Cognitive Link Layer for Wireless Local Area Networks

Dzmitry Kliazovich[1], Jorge Lima[2], Nelson L. S. da Fonseca[2], Fabrizio Granelli[1], and Edmundo Madeira[2]

[1] DISI – University of Trento
Via Sommarive 14, I-38050 Trento, Italy
[kliazovich, granelli]@disi.unitn.it

[2] Institute of Computing
State University of Campinas
Av. A. Einstein, 1251, Campinas, Brazil
[jlima, nfonseca, edmundo]@ic.unicamp.br

Abstract—**Several attempts have been made to optimize a link layer of wireless local area networks. Most of them focus on one parameter at a time and propose optimizing approaches which rely on network state information usually not directly available at network nodes.**

In this paper, we propose a cognitive link layer optimization capable of taking large number of parameters into account and perform optimization with no explicit knowledge on the causes of the performance degradation at the end-to-end paths.

Performance evaluation results demonstrate advantages of the proposed cognitive link layer over the case when link layer parameters are fixed during runtime.[1]

Keywords-cognitive link layer, cognitive optimization

I. INTRODUCTION

802.11 Wireless Local Area Networks use the CSMA/CA (carrier sensing multiple access/collision avoidance) with exponential random backoff [1] protocol at the link layer. There have been several attempts to optimize its functionality to improve the handling of fundamental problems of the wireless medium such as hidden terminals, exposed nodes and link errors caused by signal fading or interference, and node mobility.

Tuning and Optimization: Based on the outcomes of several 802.11 MAC models [3-5], some solutions focus on contention window optimization. In fact, the choice of the contention window size creates a tradeoff between the number of active nodes, medium access delay, collision rate, and current traffic demand. Typically, low contention window sizes work well for small number of nodes when providing small medium access delay. Large contention windows are used to diversify medium access when the number of active nodes is high as well as to keep collision rate low.

The process of rate adaptation at the IEEE 802.11 MAC/PHY levels aims at selecting the most appropriate rate based on the feedback on the channel quality, and it can be either statistically-based or SNR-based. However, the performance of several of these algorithms can be limited either due to insufficient correlation between SNR and the data delivery ratio or due to the fact that rate adaptation is performed at the sender node while channel quality is assessed at the receiver [6].

Another important IEEE 802.11 link layer parameter is the retry limit. It defines the maximum number of retransmission attempts that can be taken for a data frame delivery before the link layer gives up and discards the frame. It defines the strength of the link layer ARQ mechanism which aims at compensating high error rate of the wireless link in an attempt to make it suitable for high level protocols like TCP. Several approaches analyzed retry limit configuration [3]. However, achieving optimal TCP throughput performance is often done at the expense of delay-based operational parameters such as packet delivery delay, delay jitter, medium access delay, etc.

The Request-to-Send (RTS)/Clear-to-Send (CTS) mechanism is included into the standard to cope with the hidden node problem [7]. It requires the sender to precede data frame transmission by sending a short RTS frame. This should be acknowledged by a CTS message at the receiver. Such mechanism improves performance when large data frames are used for the transmission and it should be avoided for the transmissions of small frames. Ideally, RTS/CTS threshold should be setup based on current rate of collisions caused by hidden terminals in the network. However, such metric is difficult to derive and several research proposals have tried to base their decisions on other parameters. In [8], the authors tune RTS/CTS threshold dynamically based on packet delivery ratio.

All these approaches try to optimize WLAN performance by tuning different link layer parameters. However, it appears that their optimal configuration is highly dependent on current network conditions, i.e. on the number of nodes and traffic intensity for the choice of contention window, on the channel error rate and current state for the choice of retry limit, and on the rate of collisions caused by hidden terminals for the choice of proper RTS/CTS threshold. Moreover, these parameters correspond to network state which changes dynamically and depends on traffic patterns, interfering sources and node

[1] This work has been partially supported by the Italian National Project: Wireless multiplatfOrm mimo active access netwoRks for QoS-demanding muLtimedia Delivery (WORLD), under grant number 2007R989S

[2] The authors would like to thank FAPESP for the financial support, under grant number 2007/57336-0.

978-1-4244-4388-8/09/$25.00 ©2009 IEEE

mobility in the network. Therefore, it is often difficult or sometimes impossible to maintain accurate network state information to justify proper configuration of link layer parameters.

Main contributions: In this paper we propose a framework and study cognitive algorithms for the task of optimal link layer configuration. These algorithms do not rely on network state information. Instead, they constantly perform monitoring of data flow performance over time and adjust the configuration of protocol parameters to fit optimal operation without emphasizing the potential causes influencing selected performance metrics.

Specifically, we focus on three link layer configuration parameters: contention window, retry limit, and RTS/CTS threshold, and demonstrate that the use of cognitive adaptation allows maintaining their configuration close to the optimal value under different network conditions.

Paper organization: Section II presents the core of the proposed approach by, first, describing general rules for the proposed cognitive adaptation, and then providing details on cognitive tuning of selected link layer parameters. Section III presents performance evaluation results obtained using simulations in several network scenarios. Section IV concludes the paper outlining directions for future research on the topic.

II. COGNITIVE LINK LAYER

Cognitive adaptation has recently emerged as a promising way of network evolution towards self-aware autonomous adaptation, i.e. networking that is aimed at flexible and efficient network setup and runtime reconfiguration to cope with constantly changing network conditions.

In this section, we propose an application of cognitive algorithm for tuning WLAN link layer parameters with the purpose of improving performance. The core idea of the approach is derived from the CogProt approach in [9].

A. CogProt Background

CogProt framework aims at enabling cognitive adaptation of protocols running inside a single network node as well as between different nodes of the network. It introduces a cognitive plane parallel to the protocol layers capable of monitoring protocol parameters as well as controlling them by issuing configuration commands. This cognitive plane monitors local and end-to-end performance of communication protocols using well-defined quality metrics, as well as by the use of a quality feedback loop.

The obtained performance measurements are stored in a local database along with protocol configuration parameters they were obtained with. Then, the engine performing the cognitive optimization makes a decision to towards optimal setup of the protocol operation. In different scenarios cognitive optimization can be constrained i) by a single layer, ii) by a network node (between different layers), or iii) by a network segment (between network nodes). In the later case, implementation of cognitive interactions can be either distributed, centralized, or hybrid. For the sake of simplicity and scenario considered in this paper we will study cognitive optimization of multiple parameters constrained by the link layer of a single network node only.

For each protocol parameter, CogProt framework selects minimum (P_{min}), maximum (P_{max}), and default (P_{def}) values. Whenever a new value for the parameter is needed it is obtained from a random number generator which assigns values in the [P_{min}, P_{max}] interval, according to a normal distribution with mean P_{def} (Fig. 1). As a result, most of the time P_{def} is selected by the distribution. P_{def} is assumed to be the value leading to the optimal performance. Nevertheless, neighboring to P_{def} values are also checked from time to time, and if another value (P_{new}) leads to better performance, the mean of the normal distribution of shifted from P_{def} to P_{new}.

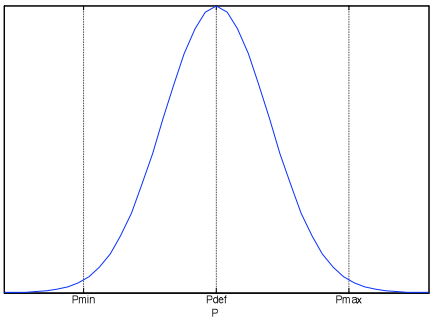

Figure 1. Sample distribution for parameters setting in CogProt.

B. Cognitive Link Layer

The Cognitive Link Layer (CLL) proposed for WLANs is based on the CogProt framework and it performs optimization of the following IEEE 802.11 link layer parameters: retry limit *(retr)*, contention window *(cw)*, and RTS/CTS threshold *(rthr)*.

Table I summarizes P_{min}, P_{max}, and P_{def} values for the considered parameters. Parameter *retr* specifies the maximum number of retransmission attempts taken by the link layer until the frame is silently dropped. When *retr* value is equal to zero the link layer never retransmits a packet. Parameter *cw* defines the initial congestion window interval for the exponential backoff procedure defined by the IEEE 802.11 standard. The parameter *rthr* defines the minimum frame size for which the RTS/CTS frames precede the data frame transmission.

TABLE I. OPERATING INTERVALS AND DEFAULT VALUES

Parameter	**Parameter Values**		
	Pmin	***Pmax***	***Pdef***
Retry limit (retr)	1	7	4
Contention window (cw)	8	64	31
RTS/CTS threshold (rthr)	0	1800	500

At the end of each sampling interval CLL performs the following actions:

Performance Monitoring computes a metric which optimizes the MAC-layer throughput with a given medium access delay bound. In order to do so, the algorithm calculates the number of bytes received within the last sampling interval

and it computes the average throughput value. Before storing it, in a local database the Exponentially Weighted Moving Average (EWMA) is computed:

$$P_i = P_c \cdot w - (1 - w) \cdot P_{i-1}. \quad (1)$$

where P_c denotes the current measurement sample, P_{i-1} corresponds to the last value of EWMA, and w is associated with a weight distribution between current and previous values.

In a similar way, medium access delay is measured as a difference between the time packet is removed from the outgoing queue and the moment its physical level transmission initiated.

Performance optimization follows performance monitoring phase. The obtained performance monitoring results are analyzed and optimal configuration for operational parameters is selected. At the system startup, when performance measurements are just started to be collected CLL configuration decisions may not correspond to optimal setups due to a short track of performance experience. However, as the time passes and more performance measurement values become available the system enters the steady state and CLL decisions converge to optimal values.

System configuration phase aims at selecting new system configuration parameters which are taken from a normal distribution set according to results of the performance optimization phase.

Algorithm 1: Performance monitoring

1. **Get** number of bytes received since last time interrupt nBytes
2. **Get** time elapsed from the last timer interrupt sampleInterval
3. **Calculate** average throughput R_i as nBytes/sampleInterval
4. **Calculate** weighted throughput $R_i = R_{i-1} \times (p) + R_i \times (1-p)$, where p is a weight given to history significance
5. **Get** average medium access delay D_i
6. **Calculate** weighted delay $D_i = D_{i-1} \times (p) + D_i \times (1-p)$, where p is a weight given to history significance
7. **Store** weighted Ri and Di values in four-dimensional array on a position defined by current retr, cw, and rthr parameter values.

Algorithm 2: Performance optimization

8. **Set** max throughput Rmax to zero
9. **For** each element of four-dimensional array with Ri and Di **do**
10. **If** current Ri is greater than Rmax **then**
11. **If** current Di is within delay bound **then**
12. **Set** $retr_{optimal}$ equal to *retr*
13. **Set** $cw_{optimal}$ equal to *cw*
14. Set $rthr_{optimal}$ equal to *rthr*
15. **Endif**
16. **Endif**
17. **Endfor**

Algorithm 3: System Configuration

18. **Set** Normal Distribution Mean to $retr_{optimal}$
19. **Get** new *retr* from the distribution
20. **Set** Normal Distribution Mean to $cw_{optimal}$
21. **Get** new *cw* from the distribution
22. **Set** Normal Distribution Mean to $rthr_{optimal}$
23. **Get** new *rthr* from the distribution

III. PERFORMANCE EVALUATION

In order to evaluate the proposed approach, we implemented the CogProt framework in the Network Simulator (ns-2) [10].

A. Cognitive engine implementation

The whole cognitive engine implementation was coded in a Mac802_11 class (mac-802_11.h, mac-802.11.cc) of the ns-2 simulator. Specific variables and methods were added to accommodate functionalities of performance monitoring, performance optimization, and values update phases.

Two four-dimensional arrays are allocated to track the MAC-level throughput and the medium access delay performances at the end of each sampling interval.

In order to calculate the medium access delay the packet header structure is extended with a timestamp field filled every time the packet is sent to the link layer for the transmission. Having this timestamp, the medium access delay can be easily calculated at the moment the transmission initiated. As a result, the medium access delay accounts for the node waiting as well as for any pending transmissions, exponential backoff, collisions, and RTS/CTS frames exchanges.

In the system configuration phase ns-2 allocates one standalone random generator initialized with a different seed for each configuration value to be updated. This preserves propagation of an error to the distribution density from one configuration parameter to another.

B. Configuration parameters

In this work we consider the link layer defined by IEEE 802.11 standard and, specifically, its Distributed Coordination Function (DCF). The following three link layer parameters are chosen for the cognitive optimization as those having greater influence on the link layer performance: retry limit *(retr)*, contention window *(cw)*, and RTS/CTS threshold *(rthr)*.

The main optimization metric is composed of two components, MAC-layer throughput and medium access delay, to maximize the throughput performance while keeping the medium access delay bounded.

The parameter *retr* limits the retransmission counter associated to each packet transmission. Whenever this counter

reaches zero the packet is dropped with no further notification to upper layers. The parameter *retr* influences the Bit Error Rate (BER)/Packet Error Rate (PER) delivered to upper layers which should be within constraints required by higher layer protocols since it affects end-to-end packet delivery delay. Low *retr* values do not change the packet delay but they are not able to compensate the high channel error rates, while high *retr* values are appropriate for noisy channels at the expense of packet delay increase.

The parameter *cw* controls the size of the initial contention window selected by the exponential backoff mechanism. This way, whenever a station begins backing off, it selects a random slot in the interval [0, *cw* – 1]. Then, for each unsuccessful transmission the value of *cw* is doubled. The size of the initial *cw* value should be driven by the number of stations contending for medium access to avoid unnecessary collisions. Generally, high *cw* values are recommended for dense networks where the number of collisions can be high. However, the *cw* value should be lower in sparse networks to avoid unnecessary medium access delay increase.

The parameter *rthr* defines the minimum size of the MAC data frame used for RTS/CTS packet exchange decreasing the vulnerability to collisions due to hidden node problems. There are several network conditions influencing a proper choice of the *rthr* value. RTS/CTS exchange is useful when the number of hidden nodes is high. However, collisions due to hidden nodes cannot be easily distinguished from other types of collisions. Moreover, with the same number and intensity of traffic generated by hidden nodes *rthr* is determined by the number of regular network nodes since total time for packet transmission which includes RTS/CTS exchange and data frame becomes longer raising a probability of collision.

C. Scenario Description

The simulated network topology is presented in Fig. 2. Wireless technology is implemented in the access while the rest of the packet transport is performed by the wired network core. Wired traffic source node S is connected to an access point AP that bridges the wired and wireless parts using two IP routers R1 and R2. The link between the source node and the first router S-R1 is 100 Mb/s, 1 ms, link between routers R1-R2 is 100 Mb/s, 10 ms, and the link connecting the last router to the access point R2-AP is 10Mb/s, 340 ms.

The wireless network is configured following the IEEE 802.11b specification with a radio link data rate of 11Mb/s, basic data rate of 1 Mb/s, and two-ray ground chosen for link propagation model.

The number of mobile nodes varied from 1 to 50 with only one traffic flow associated per node. The mobile nodes are positioned to be uniformly distributed within the transmission range of AP. Such configuration allows all mobile nodes to communicate with AP but nodes located at the opposite from AP sides become hidden one from another.

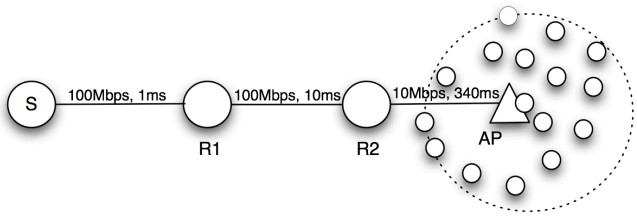

Figure 2. Simulated network topology.

The implementation of cognitive algorithm dynamically varied *retr* in the range [1;7], *cw* in the range [8;64], and *rthr* in the range [0; 1500] with increments in 300 bytes. When the *rthr* is equal to 0, the RTS/CTS threshold becomes disabled, while when the *rthr* is 1500 bytes, it is enabled for all the outgoing packets.

At the end of each sampling interval the cognitive optimization is performed. The interval duration is 100 ms. Typically, larger intervals lead to more stable behavior of the optimization algorithm and should be used in static environments. With shorter intervals the optimization algorithm can adapt to changing network conditions fast with a tradeoff in stability.

The results presented in the following sections analyze the proposed CLL approach comparing it with the default operation of WLAN network.

Standard deviation *std* equal to 0.7 and weight *w* from (1) equal to 0.5 are found to be good default values leading to expected performance of CLL approach.

D. Simulation Results: Single Parameter

First, in order to evaluate performance details and behavior of the proposed cognitive adaptation approach we simulated a scenario with only one parameter, *retr*, constrained by the proposed CLL approach. The parameter *retr* varied from 0 to 7 for simulation setups with the number of mobile nodes ranging from 1 to 50. Each mobile node established a single FTP/TCP flow originated from the source S node.

Fig. 3 presents the average medium access delay measured for the packets sent over the wireless medium. The medium access delay remains low for low *retr* values and becomes high for high *retr* values in scenarios with large number of network nodes. The CLL approach keeps medium access delay bounded with a threshold fixed at 0.004 seconds which is derived from delay budget of common VoIP and multimedia applications.

The obtained results underline applicability of the proposed approach to dynamic network scenarios: when the number of nodes is high CLL reduces *retr* to keep the delay bounded, while for sparse networks it shifts its operating point towards high *retr* values to guarantee optimal throughput performance.

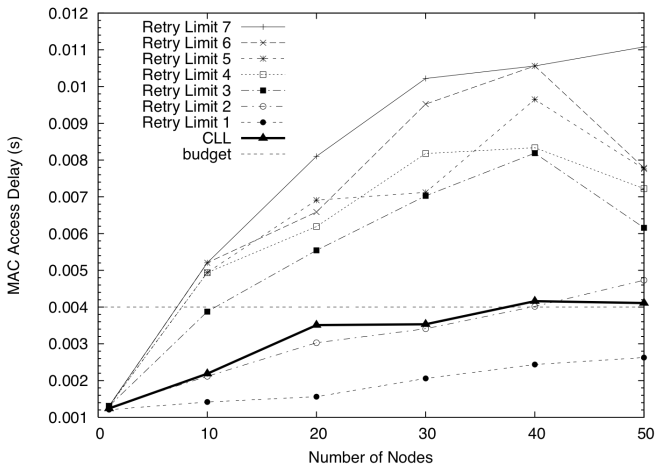

Figure 3. Average MAC access delay

Fig. 4 presents the throughput results measured at the MAC layer for different *retr* configurations. CLL shows good results with only curves corresponding to *retr* parameter equal to 5, 6, or 7 outperforming it for a large number of network nodes. However, at this interval the delay budget of these curves is out of the defined limit.

Two experiments presented above show that CLL is able to achieve maximization of the throughput performance keeping delay bounded making it a promising solution for accommodating both data transfer and multimedia applications.

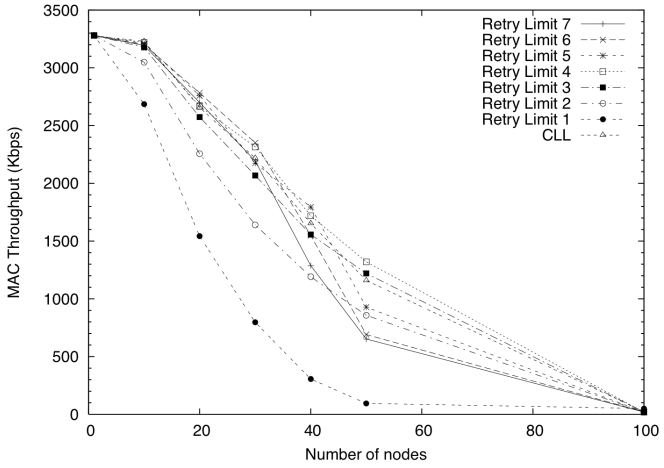

Figure 4. MAC Throughput

Fig. 5 aims at demonstration insights into CLL cognitive adaptation. It presents the number of times each *retr* value was selected by CLL for the scenario with large number of mobile nodes. As expected, optimal *retr* values which are selected for system configuration are small. However, other *retr* values are also selected by less often for having a possibility to monitor the performance and adapt when the number of mobile nodes becomes small.

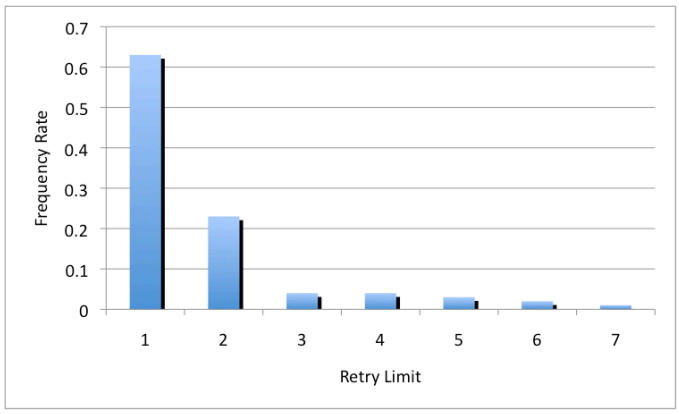

Figure 5. Frequency of *retr* value usage

E. Simulation Results: Multiple Parameters

In this section we present CLL results obtained for all three parameters considered for cognitive optimization: *retr*, *cw*, and *rthr*. In order to test CLL adaptation in a dynamic network environment the simulation starts with 1 mobile node, and proceeds with other 10 mobile nodes added every 18 seconds until the total number of nodes reaches 50.

To test the CLL mechanism with all enabled parameters, we varied the number of active nodes and the transmitted packet size over time. The simulation starts with 1 mobile node network and every 18 seconds, 10 nodes are added to the network up to 50 nodes. Then, after 100 seconds of simulation time packet size is started to be incremented by 330 bytes every 18 seconds to up 210 seconds, when simulation ends. The medium access delay threshold for CLL is set to 0.03 seconds.

Fig. 6 confirms CLL ability to adapt to high dynamic network conditions, while satisfying delay objectives.

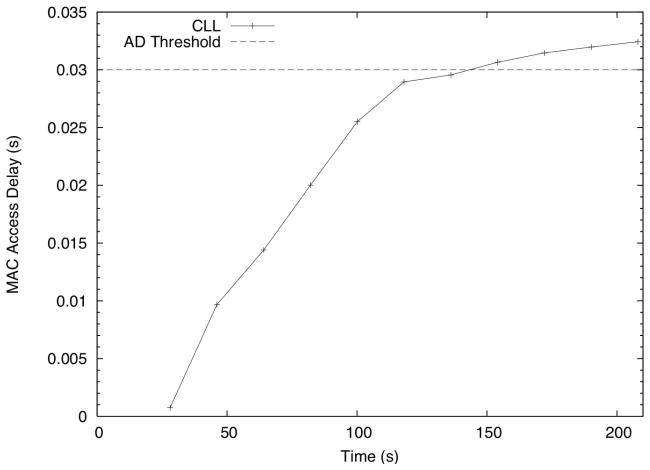

Figure 6. Average medium access delay over the time

Fig. 7 presents the measured MAC throughput over time. The throughput decreases as more nodes enter the network, due to the increase of the number of collisions.

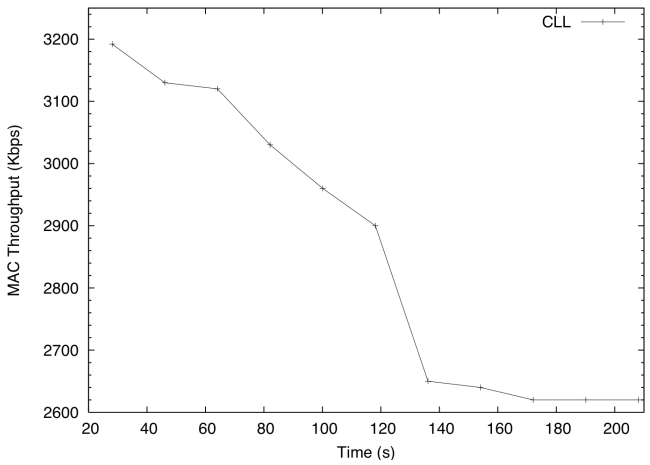

Figure 7. MAC throughput over the time

IV. CONCLUSIONS

This paper proposes a cognitive link layer optimization for wireless networks. It is opposed to traditional static setup of network configuration parameters.

The proposed approach takes into consideration a large number of link layer parameters for optimization at a time and is able to dynamically converge to the optimal setup during runtime, without having explicit knowledge about the causes of the network performance degradation.

Evaluation is performed using IEEE 802.11b layer and its control parameters, such as retry limit, contention window, and RTS/CTS threshold.

The obtained results demonstrate the advantages of the proposed cognitive link layer over the case when link layer parameters are fixed during runtime.

Future work will consider a more extensive set of simulation experiments as well as implementation of the proposed approach is hardware.

REFERENCES

[1] B. Crow, I. Widjaja, L. Kim, and P. Sakai, “IEEE 802.11 wireless local area networks,” IEEE Commun. Mag., vol. 35, pp. 116-126, Sept. 1997.

[2] Rui Jiang, V. Gupta, and C. V. Ravishankar, “Interactions between TCP and the IEEE 802.11 MAC protocol,” in Proc. Of DARPA Information Survivability Conference and Exposition, vol. 1, pp. 273 – 282, April 2003.

[3] F. Cali, M. Conti, and E. Gregori, ”Dynamic tuning of the 802.11 protocol to achieve a theoretical throughput limit”, IEEE Trans. On Networking, vol. 8, pp. 785799, Dec. 2000.

[4] S. Choudhury and J. D. Gibson, “Throughput optimization for wireless LANs in the presence of packet error rate constraints”, IEEE Communications Letters, Vol.12, No.1, pp.11-13, January 2008.

[5] Q. Xia and M. Hamdi, “Contention window adjustment for IEEE 802.11 WLANs: A control-theoretic approach”, In Proc. of IEEE ICC 2006, Vol. 9, pp.3923 - 3928, June 2006.

[6] E. Ancillotti, R. Bruno, and M. Conti, “Experimentation and Performance Evaluation of Rate Adaptation Algorithms in Wireless Mesh Networks,” 5th ACM International Symposium on Performance Evaluation of Wireless Ad Hoc, Sensor, and Ubiquitous Networks, 2008.

[7] A. Tsertou, and D. I. Laurenson, “Revisiting the Hidden Terminal Problem in a CSMA/CA Wireless Network,” IEEE Transactions on Mobile Computing, vol. 7, no. 2, pp. 817 – 831, July 2008.

[8] M. Mjidi, D. Chakraborty, N. Nakamura, K. Koide, A. Takeda, and N. Shiratori, “A New Dynamic Scheme for Efficient RTS Threshold Handling in Wireless Networks,” 22nd International Conference on Advanced Information Networking and Applications (AINA 2008), pp. 734 – 740, March 2008.

[9] D. Kliazovich, N. Malheiros, N. Fonseca, F. Granelli, R. Piesiewicz, and E. Madeira, “CogProt: A Framework for Cognitive Configuration and Optimization of Communication Protocols,” submitted to IEEE ICC, Capetown, South Africa, 2010.

[10] Ns2 Network Simulator, available at http://www.isi.edu/nsnam/ns/