Available online at www.sciencedirect.com
ScienceDirect
Ad Hoc Networks 4 (2006) 687–708
Ad Hoc Networks
www.elsevier.com/locate/adhoc

Cross-layer congestion control in ad hoc wireless networks

Dzmitry Kliazovich, Fabrizio Granelli *

Department of Information and Communication Technologies, University of Trento, Via Sommarive 14, I-38050 Trento, Italy

Received 18 February 2005; received in revised form 18 July 2005; accepted 5 August 2005
Available online 31 August 2005

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 (C^3TCP) 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. Moreover, a proposal to provide optional field support to existing IEEE 802.11 protocol, in order to support the presented congestion control solution as well as many other similar approaches, is presented. Achieved results underline good agreement with design considerations and high utilization of the available resources.

Keywords: Congestion control; TCP-over-wireless; Multi-hop wireless networks

1. Introduction

The IEEE 802.11 standard [1] represents the leading solution providing communications over wireless local area networks. A section of the standard specifies a network architecture, called *ad hoc*, which is capable to operate without the requirement for a fixed infrastructure. For such reason, in the remainder of the paper, IEEE 802.11 will be considered as the reference MAC/PHY protocol stack.

The main limitation present in the standard is the restriction of the ad hoc network to the case when all the stations are located in range of each other. However, ongoing research overcame such limitations, allowing data delivery over an end-to-end path which can consist of several wireless hops. For that reason, the paper considers the general case of data transport over an ad hoc multi-hop wireless network.

Moreover, for the purpose of the paper, the problem of network congestion is considered as

* Corresponding author. Tel.: +39 0461882062.
E-mail addresses: klezovic@dit.unitn.it (D. Kliazovich), granelli@dit.unitn.it (F. Granelli).

1570-8705/$ - see front matter
doi:10.1016/j.adhoc.2005.08.001

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 to the network exceeds the available capacity. Such situation leads to increased buffer space usage in intermediate nodes over the data path, leading to data losses in case of shortage of resources. Transmitted data start 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 communications. Transport-level 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.

The rest of the paper is organized as follows: Section 2 presents an insight related to the nature of congestion, while Section 3 outlines the state-of-the-art in the field of TCP adaptation to wireless links. The proposed approach is detailed in Section 4, and experimental evaluation of its performance is presented in Section 5. Finally, Section 6 draws some conclusions and outlines of future work on the topic.

2. 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 [3,4]. TCP is a reliable connection-oriented byte-stream protocol which performs congestion control dynamically during the data communication process.

TCP congestion control is performed on an end-to-end basis. The receiver provides an acknowledgement (ACK) feedback back to the sender. Relying on the information provided by ACKs, the sender can detect which packets are lost during transmission over the communication path.

The congestion control algorithm employed by TCP is window-based [5]. 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 mechanism for TCP congestion control. TCP uses additive increase and multiplicative decrease strategy for its window adjustment according to network conditions. The main phases of TCP window evolution are presented in Fig. 1.

The connection is initiated with window size equal to one packet (1 MSS—Maximum Segment Size). 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 the estimate of the network capacity. This technique is called slow start phase.

When the *ssthreshold* is reached, TCP enters congestion avoidance phase. The window is increased linearly by one packet for each received ACK. 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 (the receiver's advertised window).

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

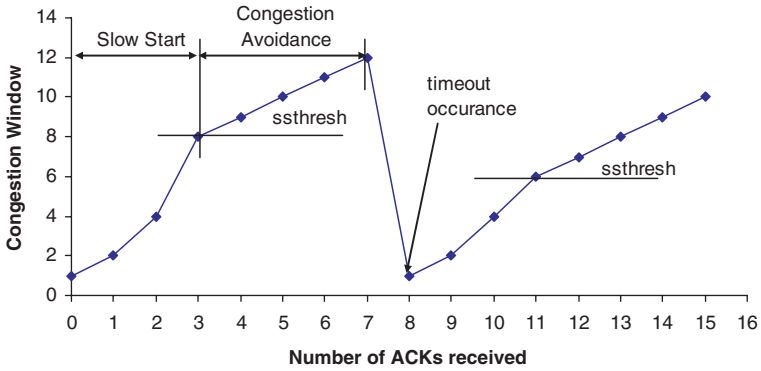


Fig. 1. TCP congestion window (*cwnd*) evolution.

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.

In summary, TCP increases congestion window until the amount of traffic present in the network exceeds the available bandwidth capacity. Upon congestion loss, it reduces the window to the half or to its minimum initial value.

Moreover, 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 [6]. In fact, while the Bit Error Rate (BER) varies from 10^{-6} to 10^{-8} for wired networks, it varies from 10^{-3} to 10^{-1} for wireless ones—several orders of magnitude higher [7]. As a consequence, in wireless networks, TCP reaction to frequent packet losses severely limits the congestion window and thus underestimates the capacity of the networks, leading to non-optimal performance.

The performance of TCP directly depends on the window size, which optimal value should be proportional to bandwidth * delay product of the entire path of the data flow [8]. 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.

The rest of the paper, being focused on a wireless multi-hop IEEE 802.11-based network, assumes that the reader is familiar with general aspects of IEEE 802.11 standard [1] as well as multi-hop routing for wireless LANs [9], which is not covered by the standard. The reader should note that most achieved results and considerations remain valid also in more generic scenarios.

At this point, it is necessary to underline the problems which arise in TCP communications over IEEE 802.11 networks.

The authors of [10] 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 [10] 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. The unfairness phenomenon mentioned above is mainly caused by improper tuning of TCP and link layer retransmission timeouts, multiple collisions present on the link layer as well as well-known IEEE 802.11 MAC unfairness (i.e. the last station is always favored in link access due to the employed exponential backoff scheme).

A new metric, called expected throughput, is introduced in [11] and used as a bound for comparison of the achieved performance for existing TCP implementations in wireless multi-hop scenarios. The simulations show that the performance of TCP, being far from the desired level, is unacceptable in several applications. Similar results are also presented in [12].

The problem of network congestion exists almost from the early beginning of the Internet [13]. Nowadays, many researchers underline that the problem of congestion is even more critical now rather than in 1980s. A relevant theoretical work on the topic presented in literature is [14], where Sally Floyd and Kevin Fall highly motivate the usage of end-to-end congestion control algorithms for the design of future protocols, in order to avoid network collapse due to congestion. Similar results are obtained by the authors of [39] using a game-theoretic approach.

3. 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, with the main focus on performance optimization, aimed at enabling uninterrupted network service provisioning.

The majority of the available solutions which modify congestion control algorithm of TCP can be logically classified into the three following categories: (1) modifications of TCP based only on the information available at the sender node; (2) solutions which enhance the previous category by allowing network feedback; and (3) approaches which obtain bandwidth * delay information by introducing measurement techniques at the transport layer.

3.1. TCP modifications

One of the first approaches to perform precise RTT (Round Trip Time) estimation is TCP Vegas [15]. First, TCP Vegas reads and stores inside TCP header the system clock every time a segment is transmitted. Receiver echoes it back without performing any modification. Then, upon ACK arrival, the sender uses the stored timestamp for RTT calculation. The congestion avoidance algorithm is based on the analysis of the actual throughput achieved by the flow through its comparison with expected throughput. The expected throughput is calculated using measured RTT value and current size of the congestion window. In case the actual throughput is much less than the expected value, TCP Vegas decreases the size of the congestion window assuming that it is sending more data than the available network bandwidth.

Another approach presented by authors of [16] introduces congestion window adjustment based on an end-to-end bandwidth estimation technique. The key idea is in continuous measurements of the rate of returning ACKs at the TCP sender side. After congestion occurrence, the source running TCP Westwood attempts to set a slow start threshold and a congestion window on the basis of the effective bandwidth estimation.

A set of protocols designed for congestion avoidance form the adaptive transport layer (ATL), presented in [17]. ATL consists of two adaptive transport layer protocols: ATL-TCP for reliable communications and ATL-UDP which is used for multimedia data delivery. The main idea of this approach is to allow more freedom in congestion window evolution. The dynamic adjustment algorithm assumes to obtain the information on bandwidth and RTT from the link layer. However, it is not mentioned in the paper the way such information could be obtained by the link layer.

In summary, the solutions within this category attempt to conquer the roots of the problem. The main problem associated with poor performance of transport protocols over wireless networks lies in the fact that they are designed for wired networks—without taking into account limitations of the wireless scenario. In order to solve this problem, the presented solutions redefine transport protocols, replacing them with versions which are designed considering the different characteristics of the wired and wireless scenarios.

Obviously, the solutions within this category provide reasonable performance improvement if compared with traditional wired 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 such as TCP. Thus, a huge effort from joined cooperation of industry and standardization committees is required to bring the proposed modification to the end-user.

3.2. Explicit feedback solutions

Data communication over wireless networks is far from being identified by a simple two-hop scenario. The path of data delivery in most cases consists of several hops with different capacity characteristics. These characteristics include not only communicational parameters such as available bandwidth and delay, but also parameters associated with nodes on the data path—their memory and computational resources.

Solutions of this category allow network to be aware of pending data transmission between any pair of nodes. The main idea is to allow intermediate nodes on the data path to dynamically inform the sender about the amount of resources available through the entire path. Relying on such feedback, the sender can adjust the amount of data sent on the network in order to avoid congestion occurrence.

Random Early Detection (RED) [18] is a scheme which is proposed to deal with network congestion through explicit signaling to the source about growing probability of congestion occurrence. RED is designed to be implemented in intermediate routers, where congestion notification is based on queue monitoring. RED defines two buffer occupancy thresholds: low threshold and high threshold. When the average size of the queue length exceeds the low threshold, the packets start to be dropped with linear probability, proportional to the current average queue length. In case the average length exceeds the high threshold, all incoming packets are dropped.

Congestion is detected by RED through the analysis of the actual queue length, comparing it to the predefined low and high threshold values. While operating with a queue size between these two thresholds, RED informs the sender node about growing probability of congestion occurrence by marking the incoming packets using a specially defined bit in the packet header. Being notified of the network congestion, TCP sender can perform congestion avoidance through congestion window reduction.

Packet drops performed by RED are designed to the purpose of compliance in operation with TCP sources which do not support RED framework.

The specification of the way for Explicit Congestion Notification (ECN) is presented in [19]. The IP layer packets are considered for ECN notification delivery through the ECN bit set. TCP header is modified as well, in order to support following indication of ECN-enabled support of TCP implementation as well as for ECN communication between TCP sender and receiver.

Another approach which targets the reduction of the mismatch between TCP window and the available bandwidth–delay product, called Explicit Window Adaptation (EWA), is presented in [20]. Similar to RED congestion detection, it is based on the analysis of available buffer size in edge routers. EWA attempts to adjust the size of the congestion window explicitly on the data path through modification of receiver's advertised window field of the packet. This scheme generalizes window advertising technique allowing the specification of available buffer space not only by the receiver but also by intermediate routers.

Summarizing, 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.

The main reason for congestion occurrence is the production of more traffic than the available resources for its delivery over the network. In case of a connection covering multiple hops of the network, data transmission is performed on an hop-by-hop basis: data is stored in the buffer, waiting, while node obtains access to the physical medium. In case the amount of incoming data overcomes

the node's forwarding capacity, input buffer size would grow with a rate which is approximately equal to the difference between incoming and outgoing data rates. When the buffer is full, the node starts to drop incoming packets and congestion is detected.

For that reason, 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.

3.3. 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 are targeting an enhancement of transport layer protocols through capacity probing techniques.

The entire capacity of communication path is represented by two parameters: bandwidth and delay.

Delay associated with an end-to-end connection can be easily obtained by TCP through the calculation of the time difference between packet transmission and reception of the acknowledgement generated by the receiver for such packet. The obtained value corresponds to the cumulative delay experienced in forward and backward directions of the connection path. However, it is shown that acknowledgement-based RTT estimation could perform poor under specific circumstances [21]. More accurate RTT estimation is performed when sender includes timestamps into outgoing packets, which are echoed back without any modification by the receiver. The recommendation for high performance extension and finer RTT estimation for TCP protocol is presented in [21].

The second parameter is bandwidth. Different bandwidth estimation solutions available in literature can be logically divided into passive measurements and active probing groups, according to the algorithm they employ [22]. Passive solutions build their measurements based on the trace history of an existing data transmission. Such solutions are limited to the network paths that have recently been used for communication. On the other hand, active probing provides faster and more accurate estimations, while having the potentiality for exploration of the whole network.

The packet pair mechanism is a reliable technique for bandwidth measurement. The key idea of the approach is based on the measurement of inter-arrival time between two back-to-head packets transmitted through the entire connection path. A simple calculation based on inter-arrival delay and packet size gives the bandwidth of the link. A detailed description of packet pair measurement technique is presented in [23,24].

The main drawback of packet pair technique is the dramatic reduction of the estimation accuracy in presence of cross-traffic. CapProbe [25] is a technique which improves packet pair measurements by filtering only those pairs of the packets which have minimal end-to-end delays. This method excludes those packet pairs which are influenced by cross-traffic in intermediate queues.

Summarizing, capacity measurement techniques presented within this section have obvious drawbacks which prevent their successful deployment:

- They simply do not work under certain circumstances, such as under presence of cross-traffic which is both intensive and non-reactive [25].
- Probing network bandwidth requires an insertion of additional traffic, which reduces the already limited network resources.
- The 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.

4. Cross-layer congestion control over multi-hop wireless networks

This section presents a novel scheme (*Cross-layer Congestion Control for TCP*: C^3TCP) for congestion control over wireless local area networks where data delivery is performed over multiple wireless hops. To the purpose of explanation

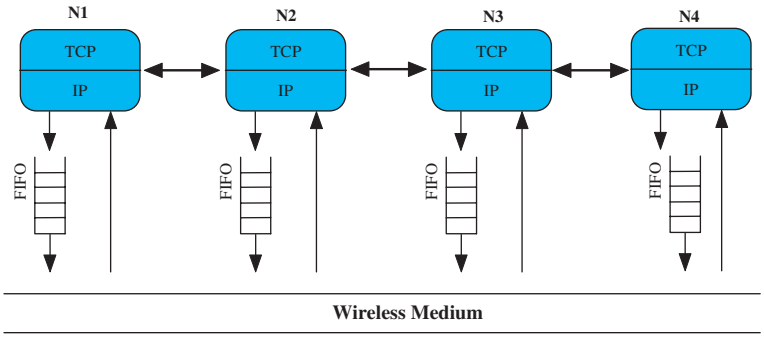


Fig. 2. String topology for a multi-hop wireless network.

of the proposed congestion control ideas, we consider a string network topology as the simplest topology which approximates the multi-hop scenario [26]. A four-node example of the string topology is presented in Fig. 2.

In our string topology, 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 due to transmissions of other nodes. Input queuing is omitted for simplicity of presentation without any influence on the results.

The 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 current paper.

4.1. 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). As a result, the whole bandwidth is divided among nodes which share the same (wireless) medium.

An optional part of the standard specifies RTS/CTS (Request-To-Send/Clear-To-Send) exchange, which takes place prior transmission at the link layer of a data frame and its acknowledgement. This scheme is designed as a solution to the hidden terminal problem. Consequently, it considerably reduces data losses caused by collisions.

From the considerations described above, one can conclude that finally the bandwidth is shared among the nodes which are located in the range of the sender as well as the receiver nodes.

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:

$$B = \frac{D}{T}. \tag{1}$$

The detailed framework for single data packet transmission is presented in Fig. 3. Having data to send at time T_{in}, the source node initiates the medium access procedure: it senses that medium is already occupied by another transmission and falls into exponential backoff with the initial size of the backoff window; during the next time of sensing the medium appears to be free, which means that the source node is allowed to initiate the transmission with RTS frame for medium reservation. Then, after Short Inter-Frame Space (SIFS) the destination node replies with CTS updating the Network Allocation Vector (NAV) of the nodes which are located within the range of the receiver.

At time T_{out}, the sender initiates data frame transmission onto the physical medium. Upon the successful reception of the data frame, the destination node replies with positive acknowledgement.

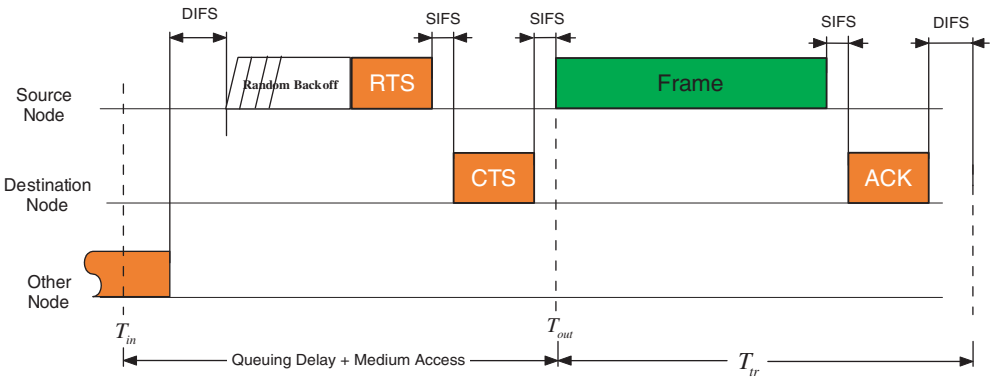


Fig. 3. IEEE 802.11 basic medium access mechanism and data delivery procedure.

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:

$$T = T_{\text{in}} - T_{\text{out}} + T_{\text{tr}}, \tag{2}$$

where the difference $T_{\text{in}} - T_{\text{out}}$ includes data queuing delay corresponding to the time the node was waiting 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 in his theoretical model for the performance analysis of IEEE 802.11 DCF [27].

In order to calculate the time T_{tr} required for data frame transmission and its corresponding acknowledgement, it is necessary to take into account the framework employed by the physical layer.

Data encapsulation performed at both physical and link layers is presented in Fig. 4. Physical Layer Convergence Protocol (PLCP) preamble is transmitted prior any communication between nodes, for the synchronization of their physical circuits. PLCP Header contains signaling information about the subsequent MAC layer frame, such as length and bit rate. PLCP preamble and header are always transmitted at the basic rate, regardless of the maximum bitrate available on the medium.

Fig. 4. Physical and link layer encapsulation.

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 related to both MAC header and frame body.

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:

$$T_{\text{tr}} = T_{\text{data}} + \text{SIFS} + T_{\text{ACK}} + \text{DIFS}, \tag{3}$$

$$T_{\text{data}} = \frac{PLCP.Preamble + PLCP.Header}{Basic.Rate} + \frac{MAC.Header + FCS}{Data.Rate} + \frac{Data}{Data.Rate}, \tag{4}$$

$$T_{\text{ACK}} = \frac{PLCP.Preamble + PLCP.Header}{Basic.Rate} + \frac{ACK.Header + FCS}{Data.Rate}, \tag{5}$$

where DIFS is the Distributed Coordination Function Inter-Frame Space.

The time required for a single data frame transmission T_{data} includes the term which corresponds to physical preamble and header (always fixed size and transmitted at the basic rate). For example, for basic rate of 1 Mbps, the time required for the transmission of physical preamble and header is equal to 192 μs. This means that the value of the first term can be calculated once and then reused for subsequent calculations.

The second term corresponds to the time required for MAC header and CRC information, which could be theoretically transmitted at any rate supported by the standard. 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, for example, by building a short table with values calculated for the entire set of possible data rates for a given physical layer extension of the standard (a, b or g).

The algorithm for bandwidth measurement is the following:

1. Store the timestamp T_{in} for every data arrived to the link layer for further transmission. The data can arrive for forwarding from other nodes or it can be generated locally by upper layers of the protocol stack.
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} using Eq. (3).
3. Calculate the bandwidth experienced by the packet using Eq. (1).

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

The overhead added at physical and link layers is directly related to the utilization level. For example, utilization of wired local area networks is relatively high (97%) [28] while the utilization of wireless IEEE 802.11 networks is on low level [29].

4.2. 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 the four-node example presented in Fig. 2, let us assume that node N1 wishes to communicate with node N4. The data generated by the transport layer of node N1 is placed in the output queue on the link layer. Then, after node N1 gains medium access, the data packet can be transmitted to node N2. Node N2, upon the reception of the data packet which should 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 of node N2. Similar procedures are performed by the node N3 as well, before the data reach the destination node N4.

Finally, in our multi-hop scenario, queuing delay is present on all the nodes except the destination node. Such queuing delay does not correspond to the length of data pipe between end nodes N1 and N4. On the contrary, 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 measurements approximates forward and backward links within a single data pipe.

However, TCP assigns different communicational purposes to links in forward and backward directions. Thus, forward direction is used for transfer of application payload while the backward direction serves for the functionality of the TCP acknowledgement scheme.

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.* According to the considerations described above, the single-hop forward delay experienced by a data burst includes channel access delay, time required for data delivery on the physical layer including related physical and link

layer overhead, but excluding link layer queuing delay.

Forward delay is calculated using (2) 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 $T_{in} - T_{out}$ 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: the end-to-end data pipe is considered artificially longer than it actually is. In other words, a simple insertion of an additional traffic increases the bandwidth–delay product through an increased measured delay. However, the bandwidth–delay product should be decreased in this situation through reduction of the available bandwidth component while leaving forward delay unaffected.

Considerations described above bring advantages for link layer per-hop delay estimation if compared with end-to-end transport layer measurements.

2. *Backward delay.* The reliability of TCP 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. Backward path delay on each single link is calculated using (2).

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.

4.3. "Options" support for IEEE 802.11

The previous two sections describe techniques for available bandwidth and delivery delay measurements at the link layer. Such measurements are performed by wireless stations and correspond to the neighboring links which in fact constitute a single shared medium the nodes belong to.

Wireless multi-hop networks perform data communication over several short-range links. 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 $B_{\text{end-to-end}}$ over an n-hop path is equal to the bandwidth of the bottleneck on the path, while the end-to-end delay $D_{\text{end-to-end}}$ is obtained by the superposition of delays D_i introduced by individual links

$$B_{\text{end-to-end}} = \min(B_1, B_2, \ldots, B_n), \tag{6}$$

$$D_{\text{end-to-end}} = \sum_{i=1}^{n} D_i. \tag{7}$$

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 performed 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.

Optimization of IEEE 802.11 link layer is a hot topic. Huge amount of optimization solutions are proposed by the research community which introduce an enhanced signaling for optimization of the link layer performance. In most cases, enhanced signaling requires the modification of the standardized MAC protocol or the specification of an additional protocol. Indeed, research work continues to go on after the specification of a particular protocol has taken place. Many novel approaches and optimization solutions appear which cannot be easily applied to the existing specification. The idea to include an universal way for inserting additional information has touched most important protocol specifications nowadays. Thus, IPv4 [30], IPv6 [31], TCP [2] specifications contain the support of optional fields.

The proposed modifications are aimed at enabling optional support within the IEEE 802.11 MAC header (Fig. 5).

"Options" is a variable length field which extends standard MAC header. It consists of

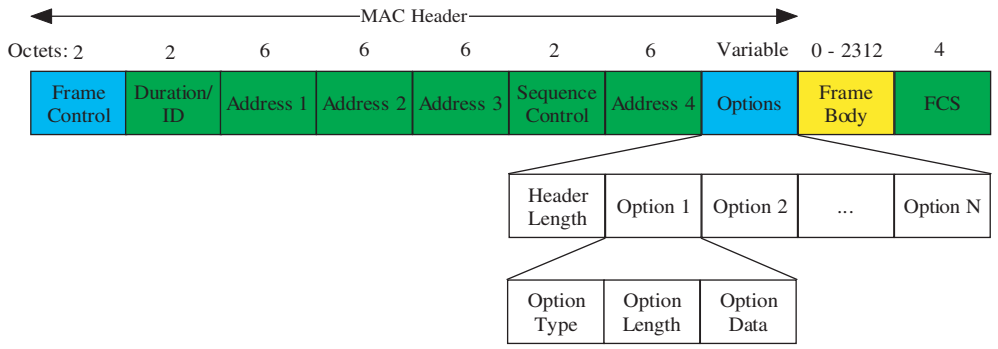


Fig. 5. "Options"-enabled IEEE 802.11 data frame.

"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. Length is required to handle the case when a node does not support the corresponding option. The knowledge of the option's length makes skipping the current option easier, jumping to the next one for processing.

Current wireless devices do not support optional fields within the MAC layer header. In order to provide backward support within the existing IEEE 802.11 standard specification a new type of packet is introduced, since an options-enabled data frame should be of a different type with respect to a normal data frame. For that purpose, one of the reserved types in the Frame Control field of the MAC header of data frames can be used. For example, the type equal to '10' can be used for data transmission, while the subtype '1000' indicates options-enabled data frame.

Backward compatibility with standard IEEE 802.11 devices also requires the specification of communication of options-enabled nodes with those which implement standard MAC. In case options-enabled node wishes to communicate with another node, it should first try to establish communication using the new options-enabled data frame. If the destination node does not support the new type of data frame, it will just simply drop the received frame. The sender node will detect the frame loss through the lack of positive acknowledgement from the receiver after timeout occurrence, which is equal to SIFS + ACK transmission time. The second time the sender node should use standard data frames to communicate with such node.

It could happen that the first communication using options-enabled frames could be unsuccessful because of information corruption during data delivery in the channel. For that reason, the sender node should periodically attempt to communicate using new types of frames.

The presented backward compatibility technique is easy to implement, however it is not optimal in the sense that the sender node needs to probe the destination. These probes could be unsuccessful in case the destination does not support MAC-layer options, producing additional overhead which reduces the bandwidth utilization and increases the packet delivery delay. In order to avoid such additional overhead, the information about options-enabled capability could be encapsulated into route discovery protocol allowing nodes to have knowledge about which type of frame to use in advance.

Options-enabled data frames should not be used in case the node does not transmit any options inside the header.

4.4. The proposed approach: C^3TCP

Fig. 6 presents an example of TCP communication over a 3-hop wireless network. Sender node N1 initiates the transmission by sending a TCP data packet to node N4 over the string topology.

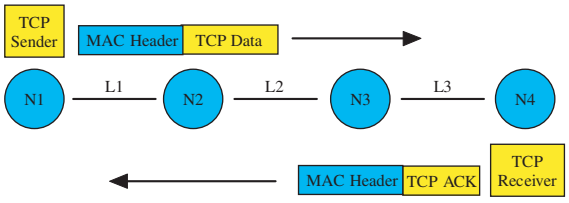


Fig. 6. C[3]TCP usage scenario: TCP connection over 3-hop wireless network.

Upon reception of the TCP data packet, the link layer of node N1 performs bandwidth and delay measurements for link L1. Then, it includes the measured information into the corresponding optional fields inside the MAC header. Node N2, after the reception of the data frame from node N1, performs the same measurements for link L2. Then, it takes the minimal value for the measured bandwidth of links L1 and L2 and updates the bandwidth option in the MAC header. Delay experienced by the data on the link L2 is summed with the delay on the link L1.

When the TCP data packet reaches the destination node N4, its MAC header contains bandwidth and delay experienced by the packet on the path through links L1–L2–L3. TCP receiver in N4 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 N4 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 N1 will have the bandwidth and the delay for both transmitted TCP data and TCP ACK packets. Based on the obtained information, the sender can adjust the outgoing traffic using calculated bandwidth–delay product. The bandwidth is taken only from TCP data packet propagating in forward direction, while the delay is obtained as a sum of propagation delays of TCP data and TCP 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. This module cooperates with the link layer providing congestion control information for the transport layer, using a cross-layer collaboration technique. The implementation of cross-layer signaling is out of the scope of current paper, however a detailed description of existing approaches is provided by authors of [32].

General architecture of C[3]TCP and position of CCM within the protocol stack are specified in Fig. 7. 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 the stored bandwidth–delay information into MAC-layer header of the outgoing TCP ACK.

Modern implementations of TCP support cumulative or selective acknowledgements, which lead to the generation of one TCP ACK packet per several TCP data packets received. In this case, 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 on the forward path 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. In current implementations of TCP, this parameter includes unoccupied buffer space left on 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 correc-

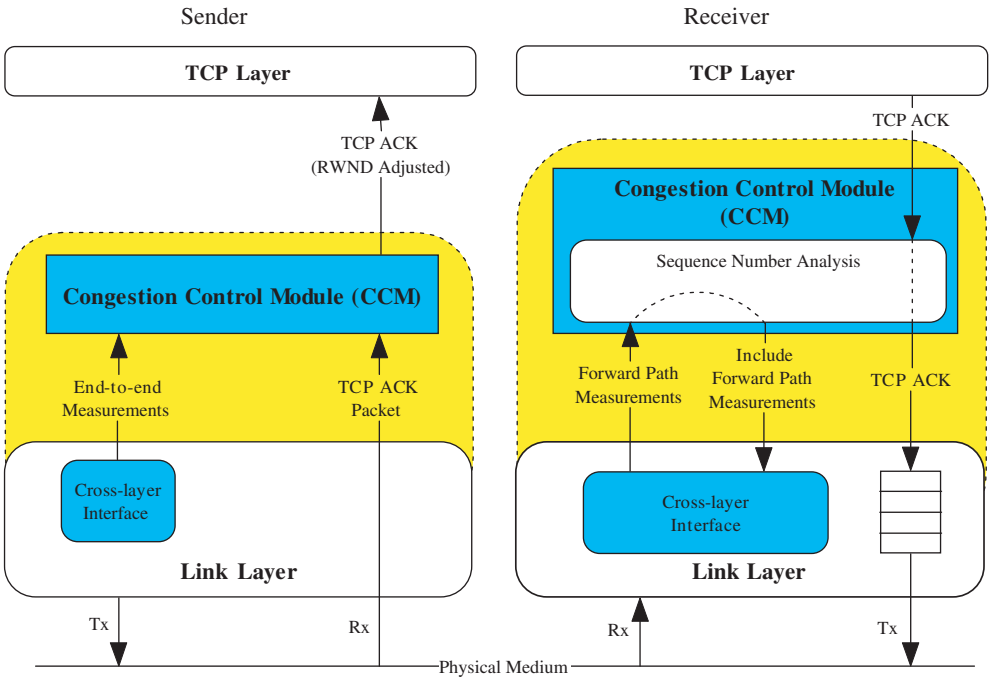


Fig. 7. Congestion control module (CCM) architecture and its position within the protocol stack of C^{3}TCP-enabled nodes.

tion of receiver's advertised window is not a novel approach, on the contrary it is a common technique for the congestion window adjustment—presented in many works. For example, the authors of [20] use RWND field of TCP header 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.

4.5. Multi-node multi-flow scenario

In previous paragraphs, congestion control was mostly focused on the simple single-flow example. However, wireless multi-hop networks are aimed at supporting more complex scenarios, where different nodes initiate transmission of multiple data flows.

Fig. 8 presents a three-flow example of multi-hop communications. Flow F1 shares a part of the data path with flow F2. Flow F3 will also influence flows F1 and F2, since the destination node N6 shares the same medium with nodes N2, N3

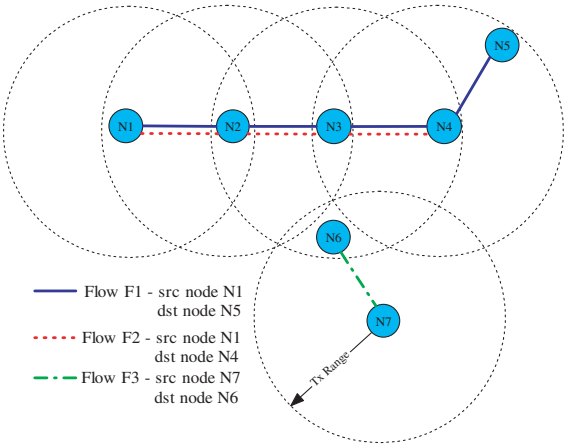


Fig. 8. Multi-flow communications between different nodes of a multi-hop network.

and N4. It is assumed that all nodes use RTS/CTS framework in order to solve the hidden node problem.

Obviously, the bottleneck shared by all three connections is located at node N3. For that reason, the measured bandwidth between nodes N7 and N6 will contain the portion of bandwidth used by flow F3.

Flows F1 and F2 are the flows produced by the same sender node N1. This makes the measured bandwidth equal to the portion of bandwidth jointly occupied by both of them. For that reason, it is not possible to adjust contention windows without performing per-flow differentiation. However, per-flow differentiation can be supported by the C^{3}TCP framework.

The general structure of a possible flow-differentiation module is presented in Fig. 9. The purpose of the Packet Classifier is to differentiate incoming packets according to the data flow they belong to. Then, the transmission scheduler allocates for every flow an appropriate portion of the available bandwidth to the node. The implementation of scheduling algorithm could be simple or contain fairness and Quality of Service (QoS) support.

4.6. Routing

A multi-hop IEEE 802.11-based wireless network is a self-organizing network, where nodes should perform route discovery in order to know the path or at least the next hop of the data communication path.

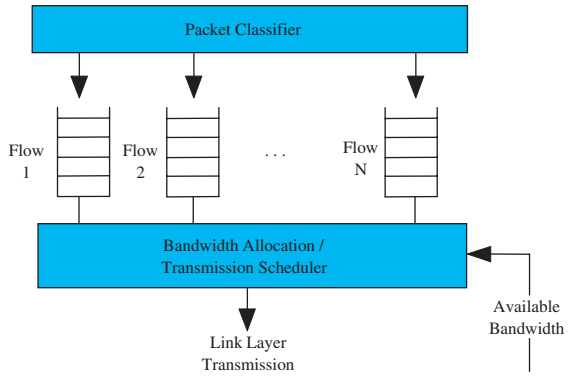


Fig. 9. Flow differentiation for congestion control in the C^{3}TCP framework.

IEEE 802.11 standard does not specify any routing protocol, relying on a simple scenario where all the nodes are located within transmission range of each other. In order to adapt wireless networks to multi-hop scenarios, a growing amount of proposals gave birth to many routing protocols optimized for the wireless environment.

The existing routing protocols are categorized into two major groups: on-demand and proactive. On-demand protocols perform route discovery in the moment of the need for communication, for example AODV [33] and DSR [34]. Proactive approaches like DSDV [35], on the contrary, try to avoid the delay overhead added by initial route discovery keeping the table with routes' description, updated at regular intervals.

However, the main aspect of routing is the metric which is chosen for route selection. Traditional routing protocols rely on shortest path routing, which brings performance optimization in wired networks through improvement in data delivery delay as well as bandwidth utilization. Wireless networks have an additional set of parameters which should be taken into account to make a proper choice. Such parameters include energy constrains, error rate, reliability of links, mobility and available throughput level.

The importance of the last metric is underlined in [36], where the authors introduce an alternative metric for route selection which is called Medium Time Metric (MTM). MTM assigns a weight to each route that is proportional to the time taken for packet delivery over that particular route.

As an extension of MTM metric, we propose to differentiate different routes according to their bandwidth–delay product. Dynamical update of the weight of the routes can be produced based on the values measured with existing data flows of the nodes. Such a way of updating does not produce an additional overhead, in opposite to the case of routing protocol update—leading to an improved efficiency in the utilization of network capacity.

5. Performance evaluation

The performance of the proposed solution is analyzed using the ns-2 network simulator [37].

C^{3}TCP evaluation is performed using two scenarios: the first scenario is used to evaluate the step-by-step operation of C^{3}TCP—showing good agreement with the design objectives, while the second scenario is more complex and better approximates the reality of ad hoc multi-hop network communications.

5.1. String topology

The first simulation scenario consists of four nodes involved in a single TCP connection and two nodes which produce cross-traffic UDP packets (Fig. 10). Node N1 is attached to a TCP agent, while TCP sink is located at the node N4. TCP packets are routed through the intermediate nodes N2 and N3 up to the destination node N4. Due to transmission range limitations, cross-traffic stream shares the same medium with TCP flow only at the link between nodes N3–N4. Each station's transmitting range is limited to 22.5 m, and the distance between each pair of nodes in the simulation scenario is equal to 20 m.

Congestion control module (CCM) is attached at the link layer of end nodes of the TCP connection. At the sender size (node N1), 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] at both physical and link layers, and briefly summarized in Table 1. In order to reduce collisions, RTS/CTS exchange is employed.

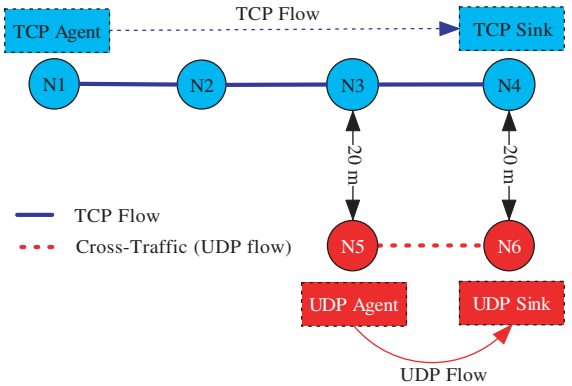


Fig. 10. C^{3}TCP evaluation: Scenario 1.

Table 1
Simulation parameters

Parameter name	Value
Slot	20 μs
SIFS	10 μs
DIFS	50 μs
PLCP preamble + header	192 μs
Data rate	11 Mbps
Basic data rate	1 Mbps
Propagation model	Two-ray ground

TCP flow is started after 3.0 s of the simulation, being initially the only traffic in the network. Cross-traffic produced by node N5 is present only in the interval between 15.0 and 30.0 s of simulation.

In order to evaluate the accuracy of bandwidth measurements provided by the presented technique, the difference between calculated bandwidth and the one obtained at the link level is presented in Fig. 11. The dashed curve corresponds to the available bandwidth on the link, which is calculated without taking into account exponential backoff performed by the nodes as well as in absence of collisions. The results show good approximation achieved by measurements in both cases: with and without cross-traffic.

Another important parameter for TCP performance is the RTT. Fig. 12 presents a comparison between RTT measured at the link level against transport layer measurements. RTT measurements at the transport layer are performed using timestamp options specified in [21].

Transport layer is not aware about the medium it operates on. For that reason, it is not possible

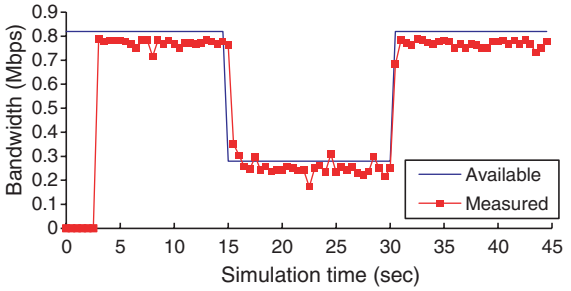


Fig. 11. Accuracy of C^{3}TCP bandwidth measurement.

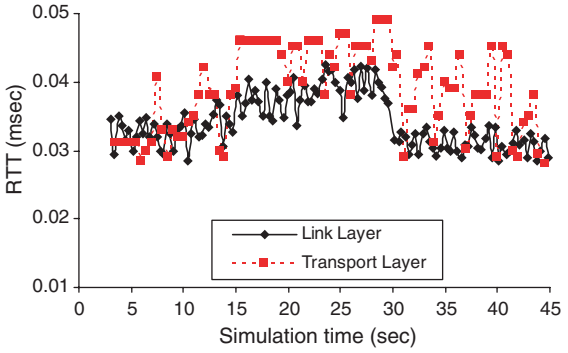


Fig. 12. C^{3}TCP RTT measurements against standard TCP measurements.

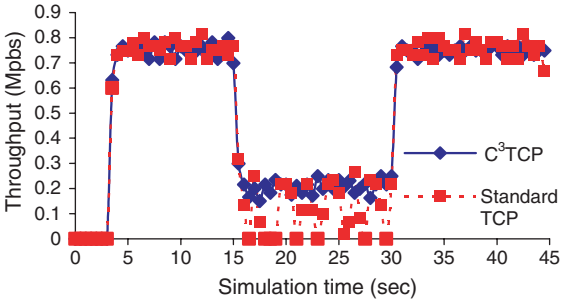


Fig. 13. Throughput of C^{3}TCP against standard TCP implementation.

for standard TCP to differentiate between queueing and transmission delay, as both of them are included into the obtained RTT value.

In case TCP continuously outputs two data packets, the last will stay in buffer waiting for the transmission of the first one. This results in an artificial increase of the RTT delay by a time interval equal to the transmission delay of the first packet over the wireless link. Such situation leads to overestimate the length of available data pipe, which results in an unnecessary increase of the TCP congestion window.

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. The jitter of the link layer curve which corresponds to single TCP flow measurements mostly reflect variations due to exponential backoff. In the presence of cross-traffic (from 15.0 to 30.0 s), the average of RTT measured values is increased mostly due to collisions.

Results presented in Fig. 12 underline the benefits obtained from the C^{3}TCP link layer measurements if compared with those performed at the transport layer, providing good confirmation of the theoretical advantages of link layer measurements described in Section 4.

The major metric for evaluating the performance of TCP flows is the obtained end-to-end throughput. The throughput comparison between TCP version with cross-layer congestion control (C^{3}TCP) and standard TCP implementation [2] is presented in Fig. 13.

Results underline stability of throughput in the case of C^{3}TCP during all the phases of the experiment. The classical implementation of congestion control, on the opposite, always tries to enlarge the window, periodically incurring into congestion.

In more details, in case TCP flow is the only traffic present in network, C^{3}TCP throughput is comparable with one obtained by standard TCP implementation, with less jitter. However, when cross-traffic is present (from 15.0 to 30.0 s), standard TCP flow periodically drops throughput to 0, while C^{3}TCP always keeps the throughput level close to the available bandwidth—showing good utilization of the link capacity.

The bottleneck on the TCP communication path in the evaluation scenario presented in Fig. 10 is the link between nodes N2 and N3. Node N5 produces cross-traffic, generating RTS for obtaining medium access which is heard by node N3 but not by node N2. As a result, node N2 does not receive any response while trying to communicate with the node N3. The communication between nodes N1 and N2 is still possible, since none of them is aware of the cross-traffic and cross-traffic flow does not collide with transmissions of such nodes. The only assumption which is made is that signals transmitted by the nodes do not collide outside effective transmission range (22.5 m). This comes from the implementation details of IEEE 802.11 MAC inside ns-2 simulator.

The TCP source attached to node N1 delivers data packets to node N2 utilizing full bandwidth of the link between nodes N1 and N2. Then, node N2 can forward the received packets to node N3 only after node N5 finalizes its pending transmission. The difference between incoming and outgoing data rates observed by node N2 results in multiple buffer overflows, which finally cause TCP throughput reduction.

Packets dropped from the buffer of intermediate routers are partially transported to the destination. It is demonstrated that multiple drops of such packets could lead to networks collapse [14]. In order to evaluate buffer usage as well as for better understanding the advantages of C^{3}TCP, buffer utilization at the bottleneck node N2 is measured. The evaluated buffer is limited in size—allowing an allocation of up to 20 packets, which is about 5 times greater than the entire link capacity (using 1 K TCP data packets). The obtained results are presented in Fig. 14 for standard TCP and in Fig. 15 for C^{3}TCP.

Buffer usage of standard TCP flow is relatively low (less than 5 packets) when the incoming and outgoing data rates present on node N2 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 s.

On the contrary, knowledge of the capacity of the communication path greatly reduces buffer usage for C^{3}TCP if compared with standard TCP

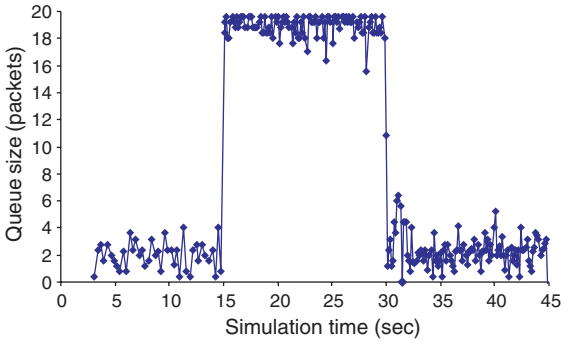


Fig. 14. Buffer utilization at node N2 for standard TCP.

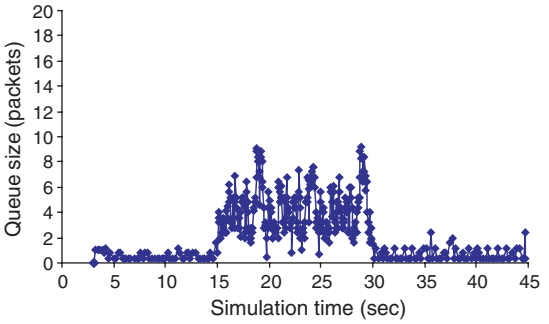


Fig. 15. Buffer utilization at node N2 for C^{3}TCP.

implementation. Thus, buffer in node N2 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. As a consequence, C^{3}TCP scheme does not produce more packets than the network can transport, saving communication resources and avoiding multiple packet drops along the communication path.

Standard TCP was chosen for the comparison as the most wide-spread TCP implementation, in order to underline the obtained performance improvement. However, an additional comparison with other approaches which aim at optimizing TCP congestion control framework is provided in the following.

TCP Vegas [15] and TCP Westwood [16] are chosen among those congestion control solutions which most closely approximate C^{3}TCP from the theoretical point of view. The model of TCP Vegas is taken from standard ns-2 distribution, while TCP Westwood model is obtained from [38].

Throughput comparison results are presented in Figs. 16 and 17 for TCP Vegas and TCP Westwood respectively.

The results show that all the evaluated approaches achieve relatively close (with difference less than 2%) throughput in the scenario when TCP flow is not affected by cross-traffic (from 3.0 to 15.0 and from 30.0 to 45.0 s of simulation).

However, when cross-traffic is present (between 15.0 and 30.0 s of simulation), both TCP Vegas and TCP Westwood periodically drop their throughput down to zero due to overestimation of the link capacity. For clarity of presentation,

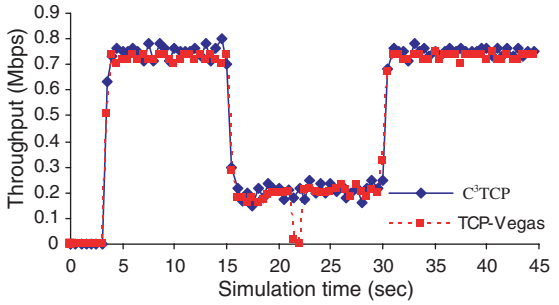


Fig. 16. Throughput of C^{3}TCP against TCP-Vegas implementation.

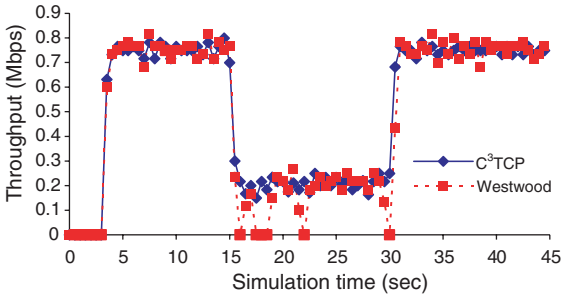


Fig. 17. Throughput of C^{3}TCP against TCP-Westwood implementation.

zoomed portions of the graphs are presented in Figs. 18 and 19. In the considered scenario, TCP Vegas performs better than TCP Westwood. The main reason for that is that TCP Vegas performs comparison between the estimated capacity and the actually achieved throughput. Based of such

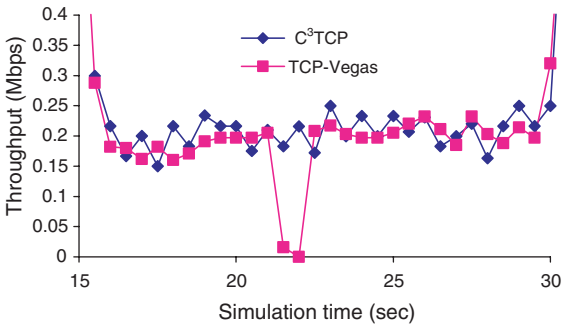


Fig. 18. Throughput of C^{3}TCP against TCP-Vegas implementation (zoomed).

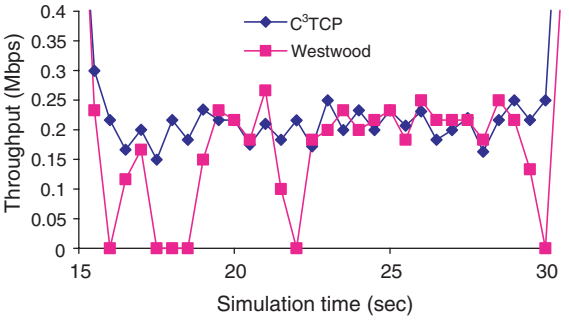


Fig. 19. Throughput of C^{3}TCP against TCP-Westwood implementation (zoomed).

comparison TCP Vegas does not increase congestion windows if this does not lead to a throughput increase. As a result, the more conservative TCP Vegas performs better than TCP Westwood.

The throughput of C^{3}TCP is stable for the entire simulation interval, showing good approximation of the available link capacity. The performed comparison underlines the advantages of the network capacity measurement at the link level rather than at the transport level.

For the purpose of quantitative comparison, results show an improvement achieved by C^{3}TCP of around 27%, 18% and 7% against standard TCP, TCP Westwood, and TCP Vegas, respectively.

5.2. Grid topology

The results presented in Scenario 1 show good agreement with the design principles of C^{3}TCP. However, the simplicity of the scenario does not guarantee similar behavior in the general case of operation in a complex ad hoc multi-hop network.

In order to approach a more realistic case, Scenario 2 specifies a flat-grid topology, consisting of 20 nodes, as shown in Fig. 20. The size of the cell in the grid of 20 m—allowing communication only between neighboring nodes (connected by a dashed line).

TCP agent attached to node 10 and TCP Sink attached to node 14 create a "long" four-hops TCP flow, while a "short" two-hops flow is initiated between nodes 5 and 17. As a result, the first two hops utilized by the "long" flow are shared with the "short" TCP flow.

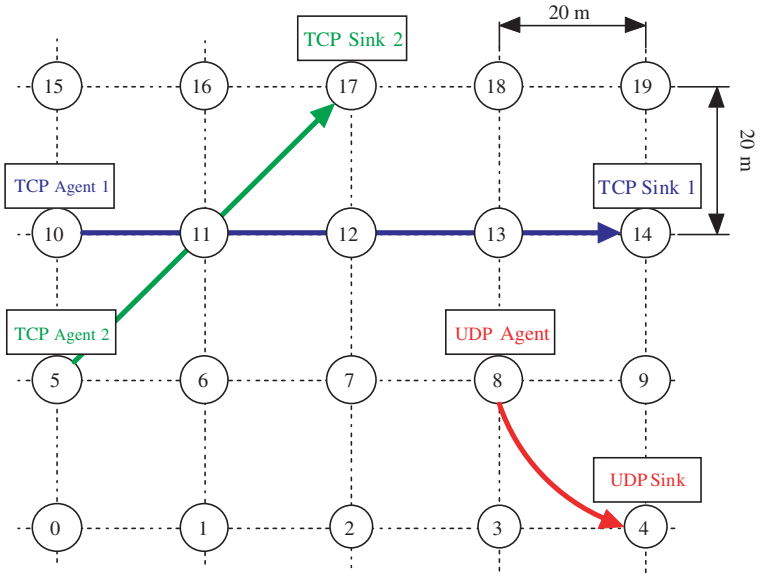


Fig. 20. C^{3}TCP evaluation: Scenario 2.

UDP agent and sink, attached to the nodes 8 and 4 respectively, generate a constant-bitrate cross-traffic flow—which impacts on the "long" TCP flow but not the "short" one.

Simulations are run for 100 s. Uninterrupted TCP traffic flows are started at the beginning of simulation, while the cross-traffic UDP flow is active only in the interval between 30 and 70 s of simulation time. Other simulation parameters as well as configuration of network simulator are fully consistent with those described in Scenario 1.

Obtained simulation results are presented in Fig. 21 for standard TCP, TCP Westwood, TCP Vegas and proposed C^{3}TCP implementations. Within the entire simulation time, standard TCP implementation (see Fig. 21a) continuously tries to increase congestion window on relatively low capacity links. As a result, due to multiple congestion-related packet losses and TCP timeout expirations, the throughput of both flows shows high fluctuations. The "short" TCP flow gets slightly higher average of 0.4 Mbps, if compared with 0.33 Mbps obtained by the "long" flow.

A similar behavior to standard TCP but with lower level of fluctuations is observed for TCP Westwood (see Fig. 21b). Both flows are continuously trying to get full available bandwidth one over another. Periodic throughput reduction present on the graph derives from bandwidth overestimation, caused mainly by the employed type of TCP Westwood ACK filter. However, the large-scale sharing of bandwidth by TCP Westwood flows is relatively fair with an average throughput of 0.4 Mbps per data flow.

Much more stable behavior is observed with TCP Vegas flows (see Fig. 21c). The presented results show relatively large unfairness in sharing the available bandwidth: the "short" flow always show better throughput if compared with the "long" one. Additionally, in presence of constant-bitrate UDP traffic, the throughput of the "long" flow is dramatically decreased (down to zero).

Fig. 21d shows the results obtained with the proposed C^{3}TCP scheme. While not being disturbed by cross-traffic, both flows share the available bandwidth equally—keeping their average throughput at 0.45 Mbps. In the interval between 30.0 and 70.0 s of simulation time, the cross-traffic UDP flow is starting to take a part of the bandwidth from the "long" flow. As a result, the throughput of the "short" flow is increased with

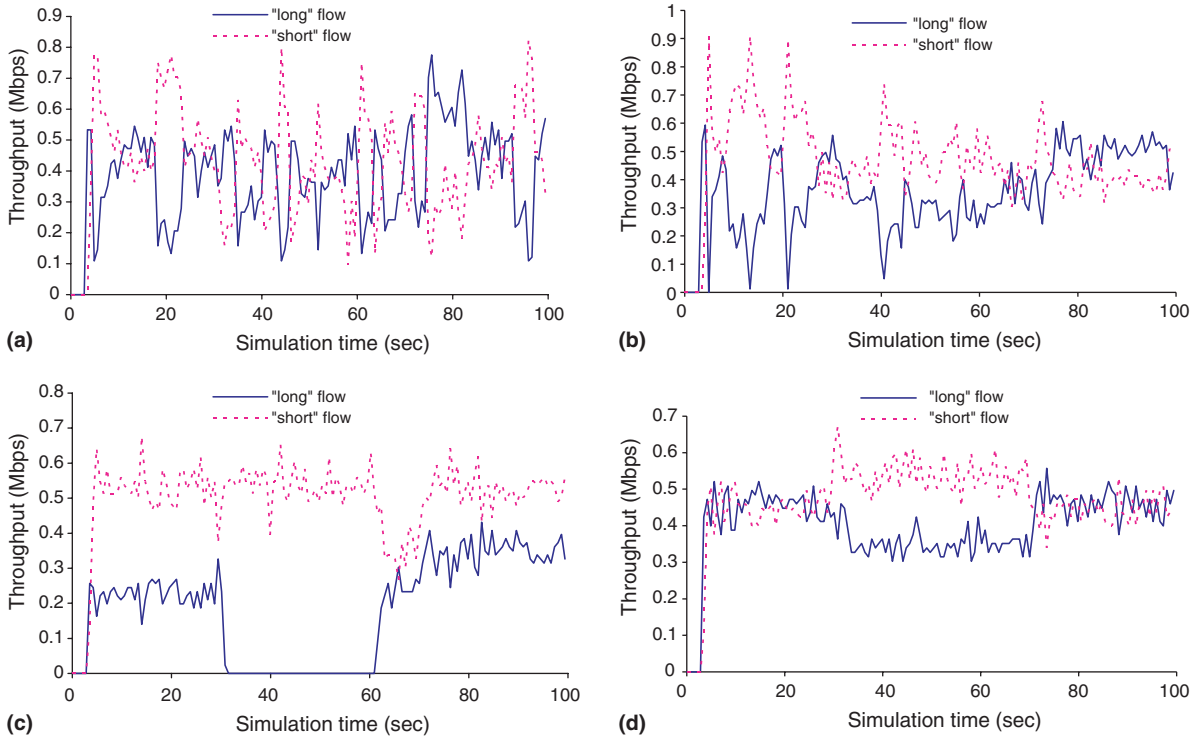


Fig. 21. Simulation throughput results for data flows of (a) Standard TCP (b) TCP-Westwood (c) TCP Vegas, and (d) C^{3}TCP implementations.

the portion of the bandwidth temporarily released by the ''long'' flow.

6. Conclusion

The paper presents the problem of performance degradation of transport layer protocols due to congestion in wireless multi-hop local area networks. Following the 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. An additional contribution of the paper is a proposal to provide optional field support to existing IEEE 802.11 protocol, in order to support the presented congestion control solution as well as many other similar approaches.

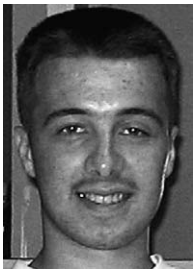
Achieved results underline good agreement with design considerations and high utilization of the available resources. Ongoing work is oriented to a comprehensive evaluation of the presented congestion control technique and a possible proposal to IEEE 802.11 working group to include the support of optional fields into next releases of the standard.

References

[1] Wireless LAN medium access control (MAC) and physical layer (PHY) specifications, IEEE 802.11 standard, 1997.
[2] J. Postel, Transmission control protocol, Request for Comment RFC 793, September 1981.
[3] G. Miller, K. Thompson, R. Wilder, Wide-area Internet traffic patterns and characteristics, IEEE Network (November/December) (1997) 10–23.
[4] N. Brownlee, K. Claffy, Understanding Internet traffic streams: Dragonflies and tortoises, IEEE Communication Magazine 40 (October) (2002) 110–117.

[5] V. Jacobson, Congestion avoidance and control, Computer Communication Review 18 (4) (1988) 314–329.
[6] W. Stevens, TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms, Request for Comment RFC 2001, January 1997.
[7] K. Pentikousis, TCP in wired-cum-wireless, IEEE Communications Surveys (2000).
[8] R. Wang, G. Pau, K. Yamada, M.Y. Sanadidi, M. Gerla, TCP startup performance in large bandwidth delay networks, in: Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2004), vol. 2, March 2004, pp. 796–805.
[9] J. Broch, D. Maltz, D. Johnson, Y. Hu, J. Jetcheva, A performance comparison of multi-hop wireless ad hoc network routing protocols, in: Fourth Annual International Conference on Mobile Computing and Networking (MobiCom'98), ACM, Dallas, TX, October 1998.
[10] S. Xu, T. Saadawi, Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks? IEEE Communications Magazine 39 (June) (2001) 130–137.
[11] G. Holland, N. Vaidya, Analysis of TCP performance over mobile ad hoc networks, Wireless Networks 8 (March) (2002) 275–288.
[12] Z. Fu, X. Meng, S. Lu, How bad TCP can perform in mobile ad hoc networks, in: Seventh International Symposium on Computers and Communications (ISCC), July 2002.
[13] J. Nagle, Congestion control in IP/TCP internetworks, Request for Comment RFC 896, Internet Engineering Task Force, January 1984.
[14] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking 7 (August) (1999) 458–472.
[15] L. Brakmo, L. Peterson, TCP Vegas: end to end congestion avoidance on a global Internet, Computer Communication Review 25 (October) (1995) 69–86.
[16] C. Casetti, M. Gerla, S. Mascolo, M. Sansadidi, R. Wang, TCP Westwood: end-to-end congestion control for wired/wireless networks, Wireless Networks Journal 8 (2002) 467–479.
[17] O. Akan, I. Akyildiz, ATL: an adaptive transport layer suite for next-generation wireless Internet, IEEE Journal on Selected Areas in Communications 22 (June) (2004) 802–817.
[18] S. Floyd, V. Jacobson, Random Early Detection gateways for congestion control for high speed data networks, IEEE/ACM Transactions on Networking 1 (4) (1993) 397–413.
[19] K. Ramakrishnan, S. Floyd, D. Black, The Addition of Explicit Congestion Notification (ECN) to IP, Request for Comments (RFC), September 2001.
[20] L. Kalampoukas, A. Varma, K. Ramakrishnan, Explicit window adaptation: a method to improve TCP performance, Infocom, April 1998.
[21] V. Jacobson, R. Braden, D. Borman, TCP Extensions for High Performance, Request for Comment RFC 1323, May 1992.
[22] H. Ningning, P. Steenkiste, Evaluation and characterization of available bandwidth probing techniques, IEEE Journal on Selected Areas in Communications (JSAC) 21 (6) (2003) 879–894.
[23] K. Lai, M. Baker, Nettimer: a tool for measuring bottleneck link bandwidth, USENIX Symposium Internet Technologies and Systems, March 2001.
[24] C. Dovrolis, P. Ramanathan, D. Moore, What do packet dispersion techniques measure? IEEE INFOCOM (April) (2001).
[25] R. Kapoor, L. Chen, M. Sanadidi, M. Gerla, CapProbe: a simple and accurate technique to measure path capacity, Technical Report TR040001, UCLA SCD, 2004.
[26] M. Gerla, R. Bagrodia, Z. Lixia, K. Tang, W. Lan, TCP over wireless multi-hop protocols: simulation and experiments, in: IEEE International Conference on Communications (ICC), June 1999.
[27] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE Journal on Selected Areas in Communications (JSAC) 18 (March) (2000) 535–547.
[28] D. Boggs, J. Mogul, C. Kent, Measured capacity of an ethernet: myths and reality, Computer Communication Reviews 18 (August) (1988) 222–234.
[29] D. Kliazovich, F. Granelli, Performance improvements in data transfer over 802.11 WLANs, in: 10th IEEE Workshop on Computer-Aided Modeling, Analysis, and Design of Communication Links and Networks, CAMAD'04, Dallas, November 2004.
[30] J. Postel, Internet Protocol: DARPA Internet Program Protocol Specification, Request for Comment RFC 791, September 1981.
[31] S. Deering, R. Hinden, Internet Protocol, Version 6 (IPv6) Specification, Request for Comment RFC 2460, December 1998.
[32] Q. Wang, M.A. Abu-Rgheff, Cross-layer signalling for next-generation wireless systems, in: WCNC 2003, vol. 2, March 2003, pp. 1084–1089.
[33] C. Perkins, E. Royer, Ad hoc on-demand distance vector routing, in: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications, New Orleans, LA, February 1999, pp. 90–100.
[34] D. Johnson, D. Maltz, J. Broch, DSR: the dynamic source routing protocol for multi-hop wireless ad hoc networks, in: C. Perkins (Ed.), Ad Hoc Networking, Addison-Wesley, 2001, pp. 139–172 (Chapter 5).
[35] C. Perkins, P. Bhagwat, Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers, in: Conference on Communications Architectures, Protocols and Applications (SIGCOMM), 1994.
[36] B. Awerbuch, D. Holmer, H. Rubens, The medium time metric: high throughput route selection in multirate ad hoc wireless networks, in: Kluwer Mobile Networks and Applications (MONET), Special Issue on "Internet Wireless Access: 802.11 and Beyond", 2005.
[37] NS-2 Simulator Tool Home Page. Available from: <http://www.isi.edu/nsnam/ns/>, 2000.
[38] TCP Westwood Home Page. Available from: <http://www.cs.ucla.edu/NRL/hpi/tcpw/>.

[39] R. Garg, A. Kamra, V. Khurana, A game-theoretic approach towards congestion control in communication networks, Computer Communication Review (July) (2002) 47–61.



Dzmitry Kliazovich received his Masters degree in Telecommunication science from Belarusian State University of Informatics and Radioelectronics in 2002. He is currently working towards the Ph.D. degree in University of Trento, Italy. His main research interest lies in wireless networking field with a focus on performance optimization and cross-layer design.

Fabrizio Granelli was born in Genoa in 1972. He received the «Laurea» (M.Sc.) degree in Electronic Engineering from the University of Genoa, Italy, in 1997, with a thesis on video coding, awarded with the TELECOM Italy prize, and the Ph.D. in Telecommunications from the same university, in 2001. Since 2000 he is carrying on his teaching activity as Assistant Professor in Telecommunications at the Department of Information and Communication Technology—University of Trento (Italy). In August 2004, he was visiting professor at the State University of Campinas (Brasil). He is author or co-author of more than 40 papers published in international journals, books and conferences, and he is member of the Technical Committee of the International Conference on Communications (ICC2003, ICC2004 and ICC2005) and Global Telecommunications Conference (GLOBECOM2003 and GLOBECOM2004). He is guest-editor of ACM Journal on Mobile Networks and Applications, special issue on "WLAN Optimization at the MAC and Network Levels" and Co-Chair of 10th IEEE Workshop on Computer-Aided Modeling, Analysis, and Design of Communication Links and Networks (CAMAD'04). He is General Vice-Chair of the First International Conference on Wireless Internet (WICON'05).

His main research activities are in the field of networking and signal processing, with particular reference to network performance modeling, medium access control, wireless networks, next-generation IP, and video transmission over packet networks.