start phases.

This fact impacts also on the average jitter depicted in Fig. 12, since the higher is the number of slow start phases, the more are the oscillations on the degree of congestion in the network and consequently the jitter values. Moreover, higher jitter values together with RTO expiration are interpreted by JTCP as a sign of pure congestion, triggering a reaction to slow start phase; this does not happen for XJTCP.

According to the above considerations, also average delay performance is affected. The average delay, shown in Fig. 13, has been calculated as the sum of the delays needed for each single packet to be correctly received, divided by the total number of received packets.

The average delay increases when the number of retransmitted packets grows and this happens more evidently in case of consecutive timeouts followed by slow start phases, which is the case of JTCP. Differently, XJTCP can take a more accurate decision about the predominant condition in the network, and hence, it avoids to enter unnecessary slow start phases, with positive effects on its average delay.

Delay and jitter values can fluctuate due to reactions carried out at level 2 (i.e., retransmission by 802.11 MAC) or level 4 (i.e., TCP retransmissions and congestion control). If a packet is retransmitted more times, its delay increases, and consequently the jitter fluctuates.

## 6 Conclusions

This work introduces a new TCP variant named XJTCP based on a cross-layer approach, and it investigates its performance in presence of wireless networks, compared with consolidated TCP solutions such as TCP Reno, JTCP, and VenoPlus. Several tests have been carried out, focusing on goodput and fairness parameters under random and burst losses and

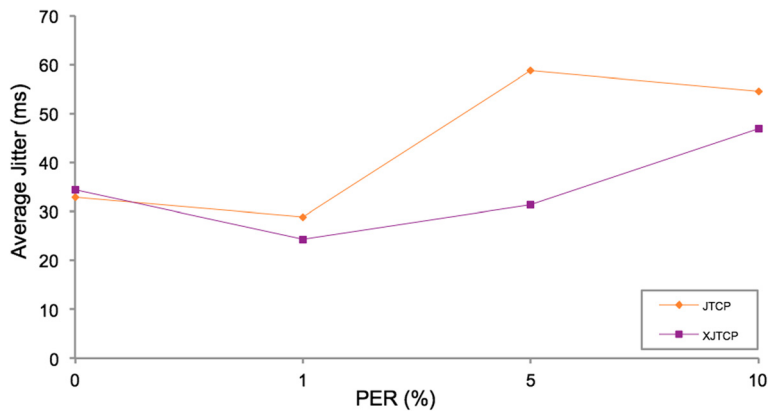


**Fig. 12** Average jitter—bursty loss, UDP congestion

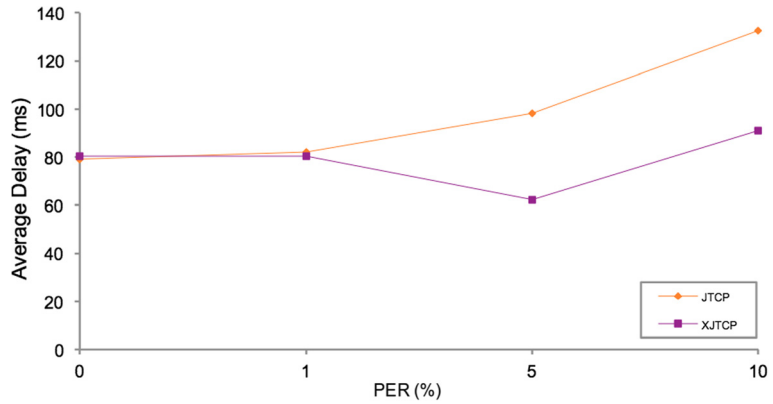


**Fig. 13** Average delay—bursty loss, UDP congestion

different congestion configurations. Performance comparison has also been carried out between JTCP and XJTCP in terms of delay, jitter, and packet losses. Results show that XJTCP exhibits a good performance in all configuration sets, thus representing an interesting solution when wireless scenarios are present in the end-to-end path. Even if the adopted cross-layer approach introduces new complexity in TCP algorithm, the computational power which characterizes today devices, included the mobile world, makes the implementation of XJTCP feasible without undermining resources significantly.

Further studies can investigate XJTCP behavior under different network technologies such as 3GPP-LTE and with user mobility.

**Acknowledgements**
The authors wish to thank the anonymous reviewers for their detailed and constructive comments, which helped so much to improve the quality of the paper.

**Competing interests**
The authors declare that they have no competing interests.

Received: 1 January 2016 Accepted: 11 August 2016
Published online: 20 August 2016

**References**

1. M Allman, VE Paxson, W Stevens, TCP congestion control request for comments, RFC 2581, Network Working Group, Internet Engineering Task Force (1999)
2. KC Leoung, VOK Li, Transmission control protocol (TCP) in wireless networks: issues, approaches, and challenges. IEEE Commun. Surv. Tutorials. **8**(4), 64–79 (2006)
3. B Sardar, D Saha, A survey of TCP enhancements for last-hop wireless networks. IEEE Commun. Surv. Tutorials. **8**(3), 20–34 (2006)
4. A Mammadov, B Abbasov, A Review of protocols related to enhancement of TCP performance in wireless and WLAN networks, IEEE 8th International Conference on Application of Information and Communication Technologies (AICT), pp.1-4, 15–17 (2014)
5. D Ros, M Welzl, Less-than-best-effort service: a survey of end-to-end approaches. IEEE Commun. Surv. Tutorials. **15**(2), 898–908 (2013)
6. EHK Wu, MZ Chen, JTCP: jitter-based TCP for heterogeneous wireless networks. IEEE J. Sel. Areas Commun. **22**(4), 757–766 (2004)
7. S Shetty, Y Tang, W Collani, TCP venoplus—a cross-layer approach to improve TCP performance in wired-cum-wireless networks using signal strength, IEEE International Conference on Networking, Sensing and Control (ICNSC), Chicago (US), pp.693-697 (2010)
8. C Liu, R Jain, Approaches of wireless TCP enhancement and a new proposal based on congestion coherence, IEEE 36th Annual Hawaii International Conference on System Sciences (HICSS) (2003)
9. H Singh, S Singh, Energy consumption of TCP Reno, Newreno, and SACK in multi-hop wireless networks, ACM International Conference on Measurement and modeling of computer systems (SIGMETRICS), vol. 30, no. 1, pp.206-216 (2002)
10. A Afanasyev, N Tilley, P Reiher, L Kleinrock, Host-to-host congestion control for TCP. IEEE Commun. Surv. Tutorials. **12**(3), 304–342 (2010)
11. CP Fu, SC Liew, TCP Veno, TCP enhancement for transmission over wireless access networks. IEEE Journal of Selected Areas in Communications (JSAC) **21**(2), 216–228 (2003)
12. IEEE Std. 802.11e-2005, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 8: Medium Access Control (MAC) Quality of Service Enhancements, IEEE Std. 802.11e (2005)
13. A Andreadis, S Rizzuto, R Zambon, Performance comparison of JTCP over QoS-aware WiMAX networks, IEEE Symposium on Computers and Communications (ISCC), Corfu, Greece, pp.371-376 (2011)
14. H Schulzrinne, S Casner, R Frederick, V Jacobson, RTP: a transport protocol for real-time application, RFC 1889, Internet Engineering Task Force (IETF) (1996)
15. W Richard Stevens, TCP/IP illustrated, Addison Wesley (1994)
16. Network Simulator ns-2. http://www.isi.edu/nsnam/ns. Accessed 16 Aug 2016
17. T S Rappaport, Wireless communications, principles and practice. Prentice Hall (1996)
18. R Jain, D M Chiu and W Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared systems, DEC Research Report TR-301 (1984)