Hindawi Publishing Corporation
Journal of Computer Networks and Communications
Volume 2015, Article ID 925207, 18 pages
http://dx.doi.org/10.1155/2015/925207

Research Article

XCP-Winf and RCP-Winf: Improving Explicit Wireless Congestion Control

Luís Barreto

Escola Superior de Ciências Empresariais, Instituto Politécnico de Viana do Castelo, Valença, 4930-678 Viana do Castelo, Portugal

Correspondence should be addressed to Luís Barreto; lbarreto@esce.ipvc.pt

Received 16 May 2014; Revised 18 December 2014; Accepted 18 December 2014

Academic Editor: Tzonelih Hwang

Congestion control in wireless networks is strongly dependent on the dynamics and instability of wireless links. Therefore, it is very difficult to accurately evaluate the characteristics of the wireless links. It is known that TCP experiences serious performance degradation problems in wireless networks. Moreover, congestion control mechanisms that rely on network interaction and network parameters, such as XCP and RCP, do not evaluate accurately the capacity and available link bandwidth in wireless networks. In this paper we propose new explicit flow control protocols for wireless mesh networks, based on XCP and RCP. We name these protocols XCP-Winf and RCP-Winf. They rely on the MAC layer information gathered by a new method to accurately estimate the available bandwidth and the path capacity over a wireless network path. The estimation is performed in real time and without the need to intrusively inject packets in the network. These new congestion control mechanisms are evaluated in different scenarios in wireless mesh and ad hoc networks and compared against several new approaches for wireless congestion control. It is shown that both XCP-Winf and RCP-Winf outperform the evaluated approaches, showing its stable behavior and better channel utilization.

1. Introduction

Reliable transport protocols such as TCP were developed to perform well in traditional wired networks where packet losses occur mostly because of congestion [1]. However, networks with wireless and other lossy links also suffer from significant losses due to bit errors and handoffs. TCP responds to the losses by invoking congestion control and avoidance algorithms. This results in degraded end-to-end performance in wireless systems and networks [2].

Recent efforts to design better congestion control have led to the origin of several explicit-feedback congestion control methods. These methods solicit active multibyte feedback from the routers to the end-hosts, delivering a precise and timely congestion signal that is used to accurately adjust flow sending rates. Therefore, they aim to achieve faster convergence, smaller packet loss rate, high link utilization, and better fairness between flows. Examples of these congestion control mechanisms that rely on network interaction are the explicit control protocol (XCP) [3] and the rate control protocol (RCP) [4]. Their main purpose is to generalize explicit congestion notification, where nodes inform each other about the degree of congestion. XCP decouples channel utilization from fairness control and requires the efficient estimation of the aggregate traffic behavior, that is, both available bandwidth and link capacity. RCP is a congestion control algorithm whose main key is to finish the flows as fast as possible. RCP dynamically updates the rate assigned to the flows, approximating a processor sharing in the presence of feedback. Although the support of explicit congestion control and network interaction is expected to lead to a more efficient congestion control in shared medium systems, such as wireless mesh and ad hoc networks, we have shown in [5] that this is not the case, since they are incapable of predicting effectively capacity in wireless networks, and also that they are not efficient and fair. This is due to the fact that the available capacity at a wireless link depends on the link rates at the neighboring edges. Ignoring this dependence will overestimate the available capacity and lead to poor performance and to instability. We believe that estimating the exact capacity of a link as a function of the link rates at the neighboring edges would provide accurate XCP- and RCP-like scheme to be implemented for wireless multihop networks.

To support efficiency and stability of wireless networks, it is crucial to develop a congestion control scheme which provides efficient and accurate sharing of the underlying network

capacity among multiple competing applications. Factors such as handoffs, channel allocation, and channel quality are directly related to link capacity. Being able to accurately monitor link capacity and available bandwidth and then use that information to perform accurate congestion control is a main challenge in wireless communications. In [6] we proposed the *rt-Winf* algorithm, which is a new wireless inference mechanism, based on *IdleGap* [7], that is able to estimate available bandwidth and path capacity. *rt-Winf* uses the information included in RTS/CTS packets to measure the transmission time and obtain the link capacity and available bandwidth.

In this paper we describe XCP-Winf and RCP-Winf, two new congestion control mechanisms based, respectively, on XCP and RCP, that are extended with the support of an accurate capacity and available bandwidth computation algorithm. The link capacity and available bandwidth obtained through *rt-Winf* are transmitted, through cross layer techniques, to XCP and RCP that use that information in their native congestion control techniques. We show how new XCP-Winf and RCP-Winf functions can be built with this new information. The proposed congestion control approaches are evaluated in different simulated scenarios, mesh and ad hoc, and are evaluated against state-of-the-art congestion control mechanisms. The obtained results show that the integration of *rt-Winf* allows clear improvement of XCP and RCP network performance against standard congestion control protocols and also against specific congestion protocols for wireless environments.

In [8] we presented a base version of both XCP-Winf and RCP-Winf with preliminary results. In this paper we present a complete proposal: (1) we present the *rt-Winf* approach with an evaluation through simulation and emulation in CMU wireless emulator [9]; (2) we describe and refine the conceptual approach of both XCP-Winf and RCP-Winf, with a new mathematical model; (3) we perform a complete evaluation of the congestion control approaches compared against state-of-the-art approaches such as XCP-b [10] and TCP-AP (TCP with adaptive pacing) [11]; (4) we complete the evaluation with the assessment of both XCP-Winf and RCP-Winf in terms of TCP-friendliness.

The remaining of this paper is organized as follows. Next section, Section 2, briefly presents the background and related work. Then, Section 3 describes the *rt-Winf* algorithm and *rt-Winf* obtained results. In Section 4, how *rt-Winf* is integrated within both XCP and RCP is presented. Section 5 describes and discusses the results obtained through simulation, using mesh and ad hoc scenarios with different characteristics. Finally, Section 6 presents the conclusions and future work.

2. Related Work

2.1. Capacity and Available Bandwidth Estimation. *AdHoc Probe* [12], *WBest* [13], *TSProbe* [14], and *IdleGap* [7] are some examples of link estimation mechanisms for wireless networks. While *WBest* calculates both capacity and available bandwidth, *AdHoc Probe* provides only the path capacity of the wireless channel. However, *WBest* is known to be very intrusive when estimating the available bandwidth. *TSProbe* [14] is a new capacity estimation tool based on *AdHoc Probe*, but it is focused on time-slotted connections such as bluetooth or WIMAX. *TSProbe* uses the interaction between several link layer properties to deploy an adaptive probing scheme that utilizes payloads that vary in size. While accurate in time-slotted connections, it lacks efficiency in dynamic wireless environments.

IdleGap takes into consideration the CSMA collision avoidance (CSMA-CA) scheme of wireless networks to obtain its available bandwidth. The idle nodes, which are waiting to transmit, use the network allocation vector (NAV) [15], which shows how long other nodes allocate the link in the IEEE 802.11 MAC protocol. The idle time in the wireless network can then be estimated from the NAV information.

All the presented mechanisms were defined with the main purpose of only estimating available bandwidth and link capacity in specific network conditions. Some mechanisms can only estimate available bandwidth and others only estimate link capacity. In a wireless network environment, it is important to have a tool that can retrieve an accurate estimation of the busy time and the total elapsed time between communications. It is also important to have a mechanism that uses not only source information but also receiver information, as this is the only way to have a precise network status. It is therefore important to introduce the concept of network cooperation in network estimation tools. Another important issue is the effective calculation of the actual data rate that is used by each communication process.

2.2. Congestion Control. The transmission control protocol (TCP) [16] is the most used congestion control protocol in computer networks. However, TCP assumes that the probability of a lost packet is higher than the one of a corrupted packet [17], which is not true in wireless networks. Several congestion control mechanisms were proposed to enhance TCP's behavior in a wireless environment. Mechanisms like TCP-F [18], TCP-ELFN [19], TCP-BuS [20], and ATCP [21] represent some examples of protocols for wireless networks in general. They concentrate on improving TCP's throughput by freezing TCP's congestion control algorithm during link-failure induced losses, especially when route changes occur. These TCP developments differ in the manner in which losses are identified and notified to the sender and in their details of freezing TCP's congestion control algorithm. Even though these schemes do not recognize the need of congestion detection and signaling over a neighborhood, their congestion metric implicitly takes some degree of neighborhood congestion into account.

WCP [22] and WCPCap [22] are congestion control mechanisms developed for wireless mesh networks. WCP is AIMD based and, for every flow, the source maintains a rate R which represents the long term sending rate for the flow. The main issue of WCP is that it does not correctly evaluate the available rate, making an assumption that all links have equal rate. WCPCap estimates the available capacity within each neighborhood and distributes this capacity to contending flows, using a distributed rate controller. Available capacity in

WCPCap is obtained through an analytical model presented in [23], which lacks sender and receiver node cooperation. Moreover, it does not take into consideration all the factors that influence the network dynamics, thus leading to inaccurate and overestimated values. New mechanisms like imTCP (in-line measurement TCP) [24] and TCP-AP (TCP with adaptive pacing) [11] have been proposed. ImTCP introduces a new bandwidth measurement algorithm that can perform in-line measurements. The available bandwidth estimation results from the arrival intervals of ACKs packets. However, these intervals in dynamic wireless environments can suffer high variation, thus introducing lack of accuracy. Another important drawback of imTCP is that it can only estimate available bandwidth. TCP-AP was developed taking only into consideration multihop wireless environments. A TCP-AP sender adapts its transmission rate using an estimate of the 4-hop propagation delay and the coefficient of variation of recently measured round-trip times, being very conservative, and not using efficiently the medium.

For ad hoc networks, other congestion control techniques were developed, such as TPA (transport protocol for ad hoc networks) [25], COPAS [26], and LRED [27]. TPA congestion control mechanism is inspired by TCP, but it is optimized to minimize the number of required packet retransmissions, transmitting blocks using a window-based scheme. As TPA uses blocks on its operation, it can suffer high performance degradation as it does not consider information on link capacity and available bandwidth measurements. COPAS proposes a route selection scheme that attempts to find disjoint paths for different flows by assigning weights to links proportional to the average number of backoffs on the link. COPAS, therefore, needs to use a large amount of network information, thus being very aggressive to the network and requiring a large amount of processing. LRED uses an exponential weighted moving average of the number of retransmissions at the MAC layer as a measure of congestion while marking packets. LRED calculates dropped packets based only on its own perception, which makes LRED behavior inefficient and unstable.

Recent research has recognized the importance of explicitly detect and signal congestion over a network. One example is the explicit wireless congestion control protocol (EWCCP) [28], which identifies the set of flows that share the channel capacity with flows passing through a congested node. EWCCP is, as TCP, an AIMD algorithm based protocol, which lacks the maximization of the network utilization to achieve the highest minimum rate possible. When in more dynamic environments, this is a main issue.

An alternative to AIMD based approaches is schemes in which intermediate routers send explicit and precise feedback to the sources. Examples of such congestion control schemes are the explicit control protocol (XCP) [3] and the rate control protocol (RCP) [4].

XCP was designed to extract congestion information directly from intermediate nodes (routers and/or switches). XCP uses a feedback mechanism to inform the sender about the best network conditions, that is, the maximum throughput. This feedback is accomplished by the use of a congestion header in each packet sent. Along the path, intermediate nodes update the congestion header. When the packet reaches the receiver, it copies the network information, obtained from the last intermediate router, into outbound packets of the same flow (normally acknowledgment packets). WXCP [29] and XCP-b [10] are variants of XCP that measure indirect parameters such as queue sizes and number of link layer retransmissions, using very complex heuristics. The direct estimation of the link capacity will allow a more accurate XCP-like scheme to be implemented for wireless multihop networks.

The rate control protocol (RCP) aims to deliver fast flow-completion or download times. RCP uses the same feedback principle of XCP and tries to emulate processor sharing. However, it uses a different approach: routers along the path do not determine incremental changes to the end-system's throughput but determine the available capacity and the rate at which the end-system should operate. More recently and taking into consideration RCP main properties, for wireless sensor networks (WSN), the wireless rate control protocol (WRCP) [30] mechanism has been proposed. WRCP uses explicit feedback based on capacity information to achieve a max-min fair rate allocation over a collection tree. In WRCP, a receiver capacity model is applied. This model associates capacities with nodes instead of links. The receiver model is also used to develop and implement the explicit and distributed rate-based congestion control protocol for wireless sensor networks.

3. *rt-Winf*

In this section we analyse the main principles and results of *rt-Winf*, since it will be the basis of the new approaches for XCP-Winf and RCP-Winf. *IdleGap* [7] was the underlying basis for the development of *rt-Winf*. The main purpose of *rt-Winf* was to mitigate *IdleGap* main issues, being compatible with all systems and evaluating both the link capacity and the available bandwidth without overloading the network. *rt-Winf* considers the link capacity as the maximum sustainable data rate at a link. The available bandwidth is the unused portion of the link capacity. *rt-Winf* does not affect the OSI model and obtains all of the necessary information to calculate the path capacity and available bandwidth. One of the main issues of *IdleGap* is that it uses the *DataRate* value of the IEEE802.11 header [31], while *rt-Winf* effectively calculates the capacity. The operational principles of *rt-Winf* allow it to rely on the Request To Send (RTS)/Clear To Send (CTS) handshake or on small probe packets.

3.1. RTS/CTS Packets. A RTS/CTS enabled *rt-Winf* mechanism relies on the RTS/CTS handshake to correctly retrieve the NAV values and uses the same node states as defined by *IdleGap*, that is, the *Sender*, *Onlooker*, and *Receiver* states. In the *Sender* state the node is transmitting data; in the *Receiver* state the node is receiving data; and in the *Onlooker* state the node is not participating in the transmission. The RTS/CTS handshake allows *rt-Winf* to immediately start evaluating both link capacity and available bandwidth after the handshake completion time.

Table 1: *rt-Winf* algorithm.

State	Available bandwidth	Capacity
Onlooker	Captured RTS packet? Yes $AB = C_{Onlooker} \times (1 - \frac{\sum NAV_{RTS}}{T_{Total}})$ No $AB = C_{Onlooker} \times (1 - \frac{\sum NAV_{CTS}}{T_{Total}})$	$C_{Onlooker} = \frac{S}{T_{ACK} - T_{CTS} - 2T_{SIFS}}$
Sender	$AB = C_{Sender} \times (1 - \frac{\sum NAV_{CTS}}{T_{Total}})$	$C_{Sender} = \frac{S}{T_{ACK} - T_{CTS} - 2T_{SIFS}}$
Receiver	$AB = C_{Receiver} \times (1 - \frac{\sum NAV_{RTS}}{T_{Total}})$	$C_{Receiver} = \frac{S}{T_{DATA} - T_{RTS} - 3T_{SIFS}}$

The RTS/CTS packets have accurate duration values, which can be used to trigger *rt-Winf* calculations. To understand how RTS/CTS packets can be used by each node state, a set of handshake captures was performed. The obtained captures showed information on how each node state manages the received packets. CTS, DATA, and ACK packets are captured in the case of the *Sender* state. In the *Receiver* state, a node is able to capture the RTS and the DATA packets, while a node in the *Onlooker* state is able to capture the complete set of packets: RTS, CTS, DATA, and ACK. This different knowledge implied the conception of different algorithms for each state. Then, we propose that each node state uses a different method to determine the *Idle Rate*, as can be seen in Table 1.

To obtain the link capacity and the available bandwidth estimations, a node in the *Sender* state relies on the NAV information of the CTS packets. Thus, a *Sender* obtains the link capacity (C_{Sender}) using

$$C_{Sender} = \frac{S}{T_{ACK} - T_{CTS} - 2T_{SIFS}}, \quad (1)$$

where S is the DATA packet size, T_{ACK} is the actual clock time when the ACK packet is received, T_{CTS} is the clock time of the CTS packet reception, and T_{SIFS} is the duration of the occurred short interframe spacing (SIFS). The time when the channel is busy can then be represented by

$$T_{Busy} = T_{ACK} - T_{CTS} - 2T_{SIFS}. \quad (2)$$

When the *Sender* obtains the capacity, it can determine the available bandwidth (AB) by

$$AB = C_{Sender} \times \left(1 - \frac{\sum NAV_{CTS}}{T_{Total}}\right), \quad (3)$$

where NAV_{CTS} is the NAV information on a CTS packet and T_{Total} represents the total elapsed transmission time obtained by the difference between the last captured ACK time and the initial transmission time.

In the *Receiver* state a node uses the NAV information on the RTS packets to obtain the capacity ($C_{Receiver}$), or *Idle Rate*, by

$$C_{Receiver} = \frac{S}{T_{DATA} - T_{RTS} - 3T_{SIFS}}, \quad (4)$$

where T_{RTS} is the time when the RTS packet was received and T_{DATA} is the clock information of the reception of the first DATA packet. The *Receiver* available bandwidth estimation is then calculated through

$$AB = C_{Receiver} \times \left(1 - \frac{\sum NAV_{RTS}}{T_{Total}}\right), \quad (5)$$

where NAV_{RTS} represents the NAV value on the RTS packet and T_{Total} is defined as the difference between the last sent ACK packet time and the initial RTS reception time.

The *Onlooker* state uses the NAV value according to the existence, or not, of the RTS packet to obtain both the available bandwidth and capacity. If a node in the *Onlooker* state captures a CTS packet of a communication without capturing the RTS packet, this implies that the communication is suffering from the *hidden nodes* problem. Thus, the algorithm will only use the NAV from the CTS packet to retrieve the correct values. An *Onlooker* node obtains the link capacity (C) by

$$C_{Onlooker} = \frac{S}{T_{ACK} - T_{CTS} - 2T_{SIFS}}. \quad (6)$$

If the node only captures a CTS packet, the AB is obtained by

$$AB = C_{Onlooker} \times \left(1 - \frac{\sum NAV_{CTS}}{T_{Total}}\right), \quad (7)$$

where T_{Total} is equal to $NAV_{CTS} + T_{CTS} + 2SIFS$. If the *Onlooker* receives a RTS packet, then

$$AB = C_{Onlooker} \times \left(1 - \frac{\sum NAV_{RTS}}{T_{Total}}\right). \quad (8)$$

And T_{Total} is defined as $NAV_{RTS} + T_{RTS}$.

Table 2 shows, for a better understanding of the available bandwidth and capacity evaluation, the variable symbols and their description used in *rt-Winf* calculations.

A general *rt-Winf* system, with its main functioning principles and states transitions, can be observed in Figure 1. The estimation process in the *Receiver* starts when a RTS packet is received. If the node transitions from the *Onlooker*

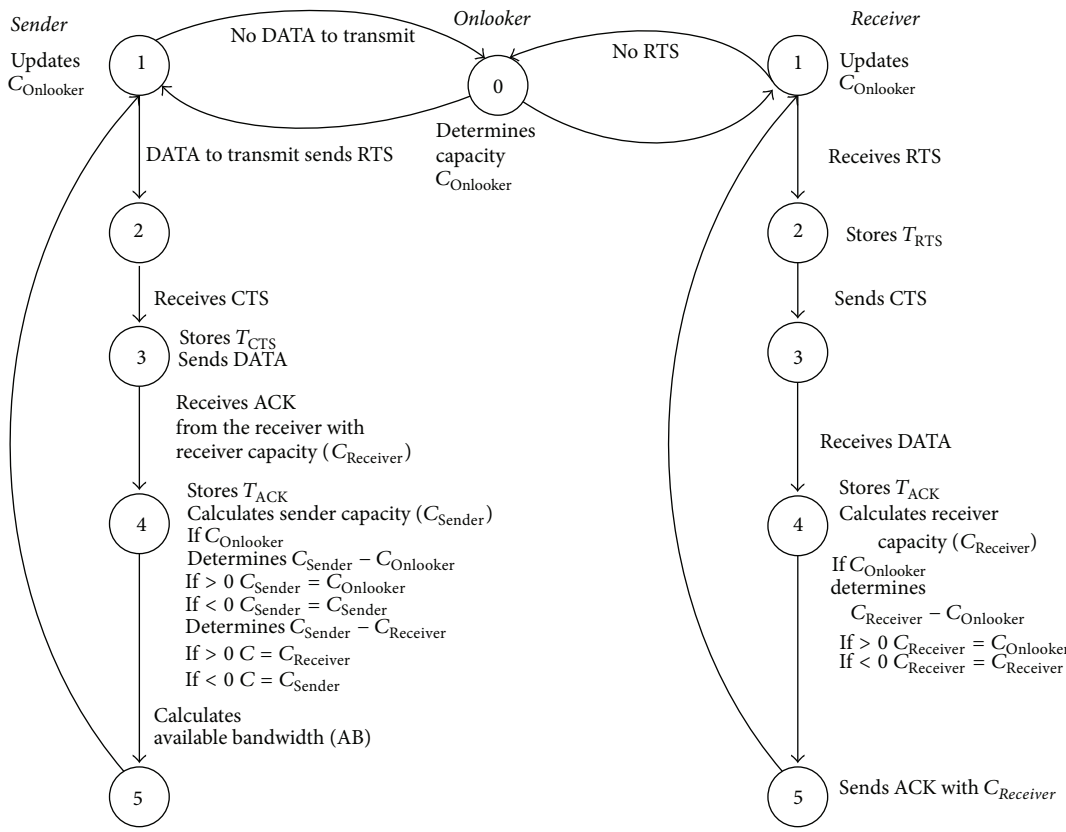


Figure 1: *rt-Winf Sender*, *Receiver*, and *Onlooker* state diagrams.

Table 2: *rt-Winf* variables.

Variable	Description
C	Capacity
C_{Sender}	*Sender* state capacity
$C_{Receiver}$	*Receiver* state capacity
$C_{Onlooker}$	*Onlooker* state capacity
AB	Available bandwidth
S	Packet size
$T_{Transfer}$	Transfer time
T_{Total}	Total elapsed time
T_{ACK}	Time of ACK packet
T_{DATA}	Time of DATA packet
T_{BO}	Backoff time
MSDU	MAC service data unit
NAV_{CTS}	NAV value in CTS packet
NAV_{RTS}	NAV value in RTS packet
T_{CTS}	CTS packet time
T_{RTS}	RTS packet time
T_{MSDU}	Delay per MSDU
T_{80211b}	Maximum 802.11b theoretical throughput

state to the *Receiver* state, it stores the *Onlooker* state capacity ($C_{Onlooker}$) estimated value. First, the node stores the time at which it received the RTS packet (T_{RTS}). Then, it sends a CTS packet to the *Sender* initiating the communication process. When the *Receiver* receives the actual data, it stores its reception time (T_{DATA}) and then, using the corresponding capacity estimation equation, it obtains its link capacity. If the *Receiver* has stored $C_{Onlooker}$, meaning that there was a transition from the *Onlooker* state to the *Receiver* state, the node compares its capacity estimation value ($C_{Receiver}$) with $C_{Onlooker}$. If it returns a positive value, it updates the receiver capacity value to $C_{Onlooker}$; otherwise it uses its capacity estimated value of $C_{Receiver}$. This process is described in (9). The node will always use the smallest value of the two capacities for better channel utilization:

$$\begin{aligned} &C_{Receiver} - C_{Onlooker} \\ &\textbf{if } (> 0) \Longrightarrow C_{Receiver} = C_{Onlooker} \\ &\textbf{else if } (< 0) \Longrightarrow C_{Receiver} = C_{Receiver}. \end{aligned} \tag{9}$$

Then the $C_{Receiver}$ value is sent to the *Sender* on an ACK packet.

In the *Sender* state, and when a CTS packet is captured, the node starts to evaluate the available bandwidth and link capacity. First, the node stores the time at which the CTS packet is received (T_{CTS}) and then starts to send the actual data. When it receives an ACK packet acknowledging data reception, it stores the time at which the ACK packet was received (T_{ACK}) and obtains from the received ACK packet header the $C_{Receiver}$. Then, it uses the corresponding equation of Table 1 to estimate its link capacity (C_{Sender}).

If the node goes to the *Sender* state from the *Onlooker*, it first compares the C_{Sender} with stored capacity C_{Onlooker}. The node has to use the minimum value of the capacities. So, for obtaining that minimum value between C_{Sender} and C_{Onlooker}, the node subtracts the C_{Onlooker} value from the C_{Sender} value. If this value is positive, it means that the minimum value is equal to C_{Onlooker}; thus the node updates the C_{Sender} value to be equal to the C_{Onlooker} value. Otherwise, it maintains as the minimum capacity value its C_{Sender} capacity value. This operation is described in

$$\begin{aligned} &C_{\text{Sender}} - C_{\text{Onlooker}} \\ &\textbf{if } (>0) \Longrightarrow C_{\text{Sender}} = C_{\text{Onlooker}} \\ &\textbf{else if } (<0) \Longrightarrow C_{\text{Sender}} = C_{\text{Sender}}. \end{aligned} \tag{10}$$

Then, it compares C_{Sender} with the one received on the received ACK packet (C_{Receiver}). For better channel use, it has to use the minimum capacity value; this avoids queue fill-ups and bottlenecks. If the node was not previously on the *Onlooker* state, it immediately compares its C_{Sender} capacity estimated value with the C_{receiver} value received on the ACK packet. If it returns a positive value, the sender updates its capacity to C_{Receiver}; otherwise it uses the stored value of C_{Sender}:

$$\begin{aligned} &C_{\text{Sender}} - C_{\text{Receiver}} \\ &\textbf{if } (>0) \Longrightarrow C_{\text{Sender}} = C_{\text{Receiver}} \\ &\textbf{else if } (<0) \Longrightarrow C_{\text{Sender}} = C_{\text{Sender}}. \end{aligned} \tag{11}$$

Finally, and with the stored capacity value, it determines the corresponding available bandwidth. This cooperation process between the *Sender* and the *Receiver* is a great improvement when compared to *IdleGap*. The onlooking capacity is obtained as described in Table 1.

3.2. Probe Packets. If RTS/CTS packets are not present, *rt-Winf* can use probe packets in order to retrieve the transfer time values. Probe packets can be sent between nodes. Probe packets are just sent in the beginning of the communication and, for each new communication, new probe packets are sent. To achieve good results, we use packets with 1500 bytes and frequency of 4 samples before the actual transmission starts (as stated in [12]). It must be noticed, however, that probe packets with reduced sizes can be used. The probe packets must be UDP generated packets with altered frame control IEEE 802.11 header: type data and subtype reserved. We use packets with frame control type set to 10 (data) and subtype to 1001 (reserved). This way the *Sender* and the *Receiver* can successfully differentiate these packets from the ordinary data packets. The IEEE 802.11 standard defines that, for each successfully received packet, it must be sent a MAC ACK packet [31]. The whole process is very similar to the one with the RTS/CTS handshake.

The process for determining the initial capacity lasts the duration of the period that corresponds to sending a packet and receiving its correspondent MAC ACK packet.

The generated packets are used to retrieve the capacity (C) and available bandwidth (AB) values, according to the following:

$$C = \frac{S}{T_{\text{Transfer}}}, \tag{12}$$

where S is the packet size and T_{Transfer}, the transfer time, is equal to $T_{\text{ACK}} - T_{\text{DATA}}$, and

$$\text{AB} = 1 - \left(\frac{\sum_i^p T_{\text{Transfer}}}{T_{\text{Total}}} \right) \times C, \tag{13}$$

where p is the number of packets transmitted and T_{Total} is the total elapsed time since the beginning of the process.

The generated packets are only sent before a node starts a transmission and in the absence of traffic. This allows the system to initially determine the available bandwidth and capacity. Then, the existing traffic and the MAC layer ACK will be used to trigger the calculations. As NAV values are not correctly defined in DATA packets, *rt-Winf* uses clock time information to determine the busy time. Each node state has to manage independently its clock information. Therefore, NAV values are not considered in this specific implementation with probe packets. To be fully operational, both *Sender* and *Receiver* must be running the *rt-Winf* mechanism.

In a normal VoIP call using G.711 codec [32], the overhead introduced by this mechanism is ~1.66%. For a flow with more than 1 Mbps, the overhead is less than ~0.15%.

3.3. rt-Winf Results. We have implemented *rt-Winf* in the CMU wireless emulator [9] and in the ns-2 simulator [33]. The three states defined by *rt-Winf* mechanism and the cooperation between them and between the nodes was developed in *C* language. In base *rt-Winf*, the system is configured with enabled RTS/CTS/ACK handshake packets. In *rt-Winf* probe, probe packets are implemented. The maximum achievable data rate is set to 11 Mbps. Nodes are placed in such a distance that the path loss effect is considered negligible. Path capacity and available bandwidth are evaluated in different scenarios.

For path capacity evaluation, *rt-Winf* results are compared against *AdHoc Probe* results and maximum throughput (that represents the maximal theoretical throughput) in a simple 2 ad hoc nodes testbed. An UDP flow with constant bit rate (CBR) of 64 Kbps is transferred between the two nodes. According to [34], the maximal theoretical throughput is obtained through

$$\text{TH}_{\text{80211b}} = \frac{\text{MSDU}}{T_{\text{MSDU}}}, \tag{14}$$

where MSDU is the MAC service data unit.

The maximum throughput represents, in ideal conditions, the maximum achievable capacity. Figure 2 shows the path capacity results. With a CBR flow in the network, the expected capacity shall be lower than the maximum throughput. The simulations validate this assumption, showing that *rt-Winf* results are close to the maximum throughput values. It is then possible to observe that *rt-Winf* uses efficiently the information present in the channel, in order to obtain the

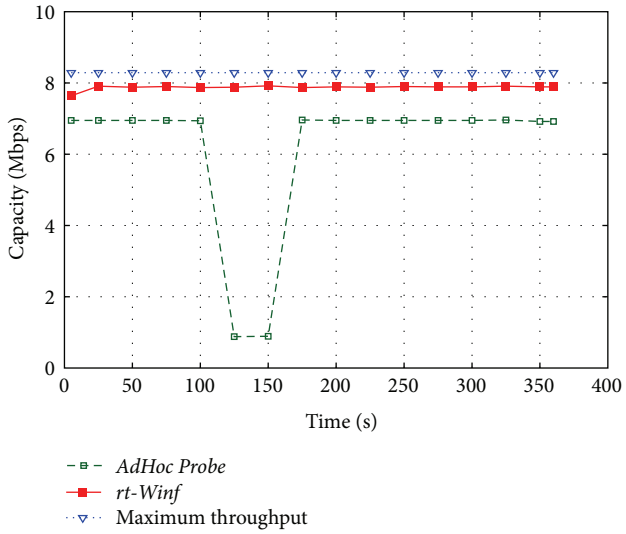


FIGURE 2: *AdHoc Probe* and *rt-Winf* path capacity estimation.

resulting capacity. This is because *rt-Winf* measures more accurately the channel occupation time, as it takes into consideration all traffic flows. Compared to the *AdHoc Probe* mechanism, and with a similar probing time, *rt-Winf* gathers more information to perform the desired calculations, thus being able to be statistically more precise and less sensitive to flow variations. *AdHoc Probe* only takes into consideration its probing packets, which can suffer dispersion and collisions, introducing a negative impact on the capacity evaluation.

Path capacity and available bandwidth evaluations are also conducted on a wireless mesh scenario (Figure 3). The two mobile nodes, *Mobile Node 1* and *Mobile Node 2*, communicate with each other through two mesh nodes that are responsible for the routing and link management. The mobile nodes are in such a distance that the traffic is always routed by the mesh nodes.

Path capacity results are shown in Figure 4, and available bandwidth results are shown in Figure 5. Figure 5 contains the results of *rt-Winf*, *IPerf UDP* [35], and *IdleGap*. Maximum throughput values are also presented, being considered as an upper bound of the result, as described before. *IPerf UDP* results are considered the lower bound.

As observed in Figure 4, *rt-Winf* is less sensitive to variations when compared to *AdHoc Probe*. This is because *rt-Winf* is taking into consideration all packets in the network and is measuring the channel occupation time of each packet, while *AdHoc Probe* is only considering the packets that it generates, thus being more sensitive to flow variations.

The results presented in Figure 5 allow observing how *IdleGap* is not effectively measuring the available bandwidth. *IdleGap* values have a small variation but are near to the *DataRate* value. In *rt-Winf*, it is possible to observe how the results vary through time. Those results are within an upper bound, the maximum theoretical throughput, and a lower bound, *IPerf UDP*.

To observe the impact of *rt-Winf* with probe packets in a wireless mesh scenario and to allow a valuable comparison between the emulator and simulator results, simulations in the ns-2 simulator [33] were also conducted. As *rt-Winf* is based on *IdleGap*, the simulations also allow a baseline comparison of those mechanisms. In the simulations, a FTP transfer from a source to a sink is used, with different simultaneous flows. The maximum throughput is calculated using ns-2 default values and using (14). Figure 6 summarizes the obtained results. Each value is an average of 20 runs lasting 300 seconds of simulated time and nodes are stationary. As observed, *IdleGap* results are almost equal to the maximal theoretical throughput, as it is using the IEEE802.11 header *DataRate* value in the calculations. These results validate the ones obtained with the CMU emulator, since the results for 1 flow in Figure 6 are similar to the ones of Figure 4. In the simulations with *rt-Winf* probes, probe packets with different sizes are used. The results show that *rt-Winf* with probe packets is also efficiently measuring the capacity, and its values are very similar to the *rt-Winf* mechanism working with RTS/CTS control packets.

4. XCP-Winf and RCP-Winf

To improve the performance of congestion control techniques in dynamic wireless networks, we define a new congestion control approach, based on XCP and RCP. The solution proposed adopts the explicit congestion control scheme enhanced with the interaction of a link and available bandwidth estimation mechanism. As both base XCP and RCP do not rely on an effective link capacity estimation tool [36], they begin to use the link capacity at the interface to compute the rate feedback: this introduces capacity overestimation which will generate inflated feedback, and the senders will send more traffic than the link can transfer. In the new approach, *rt-Winf* estimates the available bandwidth that will be used by the congestion control mechanisms to update their transmit rate. The estimation mechanism is integrated both at *Sender*, *Receiver*, and *Onlooker* nodes.

rt-Winf estimation values are obtained in the MAC layer, and therefore this information has to be accessed by XCP-Winf and RCP-Winf. The *rt-Winf* information is sent to the network layer through a simple, but effective, cross layer communication process. For this communication system, a shared database architecture is used, with a set of methods to get/insert information in a database accessible by all protocol layers. One example of such architecture is the *MobileMan* cross layered network stack [37].

A generic XCP-Winf/RCP-Winf mechanism relies on the main functioning principles of XCP and RCP and is represented in Figure 7. *rt-Winf* inserts the available bandwidth and the link capacity information in the shared database and then XCP-Winf and RCP-Winf access that information and use it to update their functions. In the following subsections we present the algorithms performed on both XCP-Winf and RCP-Winf mechanisms.

For a better understanding of the implemented functions described in the following sections, Table 3 shows the variable symbols and its description.

4.1. XCP-Winf Functions. This section describes the proposed XCP-Winf functions. The main changes are performed on the

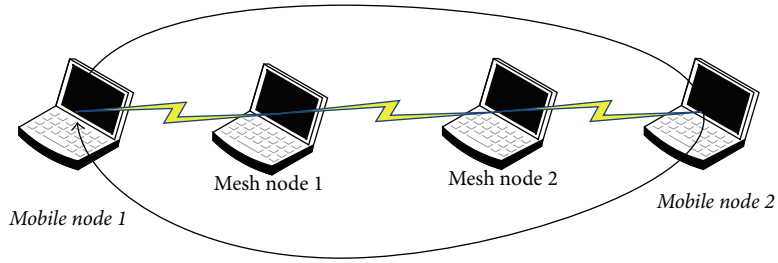


Figure 3: Wireless mesh scenario.

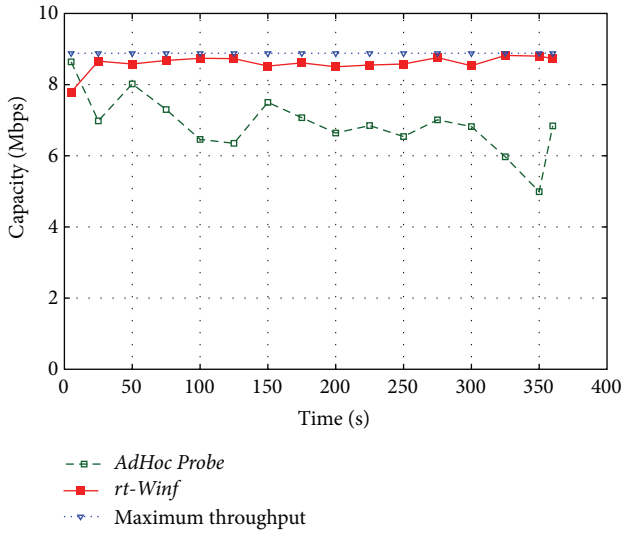


Figure 4: Path capacity in the wireless mesh scenario.

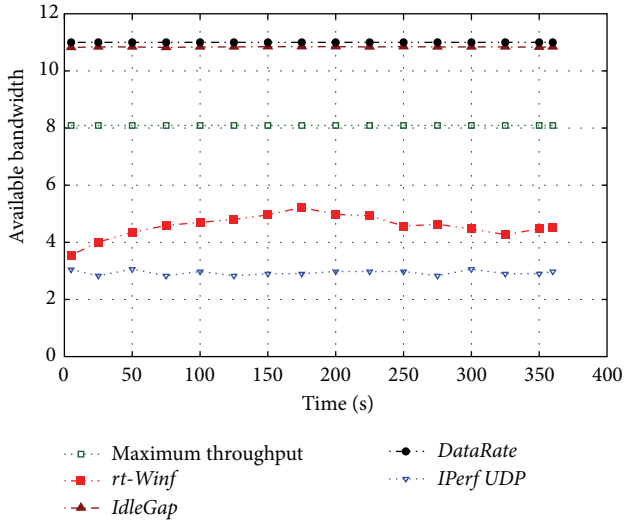


Figure 5: Available bandwidth in the wireless mesh scenario.

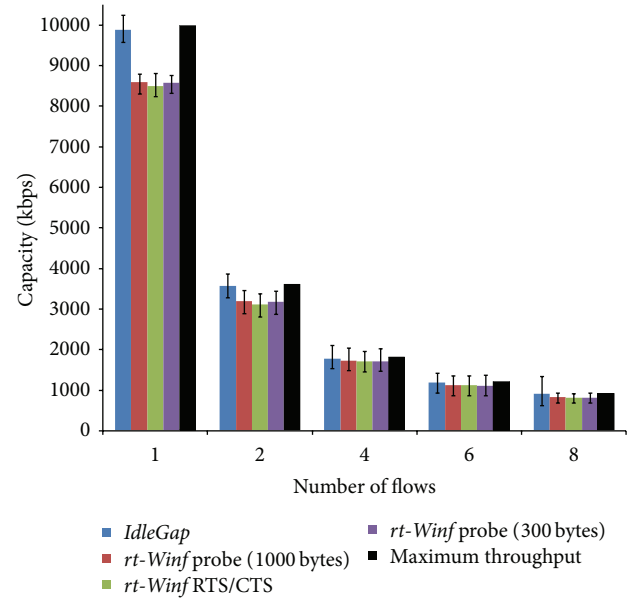


Figure 6: Ns-2 capacity results.

XCP Sender and XCP router functions. The XCP Sender uses the *Sender* state of the *rt-Winf* algorithm and the XCP router uses the *Onlooker* state. The XCP-Winf Receiver is responsible for copying the throughput value required by the sender (in the packet), represented by C_{desired}, to the reverse feedback field of outgoing packets. A XCP-Winf *Receiver* operates in a similar way as a XCP *Receiver*. When acknowledging a packet, the XCP-Winf Receiver copies the congestion header from the data packet to the corresponding acknowledgment packet and acknowledges the data packet in the same way as a TCP receiver.

When operating as a XCP-Winf Sender, several calculations need to be performed for each packet. In a XCP-Winf system, the necessary change in the throughput ($\text{TH}_{\text{change}}$) is obtained by

$$\text{TH}_{\text{change}} = \frac{C_{\text{desired}} - C_{\text{winf}}}{C_{\text{winf}} \times (T_{\text{RTT}}/M)}, \tag{15}$$

where T_{RTT} is the current round-trip time (RTT) and M is the maximum segment size used on the network. C_{winf} is the link capacity obtained from *rt-Winf* and C_{desired} is a desired change in the capacity value that might be supplied by an application or it might be the speed of the local interface. If no additional capacity is needed or desired, C_{desired} will be equal to zero, and the packet will be immediately sent. If the value of $\text{TH}_{\text{change}}$ exceeds the available bandwidth value AB_{winf}, obtained by *rt-Winf*, it is reduced to the current value of AB_{winf}.

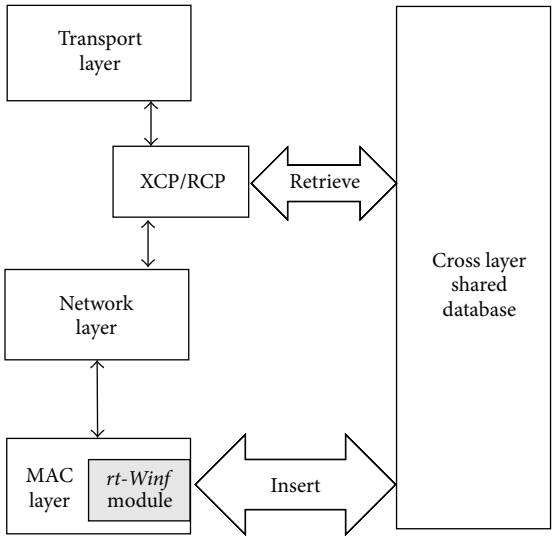


Figure 7: XCP-Winf/RCP-Winf system.

The XCP-Winf router/node system operations in the *Onlooker* state are divided into four moments: when a packet arrives, when a packet departs, when the control interval timeout packet arrives, and when it is required to assess the persistent queue. Once more, *rt-Winf* available bandwidth and capacity are used in the calculations. On a packet arrival, the total input traffic (ϕ) seen at the XCP queue is incremented by the size of the packet received. The sum of the inverse throughput is used for capacity allocation. The inverse throughput constitutes a lower bound for the throughput for any balanced fair network and can be defined as the sum of the inverse residual capacities on the link. The inverse throughput ($TH_{Inverse}$) uses the capacity value obtained by *rt-Winf* and the packet size (S), allowing having more precise values when compared to standard XCP:

$$\sum_{i=1}^{i=n} \mathrm{TH}_{\mathrm{inverse}} = \frac{S_i}{C_{\mathrm{winf}_i}}, \tag{16}$$

where n is the number of packets received. Another important parameter is the sum of T_{RTT} by throughput; this parameter is used to obtain the control interval; on its calculation, it uses *rt-Winf* capacity values:

$$\sum_{i=1}^{i=n} \Gamma + = \frac{T_{\mathrm{RTT}_i} \times S_i}{C_{\mathrm{winf}_i}}. \tag{17}$$

This algorithm also checks if the round-trip time (T_{RTT}) of each flow is exceeding the maximum allowable control interval (T_{ci}), with a default value of 0.5 seconds. If the round-trip time exceeds the threshold, the maximum allowable control interval to avoid delays when new flows are started is updated to

$$\sum_{i=1}^{i=n} \Gamma + = \frac{T_{\mathrm{ci}} \times S_i}{C_{\mathrm{winf}_i}}. \tag{18}$$

When operating on the *Onlooker* state and when the control timer expires, *rt-Winf* values are used to determine the aggregated feedback (F). The aggregated feedback value depends on the link available bandwidth. The aggregate feedback represents the desired variation, on number of bytes, that the traffic is able to allow in a time interval, normally the average RTT. A XCP-Winf router obtains the aggregate feedback [3] based on *rt-Winf* information:

$$F = \alpha \times \left(C_{\mathrm{winf}} - \mathrm{AB}_{\mathrm{winf}}\right) - \beta \times \frac{q}{\overline{T_{\mathrm{RTT}}}}, \tag{19}$$

where α and β are constant parameters, q represents the queue value, $\overline{T_{\mathrm{RTT}}}$ represents the average RTT, and $q/\overline{T_{\mathrm{RTT}}}$ represents the persistent queue. $\mathrm{AB}_{\mathrm{Winf}}$ is the available bandwidth value of the *rt-Winf* mechanism.

Table 3: XCP-Winf and RCP-Winf variables.

Variable	Description
$\mathrm{TH}_{\mathrm{change}}$	Calculated throughput change
$\mathrm{TH}_{\mathrm{inverse}}$	Inverse throughput
ϕ	Total input traffic
H_{RCP}	RCP header size
H_{IP}	IP header size
T_{pacing}	Pacing interval
Γ	RTT by throughput
S	Packet size
M	Maximum segment size
q	Queue size
$\breve{q}$	Instant queue
T_{RTT}	Sender estimate of the round-trip time (RTT)
$\overline{T_{\mathrm{RTT}}}$	Average T_{RTT}
T_{ci}	Maximum allowable control interval
T_e	Queue estimation timer
T_m	Time between the transmission of two consecutive packets
T_{RTT}	Sender estimate of the round-trip time (RTT)
T_{DIFS}	IEEE 802.11 DCF interframe space time
T_{SIFS}	IEEE 802.11 short interframe space time
T_{backoff}	IEEE 802.11 backoff time
C_{winf}	*rt-Winf* obtained capacity
C_{desired}	Packet desired capacity change
C_{φ}	Amount of capacity shuffled
C_{change}	Allocated feedback change
C_{ω}	Weighted link capacity
C_{extra}	Extra bandwidth consumed due to backoff
$\mathrm{AB}_{\mathrm{winf}}$	*rt-Winf* obtained available bandwidth
F	Aggregated feedback
R	RCP rate
p_f	Positive feedback factor
n_f	Negative feedback factor
f_p	Positive feedback
f_n	Negative feedback
$\mathrm{CW}_{\mathrm{Sender}}$	Sender congestion window

Then, as stated in [3], the amount of capacity that will be shuffled (C_φ) in the next control interval is obtained. This allows new flows to acquire capacity in a full loaded system. This parameter is obtained by $\max(0, 0.1 \times \mathrm{AB}_{\mathrm{winf}} - |F_{\mathrm{winf}}|)$. This will allow obtaining the increase—positive feedback scale factor (p_f)—or decrease—negative feedback scale factor (n_f)—on the traffic:

$$p_f = \frac{C_\varphi + \max(F, 0)}{\sum \Psi},$$
$$n_f = \frac{C_\varphi + \max(-F, 0)}{\phi}. \tag{20}$$

When a packet departs, the node has to calculate a per-packet capacity change that will be compared to the T_{change} value in the packet header. As stated in [3], "using the AIMD rule, positive feedback is applied equally per flow, while negative feedback is made proportional to each low's capacity." The allocated feedback (C_{change}) for the packet is the positive per-packet feedback (f_p) minus the negative per-packet feedback (f_n). The positive feedback is obtained using p_f and the flows interpacket time; then

$$f_p = p_f \times \frac{S}{C_{\mathrm{winf}}}. \tag{21}$$

The negative feedback is obtained using the packet size and the n_f:

$$f_n = n_f \times S. \tag{22}$$

Thus, the capacity change requested is

$$C_{\mathrm{change}} = f_p - f_n. \tag{23}$$

This value may be positive or negative. The node verifies whether the packet is requesting more capacity (via the packet's C_{desired} field) than the node has allocated. If so, this means that the sender's desired throughput needs to be reduced and verified against *rt-Winf* available bandwidth ($\mathrm{AB}_{\mathrm{winf}}$). If the node has allocated more capacity than the available bandwidth, the desired throughput is updated to the *rt-Winf* available bandwidth. If the allocated capacity is less than the available bandwidth, the C_{desired} field in the packet header is updated with the feedback allocation.

XCP-Winf, as XCP, needs to calculate a queue that does not drain in a propagation delay, which is the persistent queue. This queue is intended to be the minimum standing queue over the estimation interval. Each time a packet departs, the queue length ($\breve{q}$) is checked and the minimum queue size is calculated. When the queue estimation timer T_e expires, the persistent queue length is equal to the minimum queue value over the last T_e interval. For obtaining the duration of the T_e interval, the capacity value of *rt-Winf* is used:

$$T_e = \max\left(q, \left(\overline{T_{\mathrm{RTT}}} - \frac{\breve{q}}{C_{\mathrm{winf}}/2}\right)\right). \tag{24}$$

Comparing XCP-Winf with XCP, it is possible to conclude that both use the same principles but differ in the way link capacity and available bandwidth are obtained and used.

4.2. RCP-Winf Functions. RCP-Winf updates RCP operations, using *rt-Winf* link capacity values. A RCP-Winf Receiver operates in the same way as a standard RCP Receiver. The RCP-Winf Receiver just updates the RCP congestion header with the bottleneck rate, that is, the rate of the most congested link, in the ACK packet, and then sends it to the sender.

RCP-Winf relies only on link capacity evaluation and there is no need for determining the aggregate feedback (F); thus there is also no need for explicitly using the available bandwidth. A RCP-Winf implementation also keeps unchanged the standard operations performed by RCP routers when a packet arrives and departs.

When operating as a sender, RCP-Winf needs to perform operations that allow it to modulate the congestion window. The RCP-Winf Sender will evaluate the desired change in throughput C_{desired}, through the value obtained in the ACK packet congestion header field and the link capacity obtained by the *rt-Winf*, gathered through the cross layer communication process:

$$C_{\mathrm{winf}} - C_{\mathrm{desired}} \leq 0. \tag{25}$$

According to this evaluation, RCP-Winf will update or not the desired change in throughput. If the evaluation returns a negative value, the C_{desired} value will be updated to the C_{winf} value. Then, it modulates the sender congestion window ($\mathrm{CW}_{\mathrm{Sender}}$):

$$\mathrm{CW}_{\mathrm{Sender}} = \frac{C_{\mathrm{desired}} \times T_{\mathrm{RTT}}}{M + H_{\mathrm{RCP}} + H_{\mathrm{IP}}}, \tag{26}$$

where H_{RCP} is the RCP header size (12 bytes) and H_{IP} is the IP header size. Then, it calculates the pacing interval (T_{pacing}) that will be used to send packets from the queue:

$$T_{\mathrm{pacing}} = \frac{M}{C_{\mathrm{desired}}}. \tag{27}$$

When the rate timer of a RCP-Winf router expires, the node first gets the *rt-Winf* capacity values. Then, it assumes that the aggregate incoming traffic rate is defined by the *rt-Winf* capacity value (C_{winf}). Next, it obtains the average round-trip time of the traffic that has arrived in the rate estimation interval (T_e). After that, the node updates the RTT estimate ($\overline{T_{\mathrm{RTT}}}$) and then it updates the rate value that will be offered to the flows (R) using the *rt-Winf* capacity:

$$R = R \times \left(1 + \left[\left(\left(\frac{T_e}{\overline{T_{\mathrm{RTT}}}}\right) \times \left(\alpha \times (C_w \times C_{\mathrm{winf}} - C_{\mathrm{winf}}) - \beta \times \frac{q}{\overline{T_{\mathrm{RTT}}}}\right)\right) \times (C_w \times C_{\mathrm{winf}})^{-1}\right]\right). \tag{28}$$

The node tests the rate value and updates it. The rate value cannot be under a minimum rate value ($R_{\min}$) or above a weighted (C_w) link capacity value:

$$\begin{aligned} &\textbf{if } (R < R_{\min}) \Longrightarrow R = R_{\min} \\ &\textbf{else if } (R > C_w \times C_{\text{winf}}) \Longrightarrow R = C_w \times C_{\text{winf}}. \end{aligned} \tag{29}$$

Then, the node decides the length of the next rate estimation interval. Before finishing, the node resets the variables and restarts the timer. C_w controls the target link-utilization and can be any value in the range $0.95 < C_w < 1$. It is important to choose a value less than 1 as it allows some comfort to drain excess traffic before building up a queue. In extreme congestion scenarios, the minimum rate value allowed ($R_{\min}$) is considered to be

$$R_{\min} = \frac{(00.1 \times \text{MTU})}{\overline{T_{\text{RTT}}}}, \tag{30}$$

where MTU is the maximum transmission unit. The result of the operation will be the value of the minimum rate.

A RCP-Winf router also has to perform per-packet operations, namely, when a packet arrives and when a packet departs. Whenever a packet arrives carrying a valid round-trip time, its value is added to the stored sum of RTTs and the number of packets carrying a valid RTT is incremented. This allows for a more precise calculation of the average RTTs. This is also the normal operation of a RCP standard system. RCP-Winf router uses the obtained values of *rt-Winf* as their underlying value. When a packet requests an unspecified value or a value that exceeds the link capacity, the system is updated with the *rt-Winf* obtained capacity value; that is,

$$C_{\text{desired}} = C_{\text{winf}}. \tag{31}$$

5. Simulation Results

This section introduces the simulation setup to evaluate the performance of the proposed congestion control mechanisms and the simulation results. The results are obtained using the ns-2 [33] simulator. To evaluate the performance, three metrics are used: throughput, delay, and number of received packets. The proposed control mechanisms, XCP-Winf and RCP-Winf, are evaluated against the base protocols, TCP, XCP, and RCP, and against the XCP-b and TCP-AP protocols, which are especially developed for wireless networks.

Different wireless mesh and ad hoc scenarios were used. The parameters of the simulations are presented in Table 4. The configured default transmission range is 250 meters, the default interference range is 500 meters, and the channel data rate is 11 Mbps. For the data transmissions, an FTP application with packets of 1500 bytes or a constant bit rate (CBR) application is used. The mobility is emulated through the ns-2 *setdest* tool to provide a random node movement pattern. We configure *setdest* with a minimum speed of 10 m/s, a maximum speed of 30 m/s, and a topology boundary of 1000 × 1000 meters. All results were obtained from ns-2 trace files, with the help of *trace2stats* scripts [38] adapted to our own needs. The routing protocol used was the destination-sequence distance-vector (DSDV) [39]. The presented results show the mean values obtained through different simulation runs with different seeds and the 95% confidence interval.

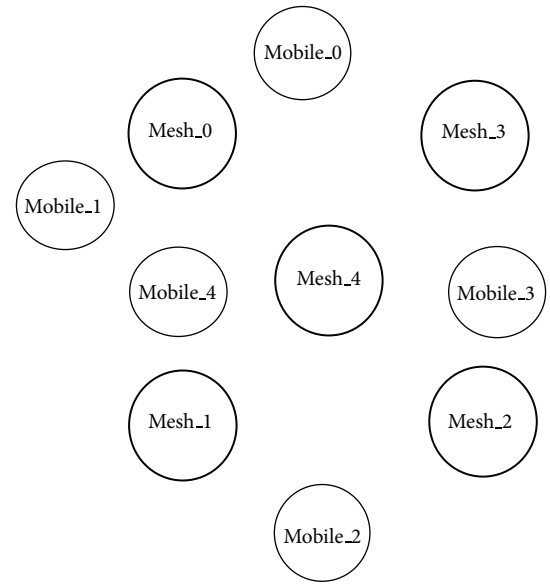


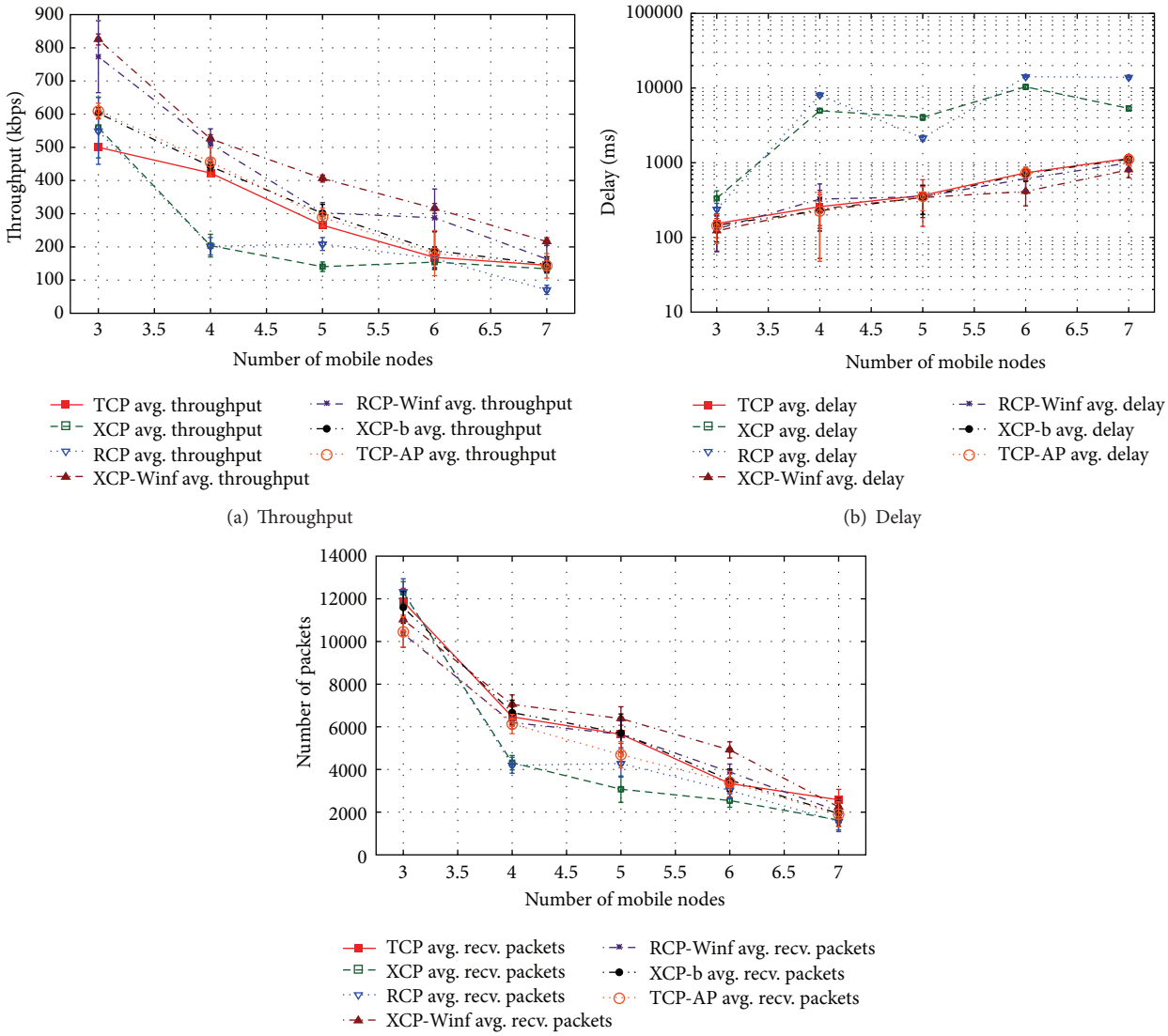
Figure 8: Topology of 5 mesh nodes, 5 mobile nodes.

Table 4: Simulation environment.

Simulation parameters	
Topology area	1000 m × 1000 m
Simulation time	300 sec.
Simulation repetition	30 times
Ad hoc scenario number of mobile nodes	8, 16, 32, 64, 128, 256
Mesh scenario number of mobile nodes	3, 4, 5, 6, 7
Mesh scenario number of mesh fixed nodes	5, 9, 12, 16
Mesh nodes position	Random
Path loss model	Two-ray
Mobility model	Random way point
Maximum movement speed	30 m/s
Mac layer	IEEE 802.11
Propagation model	Two-ray ground
Routing protocol	DSDV

5.1. Mesh and Ad Hoc Scenarios Results. Next we present, analyze, and compare the results of the congestion control approaches in the mesh topology scenario. The mesh topologies defined comprise a grid of 5, 9, 12, and 16 fixed mesh nodes. In all mesh topologies, a combination of 3, 4, 5, 6, and 7 mobile nodes is used. Figure 8 represents a mesh topology of 5 mesh nodes and 5 mobile nodes. The mobile nodes are simultaneously sources and sinks. The results show throughput, delay, and the number of received packets.

Figures 9(a), 9(b), and 9(c) show the previously referred performance metrics for five different scenarios. In each scenario, a fixed number of 16 mesh nodes and a variable



(a) Throughput

(b) Delay

(c) Received packets

Figure 9: Mesh scenario: 16 mesh nodes, variable number of mobile nodes.

number, from 3 to 7, of mobile nodes were used. Each mobile node, as previously stated, is simultaneously sending and receiving data.

The obtained results show that the integration of *rt-Winf* in XCP and RCP improves significantly their behavior, which makes XCP and RCP behave more efficiently and with better channel utilization, which also leads to less channel losses (more received packets). The use of *rt-Winf* in the mesh nodes (*Onlooker* state) makes the feedback mechanism more accurate, as all nodes in the network can determine available bandwidth and capacity and send that information to the other nodes that are participating in the communication. Both XCP-b and TCP-AP have better results than TCP and standard XCP and RCP. However, they are outperformed by both XCP-Winf and RCP-Winf. TCP-AP is the most conservative one and obtains worse results on the number of received packets. XCP-b relies on the maximum buffer size of nodes, and therefore, with the current scenario conditions, XCP-b is less efficient and less accurate than both XCP-Winf and RCP-Winf.

To evaluate XCP-Winf and RCP-Winf when using CBR applications, an UDP application (simulating a VoIP application) is configured, for the 16 mesh nodes scenario and variable number of mobile nodes. Figure 10 shows the obtained results. Without *rt-Winf* enabled, XCP obtains better results than RCP for a lower number of mobile nodes. This is due to the fact that RCP was developed having in mind Internet bursts traffic. With less mobile nodes exchanging information, the number of collisions is lower, and less retransmissions and burst traffic are present in the network. It is also possible to conclude that both XCP and RCP are not evaluating correctly the link capacity and do not have the necessary mechanisms to overcome this situation. With *rt-Winf*, the throughput results are considerably better. However,

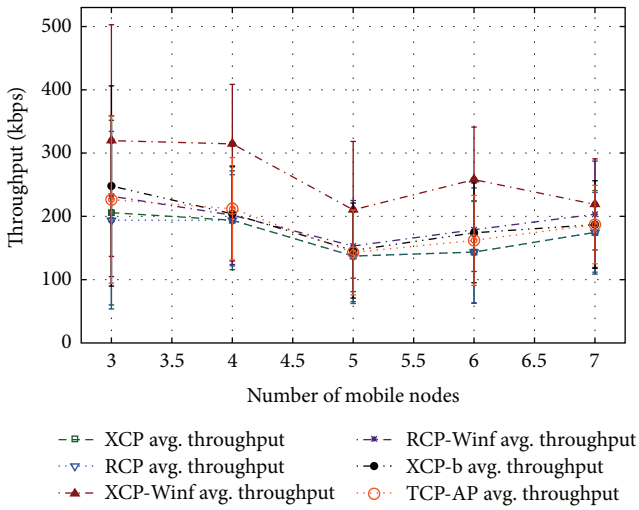


Figure 10: CBR throughput in mesh scenario: 16 mesh nodes, variable number of mobile nodes.

it can be seen that XCP and RCP have lower performance in controlling congestion when the traffic is UDP.

Once more, RCP-Winf reflects its base development for bursty traffic. The CBR application is sending data at a constant rate; with more mobile nodes sending data, more collisions will occur and more bursts of traffic will be present in the network. This situation will allow RCP to react more precisely and, with more mobile nodes, to have better throughput results. The results also show that XCP-b is better suited when the number of mobile nodes is small. The results show that TCP-AP, though specially developed for wireless environments, has very poor results. This is because a TCP-AP sender adapts its transmission rate using an estimate of the 4-hop propagation delay and the coefficient of variation of recently measured round-trip times. This means that TCP-AP is very conservative, not using efficiently the medium.

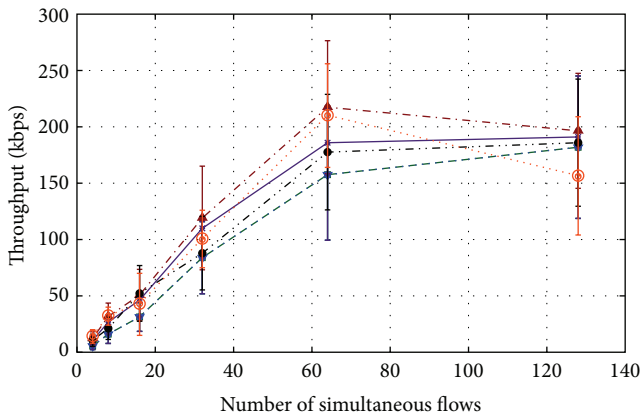
The congestion control approaches are also evaluated in ad hoc scenarios using CBR UDP 64 Kbps flows. The scenarios are composed of 8, 16, 32, 64, 128, and 256 nodes, and for each scenario there are 4, 8, 16, 32, 64, and 128 simultaneous flows. The flows are randomly generated through the ns-2 *gencbr.tcl* tool. The mobility was also dynamically generated through different seed values. The obtained results are presented in Figures 11(a), 11(b), and 11(c).

From the obtained results, it is possible to conclude that standard XCP and RCP have the same behavior when the traffic is UDP; however, the integration of *rt-Winf* makes them react differently as they both use the information from the MAC sublayer in a different way. It is also possible to see that, with *rt-Winf* integrated, both XCP and RCP can receive more packets, which reflects a lower rate of lost packets. This is due to the fact that XCP-Winf and RCP-Winf, with accurate link capacity and available bandwidth, are using more efficiently the medium and improving each node queue management. As more packets are transmitted, more throughput is obtained and the medium is better used: it is possible to infer that both XCP-Winf and RCP-Winf are more stable and fair. In the same conditions, it is possible to send more information with a higher rate. It must be noticed that the results also reflect the mobility randomness, where more nodes are in each other's influence area. Another factor that is influencing the results is the routing information and the exchanged routing messages: as flows increase, the collisions and delay also increase, which is also reflected in throughput values. Once more, XCP-b obtains good results when the network is not heavily utilized, where XCP-b increases its available bandwidth and, consequently, its rate. However, with more nodes and flows in the network, XCP-b behavior is less efficient due to the higher number of losses, reducing the flow rate. With respect to TCP-AP, its behavior is also degraded as the number of flows increases: while the throughput results are improved, they are obtained with less received packets.

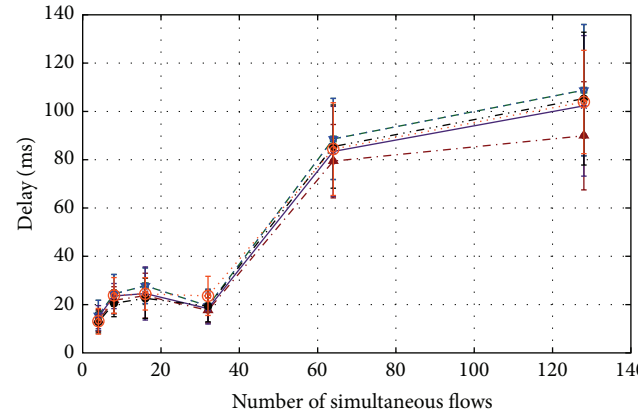
5.2. Building Scenario Results. For a more real evaluation, we defined a new mesh network scenario to simulate a public building, with public services and a public garden (Figure 12). According to Table 4, the different parameters are the following: the numbers of mobile nodes are 10, 20, and 30; the fixed mesh nodes are 6 and two different flows are used. In this scenario, mobile nodes start to transmit in a random way and their transmission lasts 240 seconds. The mesh nodes position is randomly defined in the inside area. A randomly chosen mesh node is shut down for 100 seconds during the simulation period. Two types of flows are used, to represent a light traffic flow (4 CBR 64 Kbps flows, sent each 400 ms) and a heavy traffic flow (4 CBR 128 Kbps flows, sent each 100 ms).

The obtained results of the light traffic flows are shown in Figures 13(a), 13(b), and 13(c); the results of the heavy traffic flows are presented in Figures 14(a), 14(b), and 14(c). As can be observed from the results, the integration of *rt-Winf* in both XCP and RCP improves their standard behavior. Since the nodes that are not participating in the communication enter the *Onlooker* state and evaluate the network performance, it is possible to have a state by state and rate by rate overall performance evaluation. As *rt-Winf* uses three different states and network cooperation, it is possible to have a more effective and efficient hop by hop performance evaluation. This results in a more efficient evaluation and use of the channel capacity. As XCP-Winf and RCP-Winf also have more capability to adapt to the changing conditions of the network, this is expressed in better transmitting rates and better channel usage. XCP-b has again poor performance for high load scenarios: XCP-b is not taking into consideration packet loss, considering packet loss as a buffer overflow, thus having a more inefficient behavior and introducing unnecessary capacity slowdowns on the network. TCP-AP presents good overall results. However, it obtains those results with a considerable reduction in received packets, which indicates that TCP-AP is more conservative and is not efficiently using the medium.

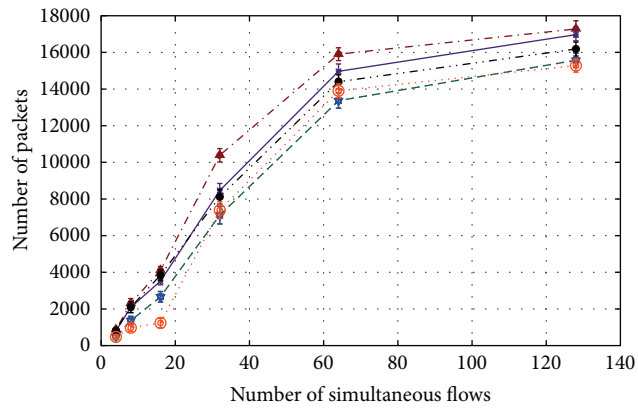
5.3. Utility Results. As TCP is the most used and deployed congestion control protocol on the Internet, it is important (as described in [40]) to analyze how XCP-Winf and RCP-Winf



(a) Throughput



(b) Delay



(c) Received packets

Figure 11: Ad hoc scenario: variable number of flows.

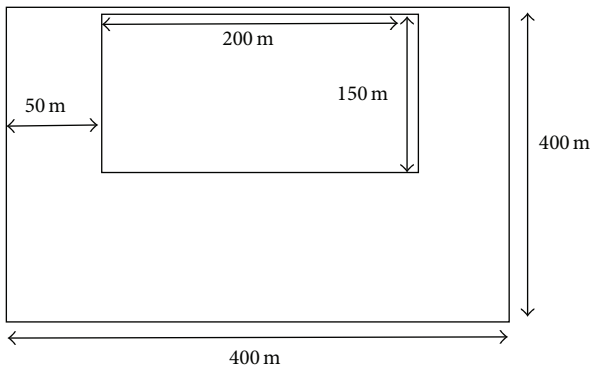
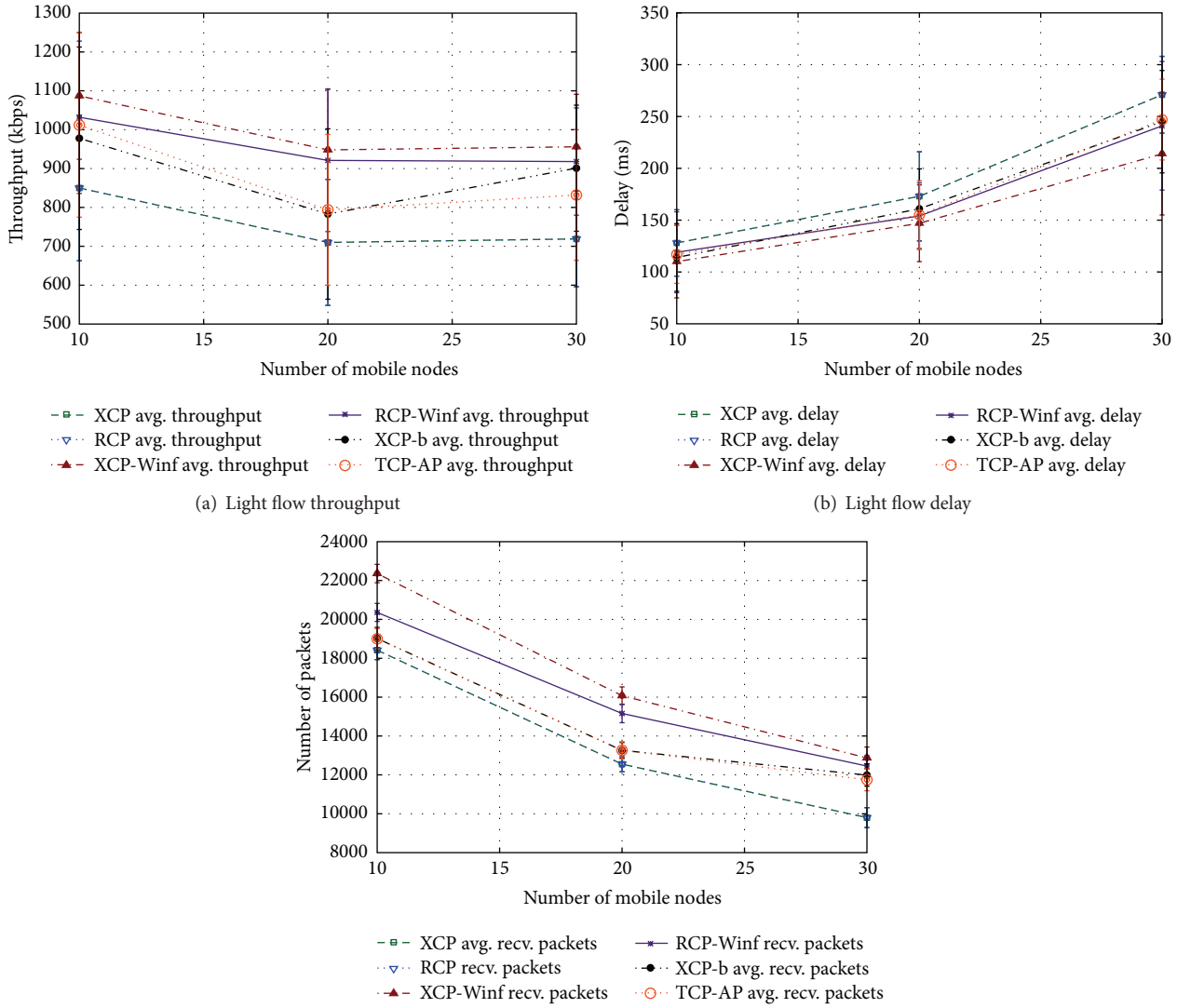


Figure 12: Building layout simulation.

flows interact and compete with TCP. For this purpose, we use the average data rate over time for each flow, thus allowing observing how bandwidth is being managed between TCP and the *winf* proposals. This is called *utility* of a networking protocol.

Two scenarios were defined: one using XCP-Winf and TCP and the other using RCP-Winf and TCP. The two scenarios consist of a 1000 m × 1000 m area, divided into three distinct parts: an area of 250 m × 250 m where there are two mobile node sources, one with TCP and the other with XCP-Winf or RCP-Winf; a middle area of 500 m × 500 m with two mobile nodes with the *rt-Winf* mechanism activated (the average data rate is measured on these two nodes, as they will have TCP and *winf*-like flows competing); finally, another area of 250 m × 250 m for the mobile nodes sinks. Each source generates two FTP flows with packets of 1500 bytes. The simulation lasts for 120 seconds.

Figures 15(a) and 15(b) show the obtained results. It is possible to observe that on both situations TCP flow grows faster and gains more bandwidth on the beginning: this is



(a) Light flow throughput

(b) Light flow delay

(c) Light flow received packets

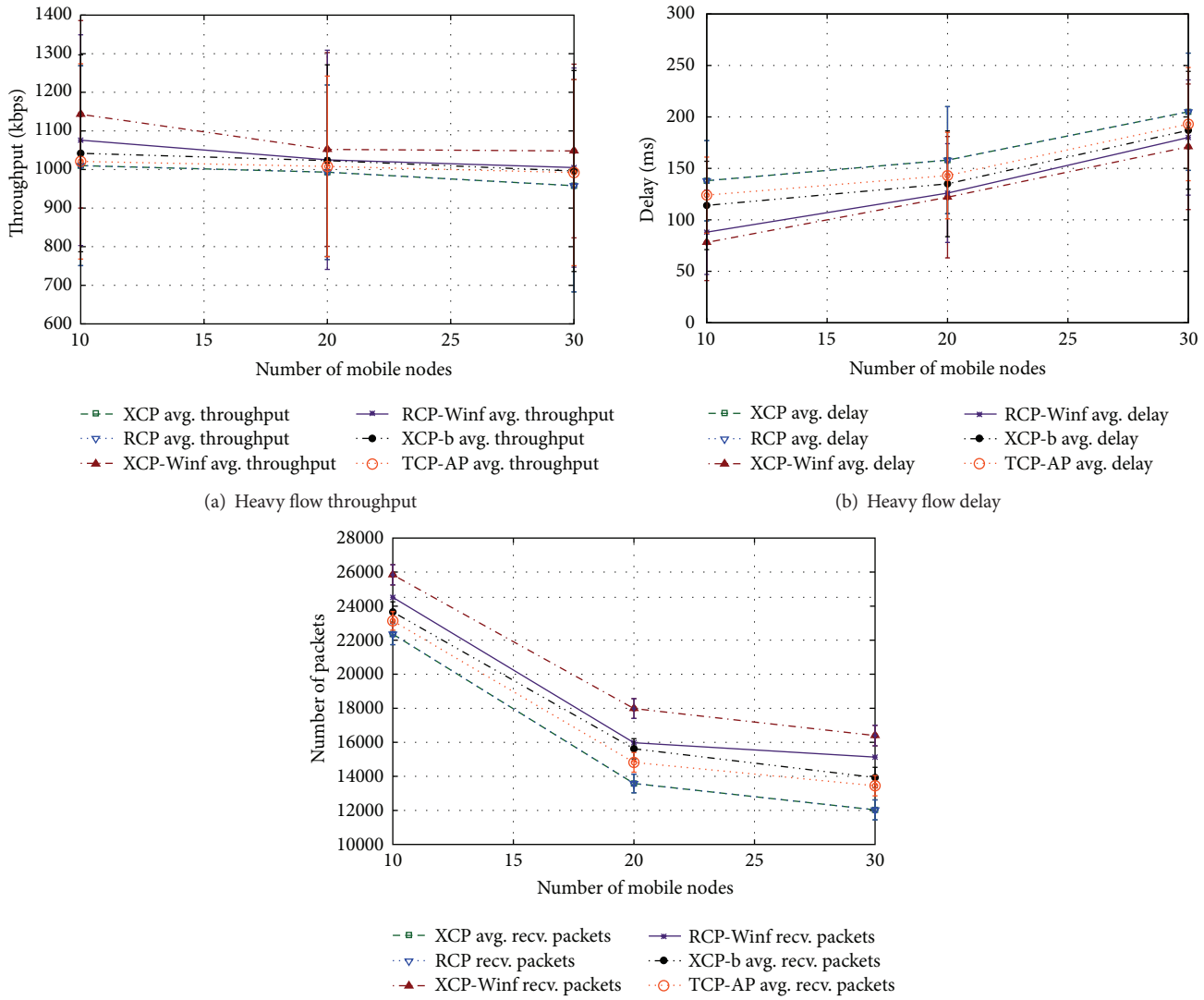
FIGURE 13: Public building light flow: variable number of mobile nodes.

because TCP's congestion mechanisms are not evaluating correctly the network status (TCP is trying to fill the nodes queues, expecting that packet loss due to queue overflow will indicate congestion). However, RCP-Winf and XCP-Winf are evaluating, and measuring consistently, how the network is behaving and adjusting the bandwidth requirements to that information. As RCP-Winf is based on RCP with its bursty traffic development, it has a more unstable behavior during the initial instant of the simulation; as more traffic is present on the network, RCP-Winf becomes more stable.

The results also show that the *winf* mechanisms are TCP-friendly on the long term and are adjusting to the unfairness nature of TCP. XCP-Winf reacts earlier to these constraints and starts to compete for the same bandwidth as TCP earlier than RCP-Winf. However, it must be noticed that the results show that both RCP-Winf and XCP-Winf take some time to allow a fair share of bandwidth between their native flows and TCP. It is advisable that this fair share is obtained as quickly as possible, thus allowing a more efficient share and coexistence of network resources.

6. Conclusions

In this paper we presented a study that demonstrates that using MAC layer information in congestion control, through a cross layer communication process, can be an important factor of network performance improvement. This MAC layer information resorts to CTS/RTS/ACK messages handshake or on small probes to know each node's channel allocation, and it allows accurate determining of the links capacity and available bandwidth. This information is then used by congestion control mechanisms based on explicit congestion notifications, XCP and RCP, to accurately determine the network status and act accordingly.



(a) Heavy flow throughput

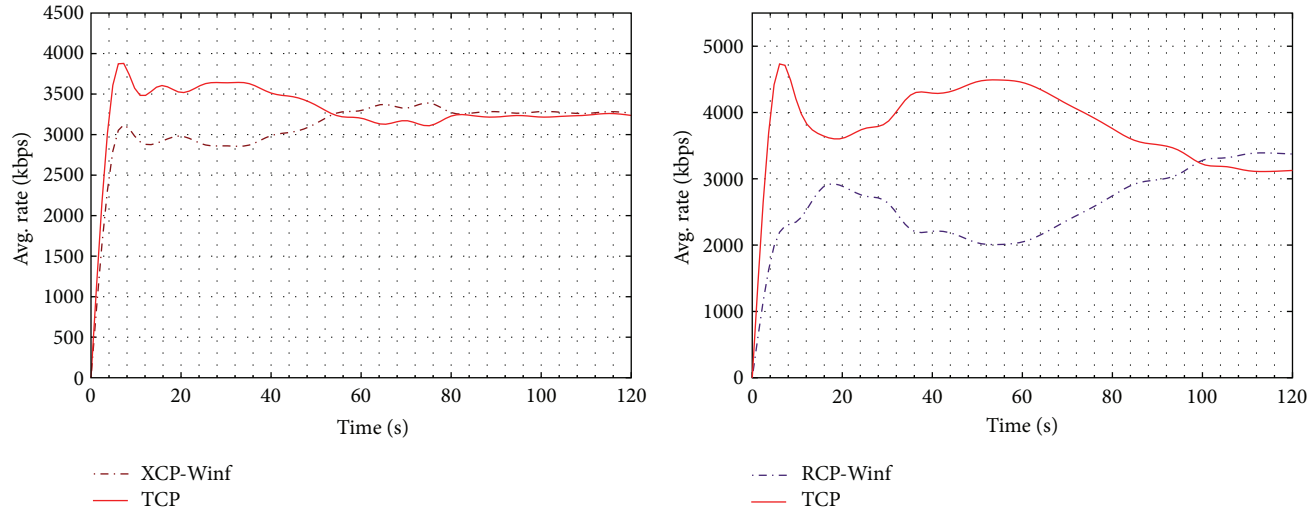
(b) Heavy flow delay

(c) Heavy flow received packets

FIGURE 14: Public building heavy flow: variable number of mobile nodes.

The evaluation results of XCP-Winf and RCP-Winf, obtained through ns-2 simulations, show that the *rt-Winf* algorithm improves significantly XCP and RCP behavior, making them more efficient and stable. To obtain the available network capacity, both XCP and RCP need all nodes in the network to cooperate, which increases network overhead, specially when dealing with special wireless environments, such as wireless mesh networks and ad hoc networks. Using *rt-Winf*, which works in the MAC layer, it is possible to perform link capacity and available bandwidth calculations without interfering in the network dynamics, allowing significant improvement of XCP and RCP performance. It was also possible to conclude that XCP-Winf and RCP-Winf behave more efficiently and use the network available information more effectively than XCP-b and TCP-AP, two protocols specifically designed for wireless environments. It is then possible to conclude that both XCP-Winf and RCP-Winf outperform XCP-b and TCP-AP in terms of medium usage, thus allowing a more stable behavior and a faster convergence to the active network conditions.

As future work, we plan to extend the evaluation with the analysis of the effect of collision probability and the improvement of the results when introducing this effect on the congestion control approaches and also to work on the improvement of XCP-Winf and RCP-Winf utility results, allowing having a much more efficient coexistence with TCP flows with the *rt-Winf* versions of XCP and RCP. An effort will also be made in optimizing other protocols behavior, such as TCP-AP, with the integration of *rt-Winf* real time and in-line information. As this work presents mainly results obtained through simulation evaluation, future work might also involve exploring the implementation of the proposed solutions against other routing protocols and also implement them directly in the Linux Kernel, allowing creating a small testbed for testing and evaluating the solutions in real environments and in different conditions.



(a) XCP-Winf utility results

(b) RCP-Winf utility results

Figure 15: Utility results.

Conflict of Interests

The author declares that there is no conflict of interests regarding the publication of this paper.

References

[1] S. Vangala and M. A. Labrador, "Performance of TCP over wireless networks with the Snoop protocol," in *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks (LCN '02)*, pp. 600–601, November 2002.

[2] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, 1997.

[3] D. Katabi, M. Handley, and C. Rohrs, "Congestion control for high bandwidth-delay product networks," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM '02)*, pp. 89–102, ACM, August 2002.

[4] N. Dukkipati and N. McKeown, "Why flow-completion time is the right metric for congestion control," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 1, pp. 59–62, 2006.

[5] L. Barreto and S. Sargento, "TCP, XCP and RCP in wireless mesh networks: an evaluation study," in *Proceedings of the 15th IEEE Symposium on Computers and Communications (ISCC '10)*, pp. 351–357, Riccione, Italy, June 2010.

[6] B. Rés, L. Barreto, and S. Sargento, "rt-Winf: real time wireless inference mechanism," in *Proceedings of the IEEE Globecom Workshop on Mobile Computing and Emerging Communication Networks (MCECN '10)*, pp. 1130–1135, Miami, Fla, USA, December 2010.

[7] H. K. Lee, V. Hall, K. H. Yum, K. Kim III, and E. J. Kim, "Bandwidth estimation in wireless lans for multimedia streaming services," *Advances in Multimedia*, vol. 2007, Article ID 70429, 7 pages, 2007.

[8] L. Barreto and S. Sargento, "XCP-winf and RCP-winf: congestion control techniques for wireless mesh networks," in *Proceedings of the IEEE International Conference on Communications (ICC '11)*, Kyoto, Japan, June 2011.

[9] CMU Wireless Emulator, http://www.cs.cmu.edu/~emulator/.

[10] F. Abrantes and M. Ricardo, "A simulation study of XCP-b performance in wireless multi-hop networks," in *Proceedings of the 3rd ACM Workshop on QoS and Security for Wireless and Mobile Networks (Q2SWinet '07)*, pp. 23–30, ACM, 2007.

[11] S. M. ElRakabawy, A. Klemm, and C. Lindemann, "TCP with adaptive pacing for multihop wireless networks," in *Proceedings of the 6th ACM International Symposium on Mobile Ad Hoc Networking and Computing (MOBIHOC '05)*, pp. 288–299, ACM, May 2005.

[12] L.-J. Chen, T. Sun, G. Yang, M. Y. Sanadidi, and M. Gerla, *AdHoc Probe: Path Capacity Probing in Wireless Ad Hoc Networks*, vol. 15, Kluwer Academic, 2009.

[13] M. Li, M. Claypool, and R. Kinicki, "WBest: a bandwidth estimation tool for IEEE 802.11 wireless networks," in *Proceedings of the 33rd IEEE Conference on Local Computer Networks (LCN '08)*, pp. 374–381, October 2008.

[14] L.-J. Chen, C.-W. Sung, H.-H. Hung, T. Sun, and C.-F. Chou, "TSProbe: a link capacity estimation tool for time-slotted wireless networks," in *Proceedings of the 4th IEEE and IFIP International Conference on Wireless and Optical Communications Networks (WOCN '07)*, pp. 1–5, July 2007.

[15] M. S. Gast, *802.11 Wireless Networks—The Definitive Guide*, O'reilly, 4th edition, 2002.

[16] J. Postel, "Transmission Control Protocol," RFC 793, 1981.

[17] B. A. Fourouzan, *TCP/IP Protocol Suite*, McGraw-Hill Higher Education, 2nd edition, 2002.

[18] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 34–39, 2001.

[19] G. Holland and N. Vaidya, "Analysis of TCP performance over mobile ad hoc networks," in *Proceedings of the 5th Annual*

ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '99), ACM, New York, NY, USA, 1999.

[20] D. Kim, C.-K. Toh, and Y. Choi, "TCP-BuS: improving TCP performance in wireless ad hoc networks," in *Proceedings of the IEEE International Conference on Communications (ICC '00)*, vol. 3, pp. 1707–1713, 2000.

[21] J. Liu and S. Singh, "ATCP: TCP for mobile ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 7, pp. 1300–1315, 2001.

[22] S. Rangwala, A. Jindal, K.-Y. Jang, K. Psounis, and R. Govindan, "Understanding congestion control in multi-hop wireless mesh networks," in *Proceedings of the 14th Annual International Conference on Mobile Computing and Networking (MobiCom '08)*, pp. 291–302, ACM, September 2008.

[23] A. Jindal and K. Psounis, "Achievable rate region and optimality of multi-hop wireless 802.11-scheduled networks," in *Proceedings of the Information Theory and Applications Workshop*, pp. 263–269, February 2008.

[24] C. L. T. Man, G. Hasegawa, and M. Murata, "ImTCP: TCP with an inline measurement mechanism for available bandwidth," *Computer Communications*, vol. 29, no. 10, pp. 1614–1626, 2006.

[25] G. Anastasi, E. Ancillotti, M. Conti, and A. Passarella, "TPA: a transport protocol for ad hoc networks," in *Proceedings of the 10th IEEE Symposium on Computers and Communications (ISCC '05)*, pp. 51–56, June 2005.

[26] C. D. A. Cordeiro, S. R. Das, and D. P. Agrawal, "COPAS: dynamic cont ention-balancing to enhance the performance of TCP over multi-hop wireless networks," in *Proceedings of the 11th International Conference on Computer Communications and Networks*, pp. 382–387, Miami, Fla, USA, October 2002.

[27] Z. Fu, H. Luo, P. Zerfos, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP performance," *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 209–221, 2005.

[28] K. Tan, F. Jiang, Q. Zhang, and X. Shen, "Congestion control in multihop wireless networks," *IEEE Transactions on Vehicular Technology*, vol. 56, no. 2, pp. 863–873, 2007.

[29] Y. Su and T. Gross, "WXCP: explicit congestion control for wireless multi-hop networks," in *Quality of Service—IWQoS 2005: Proceedings of the 13th International Workshop, IWQoS 2005, Passau, Germany, June 21–23, 2005*, vol. 3552 of *Lecture Notes in Computer Science*, pp. 313–326, Springer, Berlin, Germany, 2005.

[30] A. Sridharan and B. Krishnamachari, "Explicit and precise rate control for wireless sensor networks," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems (SenSys '09)*, pp. 29–42, ACM, November 2009.

[31] IEEE Computer Society, "IEEE Std 802.11—IEEE Standard for information technology—telecommunications and information exchange between systems—local and metropolitan area networks—specific requirements. Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications," IEEE Standard 80211-2007, 2007.

[32] Recommendation ITU-T G.711.0, 2005, http://www.itu.int/rec/T-REC-G.711/.

[33] "The network simulator—ns-2," 2001, http://www.isi.edu/nsnam/ns/.

[34] Ž. Ilić, A. Bažant, I. Čolak, and D. Jakšić, "Optimal MAC packet size in wireless LAN," in *Proceedings of the 28th International Convention MIPRO 2005: Web Services, XML and Application of XML Technology*, pp. 290–293, Croatian Association for Information and Communication Technology, Electronics and Microelectronics, Rijeka, Croatia, 2005.

[35] IPerf, https://iperf.fr/.

[36] M. Proebster, M. Scharf, and S. Hauger, "Performance comparison of router assisted congestion control protocols: XCP vs. RCP," in *Proceedings of the 2nd International Conference on Simulation Tools and Techniques (Simutools '09)*, pp. 88:1–88:8, ICST (Institute for Computer Sciences, Social-Informatics and T elecommunications Engineering), Rome, Italy, March 2009.

[37] M. Conti, G. Maselli, G. Turi, and S. Giordano, "Cross-layering in mobile ad hoc network design," *Computer*, vol. 37, no. 2, pp. 48–51, 2004.

[38] M. Fiore, trace2stats, http://perso.citi.insa-lyon.fr/mfiore/research.html.

[39] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, no. 4, pp. 234–244, 1994.

[40] J. Feng and L. Xu, "TCP-friendly CBR-like rate control," in *Proceedings of the IEEE International Conference on Network Protocols (ICNP '08)*, pp. 177–186, October 2008.