Improved Cross-Layer Congestion Control for TCP over Wireless Multi-hop Networks

Dzmitry Kliazovich, *Student Member, IEEE*, and Fabrizio Granelli, *Member, IEEE*

***Abstract*—The paper presents the problem of performance degradation of transport layer protocols due to congestion of wireless local area networks. Following the analysis of available solutions to this problem, a cross-layer congestion avoidance scheme, called C^{3}TCP, is presented, able to obtain higher performance by gathering capacity information such as bandwidth and delay at the link layer. The method requires the introduction of an additional module within the protocol stack of the mobile node, able to adjust the outgoing data stream based on capacity measurements. Achieved results underline good agreement with design considerations and high utilization of the available resources.**

I. Introduction

Recent developments in the framework of wireless networks are aimed at providing data delivery over multiple wireless links. The paper targets the problem of network congestion in multi-hop wireless networks as the main reason for potential performance degradation, while aspects related to the nature of wireless links, such as limited bandwidth, increased latency, channel losses, mobility, etc., which introduce performance degradation, are neglected.

Congestion occurs when the amount of data sent on the network exceeds the available capacity. Such situation leads to increased buffer space usage in intermediate nodes over the data path, with data losses in case of shortage of resources. The transmitted data starts to be dropped when available buffer resources, which are physically limited, are exhausted.

Such situation decreases network reliability in the sense of service provisioning for data communication. Reliable transport protocols improve reliability by implementation of different error recovery schemes. However, they could lead to excessive data retransmissions, reducing an important parameter such as network utilization, while at the same time increasing latency in data delivery.

This paper targets the core reason for network congestion – the amount of traffic emitted to the network. For such reason, the proposed solution for congestion avoidance is to control (and possibly optimize) the amount of traffic being sent onto the network, considering limited availability of network resources. IEEE 802.11 standard [1], representing currently the leading solution for providing communications over wireless local area networks, will be considered as the reference MAC/PHY infrastructure.

The rest of the paper is organized as follows: section II presents an insight related to the nature of congestion, while section III outlines the state-of-the-art in TCP adaptation to wireless links. The proposed approach is detailed in section IV, and performance evaluation is presented in section V. Finally, section VI draws some conclusions.

II. Nature of Congestion

TCP/IP reference model defines a set of protocols that enable communication over the Internet. No layer has complete and real-time information about available network resources over the multi-hop path where the communication is performed. For that reason, the sources can avoid network congestion based on network feedback, which is obtained as a reaction to a certain amount of data being sent on the network.

The dominant transport protocol in the Internet is the Transmission Control Protocol (TCP) [2], used by a variety of applications, a reliable connection-oriented byte-stream protocol which performs congestion control dynamically during the data communication process. Window-based TCP congestion control is performed on an end-to-end basis [3]. The receiver provides an acknowledgement (ACK) feedback back to TCP sender. Relying on the information provided by ACKs, the sender can detect which packets are lost during transmission over the communication path.

Congestion window (*cwnd*) controls the amount of data the sender is allowed to output to the network without acknowledgement. The congestion window evolution is the key parameter for TCP congestion control. TCP uses additive increase and multiplicative decrease strategy for its window adjustment according to network conditions. The connection is initiated with window size equal to one packet. Then, *cwnd* is increased exponentially for every non-duplicate ACK reception until the Slow Start Threshold (*ssthresh*) is reached. Prior the connection establishment ssthresh is set to an initial value, which depends on the implementation of the protocol stack, and then adjusted on the basis of an estimate of the network capacity. This technique is called *slow start* phase.

When *ssthreshold* is reached, TCP enters congestion avoidance phase. The window is increased linearly by one packet for each ACK received. The window growth in this phase is limited to a maximum window size, negotiated between sender and receiver during connection establishment and then updated on the fly during the communication process.

There are two ways for TCP sender to detect data loss occurred on the communication link: reception of duplicate ACKs (*dupacks*) and timeout occurrence. In the first case, a *dupack* is generated by the receiver upon reception of an out-of-order packet, detected through the analysis of sequence numbers of incoming packets. Duplicate ACK reception triggers congestion window reduction to the half of its current size. In the second case, upon the timeout occurrence, TCP reduces the size of cwnd to its initial value (equal to 1) entering slow start phase.

TCP was originally designed for wired networks where packet losses occur mostly due to congestion. For that reason,

congestion avoidance/recovery is the only reaction of TCP to losses [4].

Moreover, TCP performance directly depends on the window size, which optimal value should be proportional to bandwidth-delay product of the entire path of the data flow [5]. The excess of this threshold does not bring any additional performance enhancement, but only leads to increased buffer requirements in intermediate nodes along the data connection.

In this paper, a wireless multi-hop IEEE 802.11-based network is considered, even if most results have a more generic application. The rest of the paper assumes that the reader is familiar with general aspects of IEEE 802.11 standard [1] as well as multi-hop routing for wireless LANs [6], which is not covered by the standard.

The authors of [7] produced an evaluation of TCP performance in wireless multi-hop network, underlining two major problems: instability and unfairness. The observed instability in the performance is present even in the simplest scenario with only one data flow over a multi-hop connection. The main reason for that is touching TCP timeouts, which forces TCP to follow the slow start period greatly degrading the performance through congestion window reduction. The second phenomenon observed in [7] is unfairness, which happens between two TCP data flows that occupy the same number of hops on the same path: the flow started later in time will gain full bandwidth advantages, degrading the performance of previous flow down to zero.

A new metric, called expected throughput, is introduced in [8] and used as a bound for comparison of the achieved performance for existing TCP implementations in wireless multi-hop scenario. The simulations show that the TCP performance, being far from the desired level, is unacceptable by several applications. Similar results are presented in [9].

III. Available Solutions

During the past years, a relatively strong effort of the research community was devoted to TCP adaptation to the wireless multi-hop network scenario. The majority of the available solutions which modify congestion control algorithm of TCP can be logically classified into the three following categories:

TCP modifications: Solutions within this category attempt to conquer the roots of the problem, replacing transport protocols with versions designed considering the different characteristics of the wired and wireless scenarios. The main problem associated with poor performance of transport protocols over wireless networks, in fact, lies in the fact that they are designed for wired networks. Obviously, solutions within this category provide reasonable performance improvement if compared with traditional implementations of transport layer protocols. However, the main disadvantage – which prevents wide deployment of the proposed approaches – lies exactly in the requirement for modification of a standardized and widely deployed transport protocol. Example of approaches within this category are: TCP Vegas [10], TCP Westwood [11], Adaptive Transport Layer [12].

Explicit Feedback Solutions: solutions presented in this category implement different explicit feedback techniques. All of them rely on the available buffer space at intermediate nodes, which forces them to depend more on the size of allocated memory rather than on the available capacity of the communication links. Such a tradeoff leads to an increased response time to the congestion. In fact, solutions based on the analysis of buffer free space react to congestion later if compared with solutions which analyze the difference between capacities of the incoming and outgoing communication links. Examples of explicit feedback solutions include: Random Early Detection (RED) [13], Explicit Congestion Notification (ECN) [14], Explicit Window Adaptation (EWA) [15].

Transport Layer Capacity Measurement: The ideal case for the source node is to have capacity information of the entire data communication path for the entire duration of the connection. In order to approach to this ideal case, many proposals [16-20] target enhancement of transport layer protocols through capacity probing techniques. The entire capacity of communication path is represented by two parameters: bandwidth and delay. Capacity measurement techniques have obvious drawbacks which prevent their successful deployment:

- They do not work under certain circumstances, such as under presence of cross-traffic which is both intensive and non-reactive [20].
- Probing network bandwidth requires an insertion of additional traffic, which reduces available resources.
- Bandwidth information becomes available to sender after the time required for a roundtrip propagation of the probe sequence over the network path.
- Additional computation resources are required from the sender for statistical processing of the measured data.

IV. Cross-Layer Congestion Control over Multi-hop Wireless Networks

This section presents a novel scheme (*Crosslayer Congestion Control for TCP*: *C*^{3}*TCP*) for congestion control over multi-hop wireless networks. A string network topology is considered, as the simplest topology which approximates the multi-hop scenario [21]: only neighboring nodes can communicate with each other due to the limitations in transmission range; it is assumed that every node has an appropriate output FIFO buffer, where the link layer packets are queued while the wireless medium is busy; input queuing is omitted for simplicity of presentation. Second assumption consists in the availability of a routing path to every node of the multi-hop network: details related to route discovery are out of the scope of the paper.

Bandwidth Measurement: Basic medium access mechanism specified by IEEE 802.11 standard is the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) with binary exponential backoff: the node is not allowed to transmit until the medium becomes free (i.e. no pending transmissions of other nodes), dividing the available bandwidth among nodes which share the same medium. An optional part of the standard specifies RTS/CTS exchange, considerably reducing data losses caused by collisions.

The available bandwidth B for transmission of a certain amount of data can be obtained knowing the size of data

D and time T taken for transmission of such data over a specific link: $B = D/T$.

The detailed framework for single data packet transmission is presented in Fig. 1. For more details on IEEE 802.11 MAC, the reader should refer to [1].

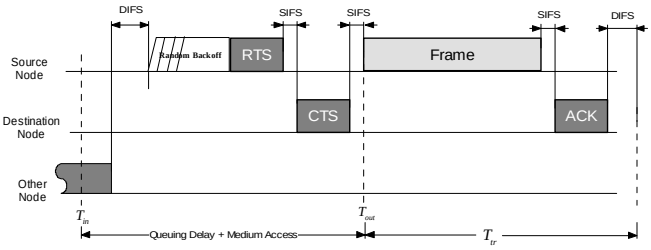

Figure 1. IEEE 802.11 Basic MAC and Data Delivery.

Taking into account the described data delivery framework, the time required for the transmission of a single data packet using CSMA/CA can be obtained as follows:

$$T = T_{out} - T_{in} + T_{tr}$$

where the difference $T_{out} - T_{in}$ includes data queuing delay corresponding to the time the node waits for all other nodes to finalize their pending transmissions as well as channel-to-channel access delay using random backoff and optional RTS/CTS exchange. A similar way of analysis was first presented by Giuseppe Bianchi for performance analysis of IEEE 802.11 DCF [22].

In order to calculate the time T_{tr} required for data frame transmission and its corresponding acknowledgement, it is necessary to take into account that data encapsulation is performed at both physical and link layers, since Physical Layer Convergence Protocol (PLCP) preamble and header are always transmitted at the basic rate, regardless of the maximum bitrate available on the medium. The MAC header, being transmitted at the data rate specified in the PLCP header, contains information about the delivered data on the link layer. FCS field finalizes frame containing CRC information.

According to physical and link layer specifications, the time required for data packet delivery (including the data frame and the corresponding ACK) is calculated as follows:

$$T_{tr} = T_{data} + SIFS + T_{ACK} + DIFS$$

$$T_{data} = \frac{PLCP.\mathrm{Pr}eamble + PLCP.Header}{Basic.Rate} + \frac{MAC.Header + FCS}{Data.Rate} + \frac{Data}{Data.Rate}$$

$$T_{ACK} = \frac{PLCP.\mathrm{Pr}eamble + PLCP.Header}{Basic.Rate} + \frac{ACK.Header + FCS}{Data.Rate}$$

The time required for a data frame transmission T_{data} includes the term corresponding to physical preamble and header (always fixed size and transmitted at basic rate), which can be calculated once and reused for subsequent calculations.

The second term corresponds to the time required for MAC header and CRC information. However, since most of the rates specified by the standard are obtained from the maximum one through simple operations, it is possible to pre-calculate them by building a short table considering the entire set of possible data rates for a given physical layer extension of the standard.

The algorithm for bandwidth measurement is the following:

1. Store the timestamp T_{in} for every frame arrived at the link layer for further transmission (from other nodes for forwarding, or locally generated by upper layers).
2. Prior actual data frame transmission, at time T_{out} calculate the time taken for packet delivery including queuing and packet transmission time T_{tr} (see eqs. above).
3. Calculate the bandwidth experienced by the packet using $B = D/T$.

The bandwidth calculated by the presented algorithm includes the following components: queuing delay, time required to gain medium access and delay associated with physical and link layer header transmission. This means that the entire overhead occurring before actual data transmission over the physical medium is considered a factor reducing the available bandwidth on the link.

Delay Estimation: Transport layer protocols provide end-to-end data delivery without having any knowledge on the network structure, like number of hops, parameters of communication link and so on, assuming to have a single data pipe between end nodes. In order to reach the maximum performance, transport protocols ideally should fill the data pipe with its bandwidth-delay product.

Considering a four-node example (see fig.3), and assuming that node N_1 wishes to communicate with node N_4, data generated by the transport layer of node N_1 is placed in the output queue on the link layer. Then, after node N_1 gains medium access, the data packet can be transmitted to node N_2. Node N_2, upon the reception of the data packet to be forwarded to the next node of the string topology, performs its medium access procedure – during which the packet can be required to wait in the output queue. Similar procedures are performed by node N_3, before the data reach the destination node N_4. Queuing delay is present on all the nodes except the destination node: the length of this pipe consists only of the time required for actual data transmission at the physical layer through all the links along the communication path.

Most solutions for optimization of the TCP performance through congestion window adjustment (presented in Section 3) rely on RTT as a delay measurement parameter. Such a way of delay measurement approximates forward and backward links with a single data pipe. However, TCP assigns different purposes to links in forward and backward directions: forward direction is used for transfer of application payload, while the backward direction for the acknowledgement procedure.

In the proposed delay estimation technique we differentiate between forward and backward delays. Forward delay contains the length of the data pipe between sender and receiver nodes, while backward delay measures the time required for the delivery of TCP ACK packets.

1) Forward delay. Single-hop forward delay experienced by a data frame includes channel access delay, time required for data delivery on the physical layer (including physical and link layer overhead), but excluding link layer queuing delay. Forward delay is calculated using the previous formulas, setting T_{in} to be equal to the time the packet leaves the queue preparing for actual transmission on the link layer. Such estimation avoids the insertion of link layer queuing delay into the $T_{out} - T_{in}$ component.

Most of the transport layer measurement techniques include also queuing delay along the entire path experienced by the data packet at the link and IP layers, so that an additional cross-traffic increases the bandwidth-delay product through an increased measured delay. In such situation, bandwidth-delay product should be decreased through reduction of the available bandwidth component while leaving forward delay unaffected.

2) Backward delay. TCP reliability is achieved through implementation of a positive acknowledgement scheme. The receiver acknowledges successful data delivery with TCP ACK packets going back towards the sender. On contrary with forward delay measurement technique described above, TCP ACK delay does include both transmission and queuing delays, i.e. it is equal to the difference between the time the TCP ACK packet was generated by the receiver node and its reception by the TCP sender.

The proposed method for estimation of the delay in forward and backward directions allows the TCP sender to adjust the amount of outstanding data to the bandwidth-delay product in the forward path, while considering the backward path as a simple delay line for TCP acknowledgement reception.

An evaluation of the capacity experienced by a certain data packet can be obtained by the analysis of capacity of the individual links: the end-to-end bandwidth over an n-hop path is equal to the bandwidth of the bottleneck on the path, while the end-to-end delay is obtained by the superposition of delays introduced by individual links.

"Options" Integration in IEEE 802.11: The obtained values for end-to-end bandwidth and delay should be forwarded to the source producing traffic to let it implement congestion control based on such measurements. In order to support such functionality, we propose to extend IEEE 802.11 MAC protocol by allowing the specification of optional fields inside the MAC header (Fig. 2).

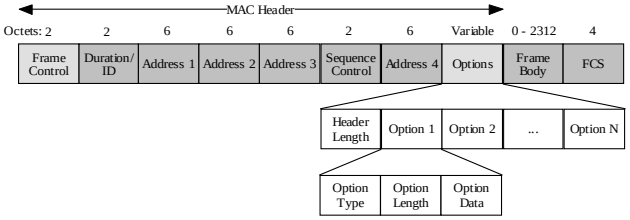

Figure 2. "Options"-enabled IEEE 802.11 data frame.

"Options" is a variable length field which extends standard MAC header. It consists of "Header Length" field which specifies the entire length of the MAC header, including the list of options. The length of the header is required to perform separation of the data encapsulated into the frame from the MAC header. Each option consists of option type, length in octets and data. The knowledge of the option's length makes skipping the current option easier, jumping to the next one for processing.

The Proposed Approach – C^{3}TCP: Fig. 3 presents an example of TCP communication over a 3-hop wireless network. Sender node N_1 initiates transmission by sending a TCP data packet to node N_4 over the string topology. Upon reception of the TCP data packet, the link layer of node N_1 performs bandwidth and delay measurements for link L_1. Then, it includes the measured information into the corresponding optional fields inside the MAC header. Node N_2, after the reception of the data frame from node N_1, performs the same measurements for link L_2, it takes the minimal value for the measured bandwidth of links L_1 and L_2 and updates the bandwidth option in the MAC header. Delay experienced on the link L_2 is added to the delay on link L_1.

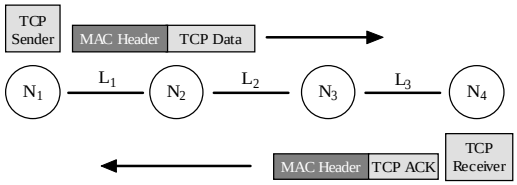

Figure 3. C^{3}TCP usage over 3-hop wireless network.

When the TCP data packet reaches the destination node N_4, its MAC header contains bandwidth and delay experienced by the packet on the path through links $L_1 - L_2 - L_3$. TCP receiver in N_4 replies with TCP ACK back to the sender indicating successful data packet reception. This TCP ACK is also encapsulated by link and physical layer headers, including the bandwidth-delay information obtained by N_4 during TCP data packet reception. Such information is simply echoed back using the appropriate optional fields.

Upon the reception of the TCP ACK packet, sender node N_1 will have the bandwidth and the delay for both transmitted TCP data and TCP ACK packets, and it can adjust the outgoing traffic using the calculated bandwidth-delay product. The bandwidth is taken only from TCP data packet propagating in forward direction, while the delay is obtained as sum of propagation delays of TCP data and ACK packets.

The goal of the presented approach is to avoid any changes at the transport layer of the protocol stack. For that reason, an additional module, called Congestion Control Module (CCM), is inserted below the transport layer (see fig. 4). This module cooperates with the link layer providing congestion control information for the transport layer, using a cross-layer collaboration technique. Implementation of cross-layer signaling is out of the scope of the paper; a detailed description of existing approaches is provided in [23].

CCM has different functionalities depending whether it is implemented at the sender or at the receiver node:

- *At the receiver's side*, upon the reception of a packet, CCM requests from the link layer the bandwidth and the delay which have been delivered with the TCP data packet. Having access to TCP headers, CCM traces the outgoing TCP ACKs. In case the produced TCP ACK acknowledges the received TCP data packet, CCM forwards the request to the link layer in order to include bandwidth-delay information into the MAC header. In the case of cumulative or selective acknowledgements, CCM will include forward path measurement obtained from the last data packet acknowledged by the outgoing TCP ACK.
- *At the sender's side*, CCM requests end-to-end measurements from the link layer upon TCP ACK packet reception. Then, it calculates the desired size of the congestion window based on the bandwidth and RTT values on the forward path.

TCP specification includes receiver advertised window function, which main idea is to allow the receiver to specify (in the TCP ACK header) the desired congestion window size, usually representing unoccupied buffer space at the receiver. CCM uses the receiver advertised window (RWND) field of TCP ACK packet for the delivery of the calculated congestion window. In detail, it leaves the lower *cwnd* value between the calculated one and the one reported by the receiver. Producing congestion control through the correction of receiver's advertised window is not a novel approach, on the contrary it is a common technique for the congestion window adjustment: for example, in [15] RWND field is used to inform the sender about network congestion from intermediate routers based on the free buffer space left on the edge device.

RWND signaling adjusts TCP window limiting its upper bound of evolution. In order gain full control on the size of the window, CCM should be enabled with acknowledgement generation for the local TCP layer in order to control the behaviour of TCP congestion control algorithm.

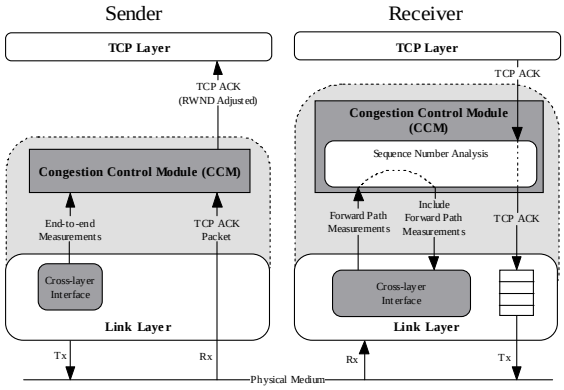

Figure 4. C[3]TCP protocol architecture.

V. Performance Evaluation

The performance of the proposed solution is analyzed using the ns-2 network simulator [24]. Simulation scenario consists of four nodes involved in a single TCP connection and two nodes which produce UDP cross-traffic (Fig. 5). Congestion control module (CCM) is attached at the link layer of end nodes of the TCP connection. At the sender size (node N_1), CCM dynamically adjusts TCP congestion window specifying its desired size by the RWND field of TCP header. Simulation parameters are set to satisfy the IEEE 802.11b specification of the standard [1]. In order to reduce collisions, RTS/CTS exchange is employed. Performance evaluation in more complex scenario of multi-hop network consisting of tens of mobile nodes is not included into current paper due to space limitations. However it shows good agreement with presented results. The results from extensive evaluation will appear in the journal version of the paper.

Figure 6 presents accuracy in terms of bandwidth and RTT estimation of C[3]TCP. In opposite to transport measurements, link layer measurements avoid including queuing delay into the measured round trip delay value. However, delay variation is still present due to the difference in medium access time as a consequence of collisions and random backoff.

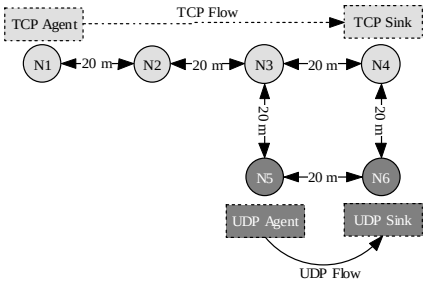

Figure 5. C[3]TCP evaluation scenario.

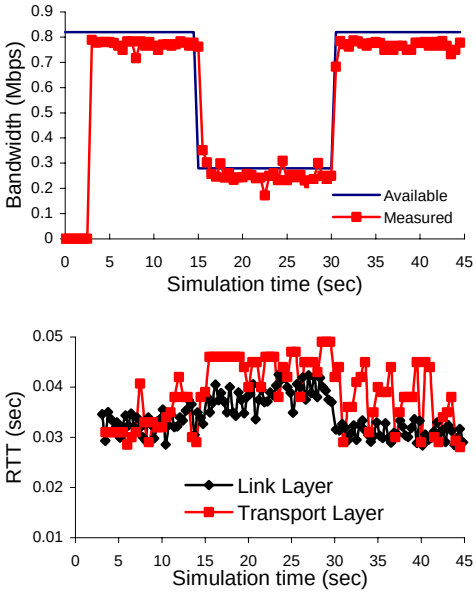

Figure 6. C[3]TCP bandwidth and delay estimation.

Moreover, a comparison in terms of end-to-end throughput between TCP with cross-layer congestion control (C[3]TCP) and standard TCP implementation [2] is presented in Fig. 7. Results underline stability of throughput in the case of C[3]TCP during all the phases of the experiment, while classical implementation of congestion control tends to enlarge the window, periodically incurring into congestion.

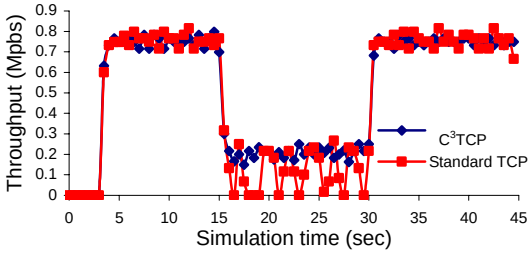

Figure 7. Throughput of C[3]TCP against standard TCP.

Buffer usage of standard TCP flow is relatively low (less than 5 packets) when the incoming and outgoing data rates at node N_2 are comparable. However, in presence of cross-traffic, standard TCP source produces much more packets, overloading the bottleneck link. Such situation leads to buffer overflow and consequently multiple packet drops in the interval between 15.0 and 30.0 sec. On the contrary, knowledge of the capacity of the communication path greatly

reduces buffer usage in the case of C^{3}TCP: buffer in node N_2 does not exceed the value of 10 packets in presence of cross-traffic and it is fixed in the interval between 0 and 3 packets when C^{3}TCP manages the only flow in the network. This is due to the fact that C^{3}TCP does not produce more packets than the network can transport, saving communication resources and avoiding multiple packet drops along the data path.

Furthermore, TCP Vegas [10] and TCP Westwood [11] are chosen among those congestion control solutions which most closely approximate C^{3}TCP from the theoretical point of view. The results show that all evaluated approaches achieve relatively close (with difference less than 2%) throughput in the scenario when TCP flow is not affected by cross traffic. However, when cross-traffic is present (between 15.0 and 30.0 seconds of simulation), both TCP Vegas and TCP Westwood periodically drop their throughput down to zero due to over-estimation of the link capacity. From the numerical point-of-view, results show an improvement achieved by C^{3}TCP of around 27%, 18% and 7% against standard TCP, TCP Westwood, and TCP Vegas, respectively.

Portions of the throughput graphs are reported in Fig. 8, that demonstrates that C^{3}TCP throughput is stable for the entire simulation interval, showing good approximation of the available link capacity. The performed comparisons underline the advantages of capacity measurement at the link level rather than at the transport level.

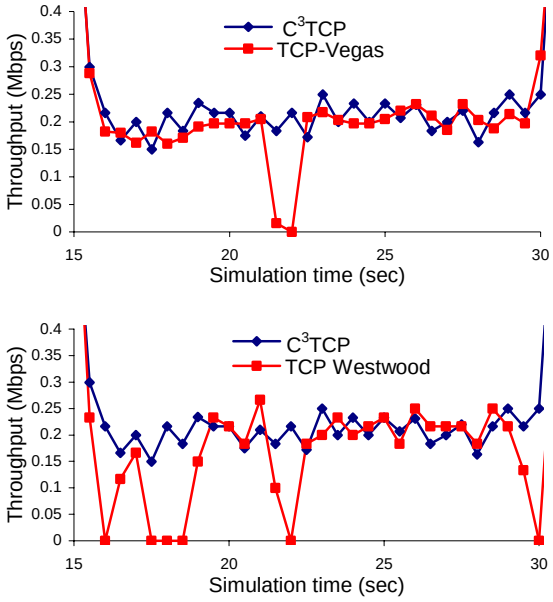

Figure 8. Throughput of C^{3}TCP against TCP-Vegas and -Westwood.

VI. Conclusion

The paper presents the problem of performance degradation of transport layer protocols in multi-hop wireless networks due to congestion. Following an analysis of available solutions to this problem, a cross-layer congestion avoidance scheme (C^{3}TCP) is presented, able to obtain higher performance by gathering capacity information such as bandwidth and delay at the link layer. The method requires the introduction of an additional module within the protocol stack of the mobile node, able to adjust the outgoing data stream based on capacity measurements. Achieved results underline good agreement with design considerations and high utilization of the available resources.

References

[1] Wireless LAN medium access control (MAC) and physical layer (PHY) Specifications, IEEE 802.11 standard, 1997.
[2] J. Postel, Transmission control protocol, RFC 793, September 1981.
[3] V. Jacobson, Congestion Avoidance and Control, Computer Communication Review, vol. 18, no. 4, pp. 314-329, Aug. 1988.
[4] W. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, RFC 2001, January 1997.
[5] R. Wang, G. Pau, K. Yamada, M.Y. Sanadidi, M. Gerla, TCP startup performance in large bandwidth delay networks, INFOCOM 2004, vol. 2, March 2004, pp. 796 – 805.
[6] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols, MobiCom'98, ACM, Dallas, TX, October 1998.
[7] S. Xu, T. Saadawi, Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?, IEEE Communications Magazine, vol. 39 , pp. 130 – 137, June 2001.
[8] G. Holland, N. Vaidya, Analysis of TCP performance over mobile ad hoc networks, Wireless Networks, vol. 8, pp. 275-288, March 2002.
[9] Z. Fu, X. Meng and S. Lu, How bad TCP can perform in mobile ad hoc networks, Seventh ISCC, July 2002.
[10] L. Brakmo, L. Peterson, TCP Vegas: End to End Congestion Avoidance on a Global Internet, Computer Communication Review , vol. 25, pp. 69-86, October 1995.
[11] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, R. Wang, TCP Westwood: End-to-End Congestion Control for Wired/Wireless Networks, Wireless Networks Journal vol. 8, pp. 467-479, 2002.
[12] O. Akan, I. Akyildiz, ATL: an adaptive transport layer suite for next-generation wireless Internet, IEEE Journal on Selected Areas in Communications, vol. 22 , pp. 802 – 817, June 2004.
[13] S. Floyd, V. Jacobson, Random Early Detection gateways for congestion control for high speed data networks, IEEE/ACM Trans. on Networking, vol. 1, no. 4, pp. 397-413, February / August 1993.
[14] K. Ramakrishnan, S. Floyd, D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, RFC, September 2001.
[15] L. Kalampoukas, A. Varma, K. Ramakrishnan, Explicit Window Adaptation: A Method to Improve TCP Performance, Infocom, 1998.
[16] V. Jacobson, R. Braden, D. Borman, TCP Extensions for High Performance, Request for Comment RFC 1323, May 1992.
[17] H. Ningning, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal on Selected Areas in Communications (JSAC), vol. 21, no. 6, pp. 879 – 894, Aug. 2003.
[18] K. Lai, M. Baker, Nettimer: A tool for measuring bottleneck link bandwidth, USITS'01, March 2001.
[19] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure? IEEE INFOCOM, April 2001.
[20] R. Kapoor, L. Chen, M. Sanadidi, M. Gerla, CapProbe: A Simple and Accurate Technique to Measure Path Capacity, Technical Report TR040001, UCLA SCD, 2004.
[21] M. Gerla, R. Bagrodia, Z. Lixia, K. Tang, W. Lan, TCP over wireless multi-hop protocols: simulation and experiments, ICC'99, June 1999.
[22] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE Journal on Selected Areas in Communications (JSAC), vol. 18, pp: 535 - 547, March 2000.
[23] Q. Wang, M.A. Abu-Rgheff, Cross-layer signalling for next-generation wireless systems, WCNC 2003, Vol. 2, pp. 1084-1089, March 2003.
[24] NS-2 simulator tool home page. http://www.isi.edu/nsnam/ns/, 2000.