Wireless Netw
DOI 10.1007/s11276-012-0536-y

CogMAC: a cognitive link layer for wireless local area networks

Jorge Lima de Oliveira Filho · Dzmitry Kliazovich · Fabrizio Granelli · Edmundo Madeira · Nelson L. S. da Fonseca

Abstract Optimization of the performance of the link layer in wireless networks is complex due to multiple parameters involved. Network management in real-time and performance adaptation are extremely challenging. In this paper, we introduce CogMAC, a cognitive link layer approach capable of tuning the network performance in highly dynamic environments. Results obtained using simulations and testbed measurements evince the superiorities of the proposed approach over existing non-adaptive techniques.

Keywords Cognitive link layer · Cognitive optimization · WLAN

J. L. de Oliveira Filho (✉) · E. Madeira · N. L. S. da Fonseca
Institute of Computing (IC), State University of Campinas, Av. A. Einstein, 1251, Campinas 13089-971, Brazil
e-mail: jlima@ic.unicamp.br

E. Madeira
e-mail: edmundo@ic.unicamp.br

N. L. S. da Fonseca
e-mail: nfonseca@ic.unicamp.br

D. Kliazovich
Interdisciplinary Centre for Security, Reliability and Trust, University of Luxembourg, 6 rue Coudenhove Kalergi, 1359 Luxembourg, Luxembourg
e-mail: dzmitry.kliazovich@uni.lu

F. Granelli
Department of Information Engineering and Computer Science (DISI), University of Trento, Via Sommarive 14, 38123 Trento, Italy
e-mail: granelli@disi.unitn.it

1 Introduction

The performance of the medium access control (MAC) protocol is usually influenced by the configuration of a large number of parameters, such as the contention window and retry limit.

Due to continuous changes in wireless environments, optimal configuration of these parameters is quite challenging given their dependence on network conditions, such as traffic intensity and channel error rate. Indeed, the state of the network changes dynamically, and it is often difficult to maintain accurate information to support proper configuration of the MAC parameters.

Different proposals have addressed different issues affecting the performance of wireless local area networks. Some proposals, for example, focus on contention window optimization [3, 4, 19] to balance the tradeoff between the number of active nodes, access delay, and collision rates. Typically, small contention windows work well for a low numbers of nodes and provide short medium access delays. Conversely, large contention windows limit the collision rate when the number of active nodes is high, but they can lead to extended medium access delays.

In addition to rate adaptation being performed at the sender node while channel quality is assessed at the receiver [1, 5], solutions aiming for optimal physical rate adaptation are often affected by an unstable correlation between the measured SNR and the data delivery ratio.

To avoid collisions caused by hidden terminals in IEEE 802.11 [10] networks, the request-to-send (RTS)/clear-to-send (CTS) threshold can be varied [18]. Ideally, the RTS/CTS threshold should be set-up as a function of the current rate of collisions caused by hidden terminals. However, this metric is difficult to derive. Therefore, the adaptability of RTS/CTS is proposed as a function of the packet

Published online: 20 January 2013

delivery ratio [14] or as a process based on the history of previous packets [9].

Moreover, the maximum number of data frame retransmission attempts can be controlled at the link layer [4] to make wireless channels suitable for TCP traffic.

Many of the existing approaches for setting up MAC parameters focus on a single parameter at a given time and rely on network state information that is often not directly available to network nodes. In this paper, we go one step further and present a mechanism for cognitive optimization of multiple link layer parameters called CogMAC. CogMAC exploits the cognitive adaptation techniques to maintain link layer performance at the optimal level during runtime. Its effectiveness is demonstrated by tuning the parameters of the CSMA-CA protocol. The notion of cognition includes the ability to observe, learn and respond to the wireless channel dynamics. Such cognition is achieved by constantly monitoring network performance, analyzing the network conditions and adjusting the link layer configuration.

The CogMAC mechanism is implemented in the ns-2 simulator [15] as well as in a Linux-based testbed. The cognitive process uses three link layer configuration parameters: contention window, retry limit, and RTS/CTS threshold. Both simulation and experimental results confirm the benefits of the proposed cognitive adaptation strategy and the ability to maintain optimal configuration of link layer parameters, even under highly dynamic network conditions.

The remainder of this paper is organized as follows: Sect. 2 presents the related works. Section 3 presents the core of the proposed approach by describing general rules for cognitive adaptation and by providing the details of the cognitive tuning of selected link layer parameters. Section 5 presents the simulation results obtained using the network simulator. Section 6 presents experiments performed on a Linux-based testbed. Finally, Sect. 7 concludes the paper by outlining directions for future research on this topic.

2 Related work

The new cognitive networks paradigm proposed [17] is motivated by the need for efficient management of the increasing complexity of communication networks. The main idea behind this new paradigm is a cognitive process that can sense the network state, plan for the future, make decisions and act accordingly [7]. The work presented in this paper uses cognition to reconfigure the parameters of the MAC layer by observing, learning and acting according to network changes. In this section, we review the most relevant cognitive network approaches.

The work proposed in [6] introduces a tool for implementing reasoning in cognitive network nodes using fuzzy cognitive maps. The main idea is to provide a methodology that enables network nodes to represent complex interactions that happen inside their protocol stacks and across the network.

The ADMA [2] is a distributed management architecture design that is used to incorporate cognition within MANETs networks. It allows network nodes to reconfigure themselves in response to environmental changes. The ADMA architecture does not require any centralized entity to perform network management functions. Each node is able to make decisions based on policies and collected information. The ADMA architecture is implemented and tested using ns-2 simulator, and improvements are achieved using VoIP applications.

The reconfigurable platform for cognitive networks proposed in [16] presents a node architecture with the ability to reason based on observations made at different layers and to adapt what is learned from these observations in a holistic manner. A cognitive cycle dynamically builds and adapts the protocol stack of a network node. The cognitive engine can be extracted from the node, a process that makes it adaptable to a variety of network technologies.

3 Cognitive link layer

CogMAC is a technique for cognitive optimization of multiple link layer parameters, such as retry limits or contention windows, in dynamic network environments. It does not rely on the explicit knowledge of network characteristics. CogMAC is implemented as a cognitive plane that functions in conjunction with the protocol stack. It interfaces with all layers of data and controls information exchange.

The CogMAC continuously infers dependencies between the achieved performance and the protocol configuration parameters. Having such information, CogMAC builds and maintains a local knowledge base that allows it to make decisions based on previous performance experience.

For each of the configured protocol parameters, CogMAC selects a minimum (*Pmin*) value, a maximum (*Pmax*) value, and a default (*Pdef*) value that defines the operational boundaries of the parameters. The value of each configuration parameter, initially fixed at (*Pdef*) is updated periodically with values randomly supplied by a normal distribution defined in the [*Pmin*, *Pmax*] interval with a mean of *Pdef* (Fig. 1). At the end of each cycle, *Pdef* is

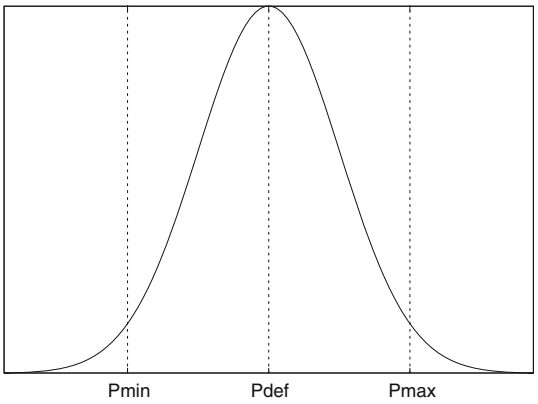

Fig. 1 Sample distribution for parameters setting in CogMAC

updated with a value leading to historically optimal performance.

This configuration allows the maintenance of system performance to function at an optimal operation point while also allowing it to test the adequacy of neighboring values and to react to dynamic changes within network environments.

For such maintenance parameters, values are sampled periodically, and at the end of each sampling interval, CogMAC performs the following actions: performance monitoring, performance optimization and system configuration, all of which are described below.

Performance monitoring counts the number of bytes received in the last sampling interval and computes the average throughput value. An Exponentially Weighted Moving Average (EWMA) value is then computed:

$$P_i = P_{i-1} * (1 - w) + P_c * w \quad (1)$$

where P_c denotes the current measurement sample, P_{i-1} corresponds to the last value of EWMA, and w is a weight given to a new sample.

Medium access delay is measured as the difference between the time a packet is removed from the outgoing queue and the moment its transmission initiates.

Performance optimization is a phase wherein the measured value is analyzed and an optimal configuration for the operational parameters is chosen. At the system startup time, CogMAC configuration decisions may not correspond to an optimal setting given the short history of collected measures. However, as time passes, additional performance measurement values become available, and convergence to optimal parameter values is achieved.

System configuration occurs when all the parameters are assigned with new values taken from a normal distribution. After that, the mean of the normal distribution is set to match the best experienced performance.

ALGORITHM ***Performance monitoring***

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

ALGORITHM ***Performance optimization***

1. Set max throughput Rmax to zero;
2. For each element of four-dimensional array with R_i and D_i
do
3. **If** current R_i is greater than Rmax **then**
4. **If** current D_i is within delay bound **then**
5. **Set** $retr_{optimal}$ equal to *retr*;
6. **Set** $cw_{optimal}$ equal to *cw*;
7. **Set** $rthr_{optimal}$ equal to *rthr*;
8. **Endif**
9. **Endif**
10. **Endfor**

ALGORITHM ***System configuration***

1. Set Normal Distribution Mean to $retr_{optimal}$;
2. Get new *retr* from the distribution;
3. Set Normal Distribution Mean to $cw_{optimal}$;
4. Get new *cw* from the distribution;
5. Set Normal Distribution Mean to $rthr_{optimal}$;
6. Get new *rthr* from the distribution.

4 Cognitive setting of 802.11 link layer parameters

In this paper, we aim to use CogMAC for the dynamic runtime configuration of IEEE 802.11 link layer parameters,

such as retry limit (*retr*), contention window (*cw*), and RTS/CTS threshold (*rthr*). The goal is to maximize the link layer throughput while ensuring the bounds of the medium access delay experienced by individual nodes. This goal is easy to achieve if we know the exact number of mobile nodes, their traffic demands and hidden nodes, and the error rates of the wireless channels. Unfortunately, this information is difficult to obtain and to maintain. However, CogMAC does not require any explicit knowledge of network parameters, and it can provide the means for performance optimization without explicit feedback from the network. In the following section, we describe CogMAC's operation in detail.

The parameter *retr* limits the retransmissions attempted at the link layer before a data frame is silently dropped, without any notification to upper layers. The parameter *retr* influences the bit and the packet error rates, BER and PER, experienced by upper layers.

Although they can compensate for low channel error rates, low *retr* values do not change the medium access delay. Instead, high *retr* values are appropriate for noisy channels at the cost of an increasing medium access delay.

The parameter *cw* defines the size of the initial contention window selected by the exponential backoff mechanism. Whenever a station backs 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 in which a large number of collisions can occur, while they should be low in sparse networks to avoid an unnecessary medium access delay increase.

The parameter *rthr* defines the minimum size of the link layer data frame for RTS/CTS packet exchange to decrease the vulnerability of collisions due to hidden nodes. However, collisions due to hidden nodes cannot be easily distinguished from other types of collisions. Moreover, under the same traffic intensity generated by hidden nodes, the values of *rthr* are determined by the number of regular network nodes.

5 Performance evaluation of CogMAC

To evaluate the proposed approach, we implement the CogMAC framework in the Network Simulator (ns-2) [15]. The next subsections provide additional details on the implementation and the results obtained.

5.1 Simulation scenario

CogMAC functionalities are coded into the wireless link layer of the ns-2 simulator (mac-802_11.h, mac-802_11 .cc).

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

To calculate the medium access delay, the packet header structure is extended with a timestamp field that is filled every time the packet is sent to the link layer for transmission. Using this timestamp, the medium access delay can easily be calculated at the moment the transmission is initiated. The obtained delay value accounts for the time that a node spends on channel contention, a process that involves waiting for the completion of all pending transmissions from other nodes.

In the system configuration phase, ns-2 allocates one standalone random generator that is initialized with a different seed for each configuration value. This is done to reduce any potential correlation between different parameters.

Figure 2 outlines the simulated network topology. The access network follows the IEEE 802.11 specification, while the rest of the network is wired. The traffic source node S, located in the wired network, is connected to an access point (AP) that bridges the wired and wireless parts. A bottleneck link with a capacity of 10 Mb/s and a propagation delay of 340 ms is placed between R2 and AP to mimic wide area network connections. The links S-R1 and R1-R2 follow the Ethernet standard, with a rate of 100 Mb/s and a propagation delay in the order of milliseconds.

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

The number of mobile nodes varies from 1 to 50, with only one traffic flow destined for each mobile node. The mobile nodes are spread uniformly within the transmission range of the AP. With this setup, all the mobile nodes can communicate with the AP, but those nodes that are located at the opposite sides of the AP are hidden from each other.

The implementation of the cognitive algorithm dynamically changes the value of *retr* in the range [1;7], *cw* in the range [8;64], and *rthr* in the range [0;1,500], with increments of 300 bytes. When the *rthr* value is equal to 0, the RTS/CTS threshold becomes disabled, and when the *rthr* value is 1,500 bytes, the RTS/CTS threshold is enabled for all the outgoing packets. *Pdef* parameters are set to the default values recommended by the IEEE 802.11 standards

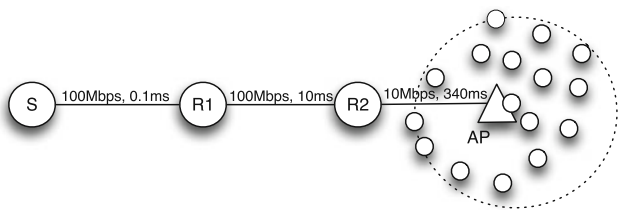

Fig. 2 Simulated network topology

Table 1 Operating intervals and default values

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

[10] and the majority of vendors. Table 1 summarizes the *Pmin*, *Pmax* and *Pdef* values used in the simulations.

The cognitive procedure is triggered periodically, every 100 ms. The length of the period controls the speed and smoothness of the cognitive tuning. Large periods lead to more stable behavior of the cognitive procedure and should be used in static environments, while shorter periods favor fast adaptation and should be used in highly dynamic network environments.

The results presented in the following sections compare the CogMAC parameter approach (parameter setting) with the operation using typical default values. A standard deviation (*std*) equal to 0.7 and a weight (*w*) equal to 0.5 were employed in these experiments.

5.2 Optimization of single parameter

To unveil the details of the proposed cognitive adaptation, we first focus on the simulation of a scenario with only one parameter, *retr*. The value of *retr* varies from 1 to 7, with the number of mobile nodes varying from 1 to 50. For each mobile node, one FTP/TCP flow that originated from the *S* node is established.

Figure 3 presents the average packet access delay. The access delay remains low for low *retr* values and increases for high *retr* values in scenarios with a large number of network nodes. CogMAC keeps the access delay bounded within a threshold fixed at 4 ms, which is a common value for the majority of VoIP and multimedia applications [8].

The results obtained underline the applicability of the proposed approach for dynamic network scenarios. When the number of nodes is high, CogMAC reduces the value *retr* to keep the delay bounded. For sparse networks, CogMAC shifts its operating point toward high *retr* values to guarantee optimal throughput.

Figure 4 presents the throughput measured at the link layer for different *retr* values. CogMAC achieves good performance for any number of nodes in the network while guaranteeing the given delay bound. There are a few cases in which other *retr* values lead to higher throughput values in Fig. 4. However, the access delay bound is violated for these cases (see Fig. 3).

The aforementioned experiments demonstrate CogMAC's ability to maximize throughput while maintaining a delay that is below the specified threshold value. This makes CogMAC a promising candidate for accommodating data transfer and multimedia application issues.

Figure 5 shows the normalized number of times each *retr* value is selected by CogMAC for a scenario with a large number of mobile nodes. As expected, small *retr* values suggest an optimal configuration as a consequence of the specified delay bound. Occasionally, near-optimal values are also selected.

5.3 Optimization of multiple parameters

In this section, CogMAC's effectiveness is evaluated when multiple parameters are adjusted by the algorithm. Specifically, three parameters are considered: *retr*, *cw* and *rthr*.

The number of active nodes and the transmitted packet size are changed. The simulation starts with only 1 mobile node. Then, at every 18 s of simulation time, 10 mobile

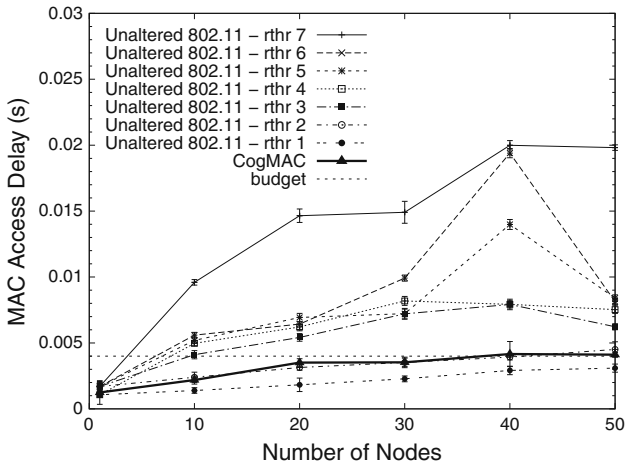

Fig. 3 Average MAC access delay

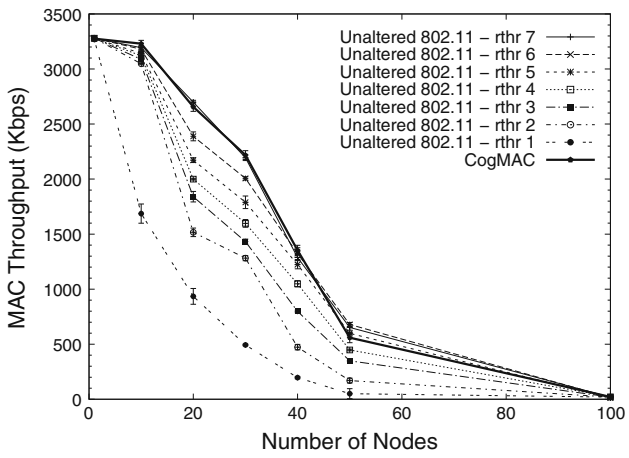

Fig. 4 MAC throughput

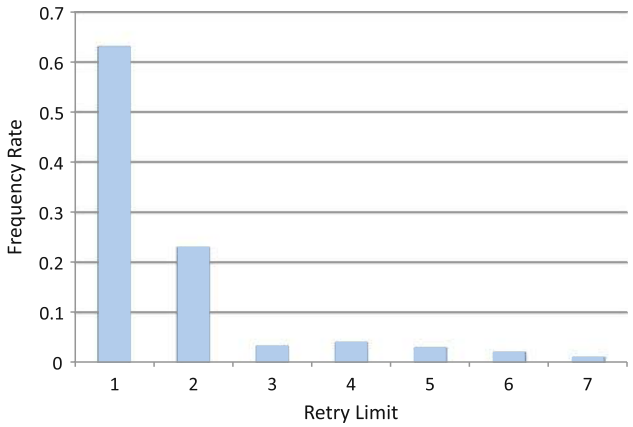

Fig. 5 Frequency of *retr* value usage

nodes enter the network (up to 50 nodes total). Then, after 100 s of simulation time, the packet size is incremented by 330 bytes every 18 s for a time period of 210 s. The medium access delay threshold for CogMAC is set to 30 ms. Varying the number of network nodes and packet sizes is performed to emphasize CogMAC's ability to adapt its operational parameters to unforeseen changes in network conditions.

Figure 6 depicts the average access delay. The access delay remains low when the number of nodes is low (25 s, 11 nodes). As the number of nodes increases, the access delay also increases and stabilizes (100 s, 50 nodes), maintaining the delay bounded. This figure confirms CogMAC's ability to adapt to highly dynamic network conditions and satisfy delay requirements. CogMAC does not guarantee a certain throughput or delay performance. Instead, it measures existing performance and tries to adjust operational parameters, which in some cases may lead to a minor violation of the desired delay threshold.

Figure 7 presents the measured throughput. The throughput decreases as more nodes enter the network, an effect that is due to an increase in the number of collisions. Despite reaching the maximum number of nodes (50 nodes, 1,250 Kbps) after 91 s of simulation time, the MAC throughput continues to decrease due to the increase of packet size.

6 Testbed experiments

6.1 Testbed structure

Figure 8 outlines the structure of the Linux-based testbed used to evaluate CogMAC's performance. There is a single cell with two laptop computers connected to a customary 802.11b/g-compliant access point. A version of the ath5k device driver is specifically modified to run the CogMAC functionalities. Laptop B is an ordinary client implementing no CogMAC functionality. Laptop B senses differents IEEE 802.11 networks, causing interference with our experiments. This scenario is representative of a residential environment with IEEE 802.11 access networks installed.

Figure 9 shows the general components of the laptop A protocol stack, including the blocks relevant to the CogMAC implementation. The kernel-level implementation ensures integrity as well as the required performance characteristics.

CogMAC implementation is divided into two parts. The first part is responsible for performance monitoring, performance optimization and adaptation. It is coded in the ath5k driver. The second part is responsible for runtime variable configuration and tuning. It is implemented in the user-space.

In OS Linux, virtual memory is divided between kernel space and user space. Kernel space is reserved for running the kernel, kernel extensions and most of the device drivers, while user space is used by user applications. A program that resides in the user space area cannot communicate with a program in the kernel space. To exchange data between the processes in the user space and

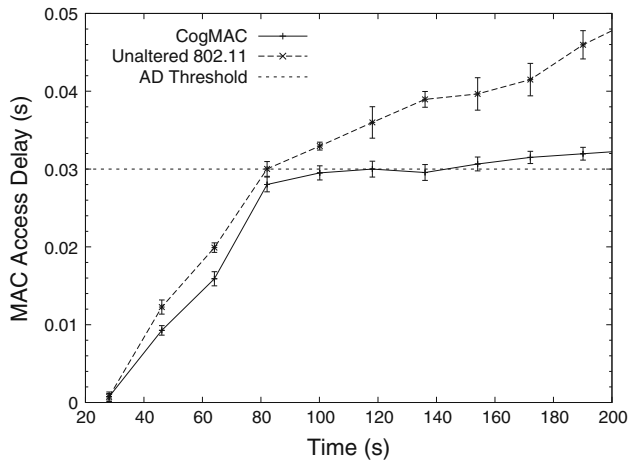

Fig. 6 Average medium access delay over the time

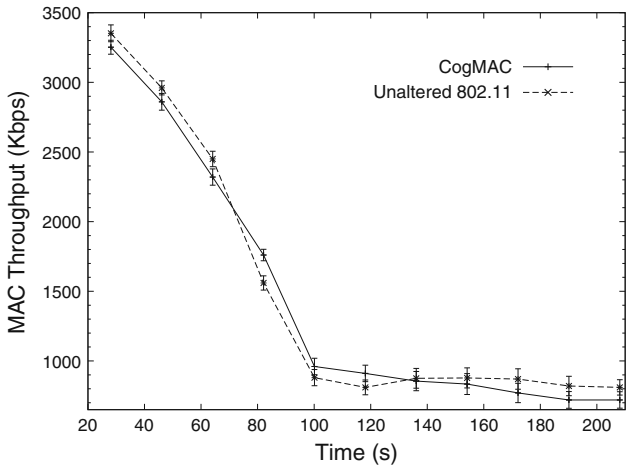

Fig. 7 MAC throughput over the time

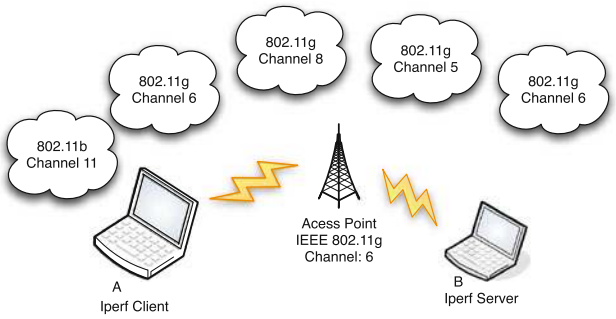

Fig. 8 Testbed network topology

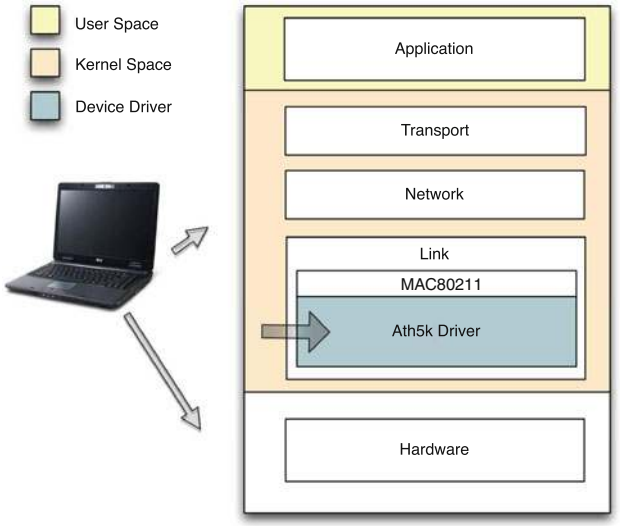

Fig. 9 Conceptual vision of the CogMAC in the Linux-based testbed

the processes in the kernel space, RAM-based file systems are implemented in the kernel.

CogMAC implementation in the form of a kernel module lifts the requirement for recompiling the entire kernel. Nevertheless, a few parameters have to be exchanged between the user and the kernel space in real time. To pass the initial configuration for the parameters *Pmin*, *Pmax* and *Pdef*, a RAM-based file system called sysctl [13] is extended. This allows applications in the user space to exchange data, such as configuration variables, with the kernel. The struct "ctl_table" is extended, and a "cogmac_table" child is set to write and read parameters.

The "cogmac_table" is populated with the parameters that need to be set-up in real time by the CogMAC mechanism in the kernel space from the user space. These parameters include the following:

- *CM_sample_interval*
- *CM_access_delay_threshold*
- *CM_retry_limit_mean*
- *CM_history_significance*
- *CM_enabled*.

Furthermore, several modifications are made in the Linux kernel source tree. The Linux SKB structure, which stores information corresponding to the representation of the packet handled by the kernel, is extended to include a timestamp value. This timestamp records the time when the packet is released by the driver to the hardware card for transmission. This information is used then for the computation of the channel access delay, which includes the time required for packet queuing, channel contention, frame transmission and possible retransmission attempts. This delay is dictated by the difference between *t1* and *t2*. The latter is stamped at the moment when the network interface card receives positive link-level acknowledgement for the transmitted frame from the receiver.

Figure 10 presents the details of the CogMAC implementation in the Linux kernel. Variables *t1* and *t2* are used as timestamps. The *t1* timestamp is recorded at the moment when the network layer forwards the allocated *skb* structure further down the MAC layer of the selected network interface. Then, after a packet enters the MAC layer, the CogMAC computes the quality metric. The aim is to optimize the MAC-layer throughput while keeping the access delay bounded. The throughput is calculated simply by counting the number of successfully transmitted bytes within a given unit of time. The delay is obtained when the device driver receives a positive link-level acknowledgement from the transmitting frame, at which moment the timestamp *t2* is recorded. This delay is obtained by the difference between *t2* and *t1*.

A five-dimensional array is allocated (6) to track the performance at the end of each sampling interval (7). Three dimensions are used to store the *retr*, *cw* and *rthr* values, while the fourth and fifth dimensions account for the throughput and MAC delay measurements.

CogMAC uses a Random Number Generator (RNG) to generate new parameter values at the end of each sampling interval. However, there is no RNG with normal distribution provided by the Linux kernel. Moreover, there are no C-library routines or float numbers supported. To overcome such limitations, we use the normal distribution expression proposed by Jain [12] to compute normal rather than uniform distribution.

The Iperf tool [11] is used for traffic generation and throughput measurements. The laptop A runs the iperf client, while the server resides in laptop B. The duration of each experiment is set to 5 minutes.

6.2 Experimental results

In the testbed experiments, special attention is given to the determination of the *retr* parameter value because it is sensitive to interference. The other two parameters, *cw* and

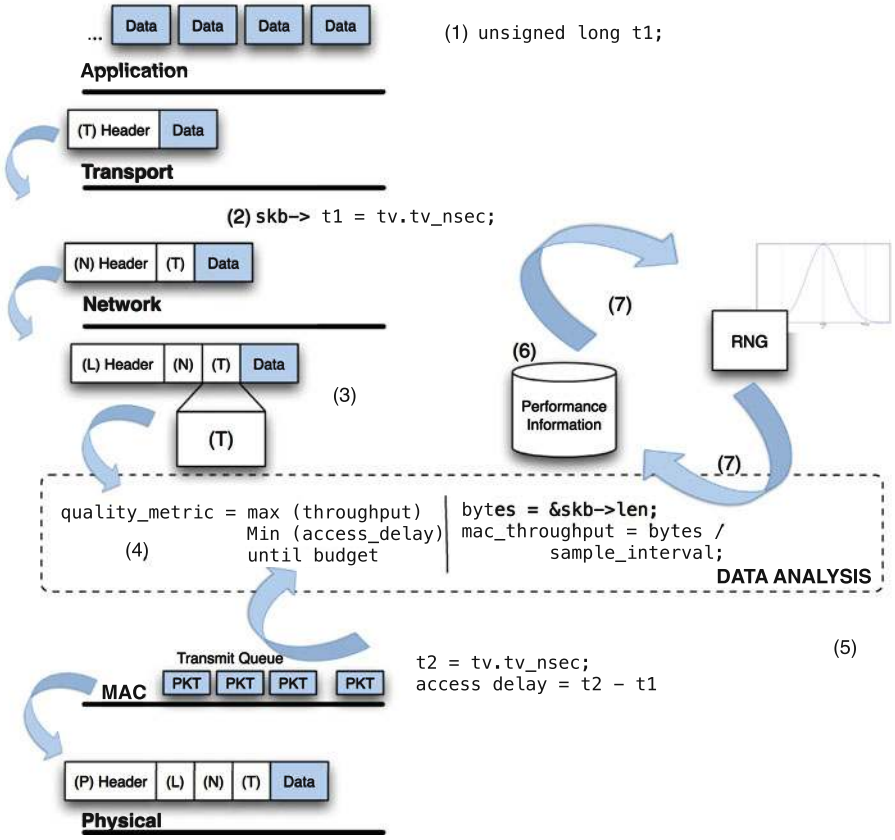

Fig. 10 Detailed scheme of the CogMAC engine in the Linux kernel divided by layers

rthr, are sensitive only to a high number of client nodes in the network.

The parameter *retr* varies from 1 to 10. Laptop A serves as the source of the TCP traffic, while the laptop B is a receiver (see Fig. 8). Each experiment is repeated 10 times, and 95% confidence intervals are reported.

CogMAC behavior is driven by the selected performance metrics and the specified delay budget. Three different budges are used: 300, 400, and 1,000 ms. Figures 11, 12, 13 present the number of times each *retr* value is selected during the experiment, while Figs. 14 and 15 provide insight into medium access delay and bandwidth variation.

For the delay budget of 300 ms, as expected, optimal *retr* values tend to be small. In Fig. 11, the maximum value is 2. The access delay value stays close to the defined bound with almost no violations (see Fig. 14); however, the throughput is low because link errors (not compensated by the small number of retransmissions) degrade TCP performance (see Fig. 15).

For a delay budget of 400 ms, the optimal value has moved from 2 to 3 with a sharp decline between 4 and 6 (see Fig. 12). The access delay value is within the defined boundaries (see Fig. 14), while the throughput increases due to the high average number of link layer retransmissions (see Fig. 15).

For a delay budget of 1,000 ms, the access delay is always below the bound value (see Fig. 14). This allows for high *retr* values (see Fig. 13), which in turn lead to a high TCP throughput (see Fig. 15).

The aforementioned results confirm CogMAC's ability to react to changing network conditions because it continuously adjusts protocol parameters to match and achieve the best setup.

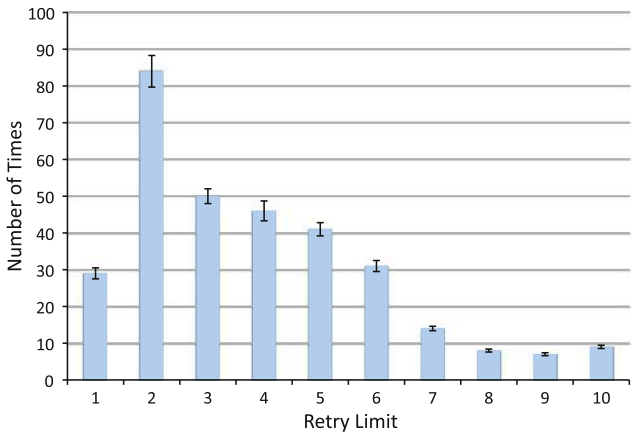

Fig. 11 Frequency of retry limit value usage with budget AD is 300 ms

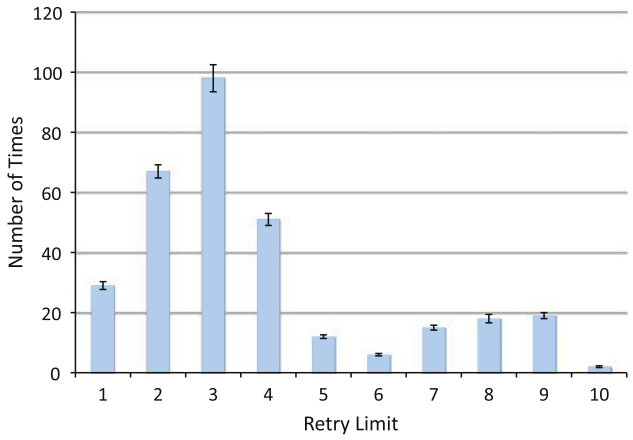

Fig. 12 Frequency of retry limit value usage with budget AD is 400 ms

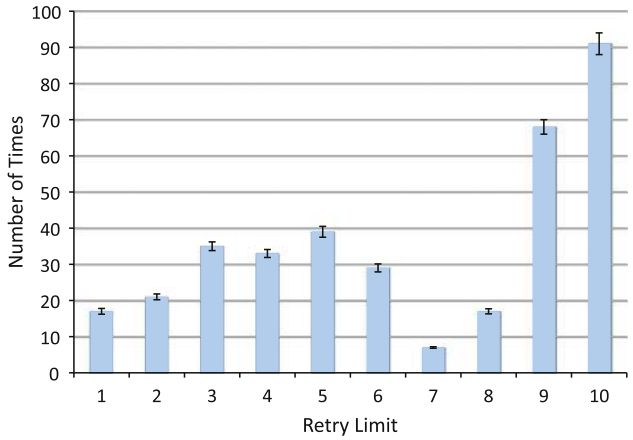

Fig. 13 Frequency of retry limit value usage with budget AD is 1,000 ms

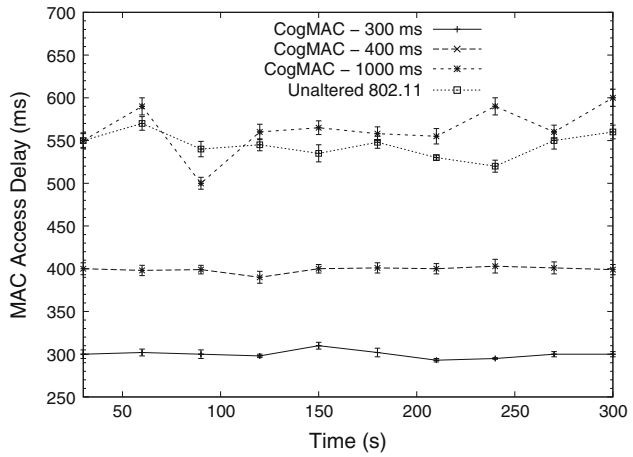

Fig. 14 Access delay

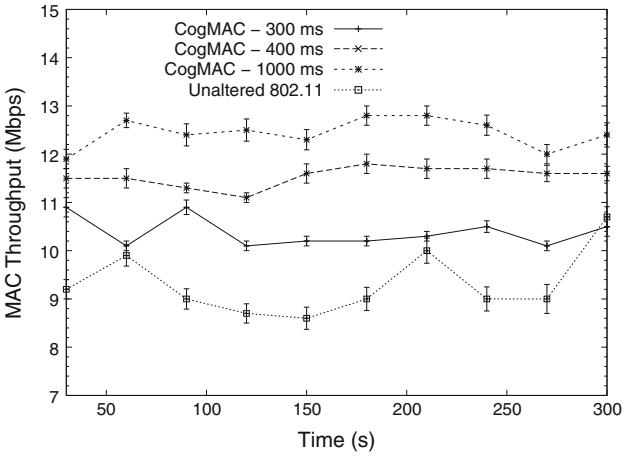

Fig. 15 Throughput with different access delay thresholds

6.3 Comparison with simulation results

Because we did not reproduce all scenarios in the testbed simulation experiments, only the results from some scenarios are compared.

Similar behavior of CogMAC in simulation and in experimentation can be observed. CogMAC can restrict the access delay in both cases (Figs. 3, 14). The lower the *retr* values, the lower the throughput that is achieved in environment with interference. This shows that CogMAC adapts the target parameter values well to this interference. For example, with the access delay for budget values of 300, 400 and 1,000 ms (see Fig. 14), CogMAC chooses *retr* values of 2, 3 and 10, respectively (see Figs. 11, 12, 13). In both the testbed (Fig. 15) and the simulation experiments (Fig. 4), low values of *retr* limit the throughput in environments with interference.

7 Conclusions

This paper proposes a cognitive approach for the optimization of link layer parameters in wireless networks. The proposed approach takes into consideration a large number of link layer parameters and it is able to calculate optimal setup values without explicit knowledge of the causes of network performance degradation.

Performance evaluation is conducted using the IEEE 802.11b link layer and its control parameters, such as the retry limit, contention window and RTS/CTS threshold, using the ns-2 simulator. Obtained results via simulation are further confirmed in the IEEE 802.11g Linux-based testbed experiments. The simulations and the Linux-based testbed experiments both unveil the advantages of the proposed cognitive link layer when compared to setups that employ fixed link layer parameters.

Future work will be focused on performing extensive comparisons with other available link layer solutions, both adaptive and non-adaptive, to truly confirm advantages of the proposed approach. Furthermore, the proposed cognitive mechanism will be adapted to allow energy saving in mobile networks.

Acknowledgments The authors would like to thank FAPESP for the financial support, under grant number 2007/57336-0 and CNPq. Furthermore, the authors would like to acknowledge the funding from National Research Fund, Luxembourg in the framework of ECO-CLOUD project (C12/IS/3977641) and Marie Curie Actions of the European Commission (FP7-COFUND).

References

1. Ancillotti, E., Bruno, R., & Conti, M. (2008). Experimentation and performance evaluation of rate adaptation algorithms in wireless mesh networks. In: *Proceedings of the 5th ACM symposium on performance evaluation of wireless ad hoc, sensor, and ubiquitous networks* (pp. 7–14). New York, NY: ACM.
2. Ayari, M., Movahedi, Z., Pujolle, G., & Kamoun, F. (2009). Adma: Autonomous decentralized management architecture for manets: A simple self-configuring case study. In: *Proceedings of the 2009 international conference on wireless communications and mobile computing: Connecting the world wirelessly* (pp. 132–137). ACM.
3. Bononi, L., Conti, M., & Gregori, E. (2004). Runtime optimization of IEEE 802.11 wireless LANs performance. *IEEE Transactions on Parallel and Distributed Systems, 15*(1), 66–80.
4. Cali, F., Conti, M., & Gregori, E. (2000). Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. *IEEE/ACM Transactions on Networking (TON), 8*(6), 799.
5. Choi, J., Na, J., Park, K., & Kim, C. (2007) Adaptive optimization of rate adaptation algorithms in multi-rate WLANs. In: *Proceeding of IEEE ICNP*. Citeseer.
6. Facchini, C., & Granelli, F. (2009). Towards a model for quantitative reasoning in cognitive nodes. In: *GLOBECOM Workshops, IEEE* (pp. 1–6). IEEE.
7. Fortuna, C., & Mohorcic, M. (2009). Trends in the development of communication networks: Cognitive networks. *Computer Networks, 53*(9), 1354–1376.
8. Goode, B. (2002). Voice over internet protocol (VoIP). *Proceedings of the IEEE, 90*(9), 1495–1517.
9. Gunes, M., Hecker, M., & Bouazizi, I. (2003). Influence of adaptive RTS/CTS retransmissions on TCP in wireless and ad-hoc networks. In: *Eighth IEEE international symposium on computers and communication (ISCC 2003)* (pp. 855–860). IEEE.
10. IEEE. (1999). Wireless lan medium access control (mac) and physical layer (phy) specifications. IEEE Standard 802.11.
11. Iperf. Available at http://iperf.sourceforge.net/.
12. Jain, R. (1991). *The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling*. New York: Wiley.
13. Love, R. (2010). *Linux kernel development* (3rd ed.). Boston, MA: Addison-Wesley. ISBN-10: 0672329468, ISBN-13: 9780 672329463 .
14. Mjidi, M., Chakraborty, D., Nakamura, N., Koide, K., Takeda, A., & Shiratori, N. (2008). A new dynamic scheme for efficient RTS threshold handling in wireless networks. In: *Proceedings of the 22nd international conference on advanced information networking and applications* (pp. 734–740). IEEE Computer Society.
15. Ns2 network simulator. Available at http://www.isi.edu/nsnam/ns/.
16. Sutton, P., Doyle, L., & Nolan, K. (2006). A reconfigurable platform for cognitive networks. In: *1st international conference on cognitive radio oriented wireless networks and communications, 2006* (pp. 1–5). Ieee.
17. Thomas, R., DaSilva, L., & MacKenzie, A. (2005). Cognitive networks. In: *New frontiers in dynamic spectrum access networks, DySPAN 2005* (pp. 352–360). IEEE.
18. Tsertou, A., & Laurenson, D. (2008). Revisiting the hidden terminal problem in a csma/ca wireless network. *IEEE Transactions on Mobile Computing, 7*(7), 817–831.
19. Xia, Q., & Hamdi, M. (2006). Contention window adjustment for IEEE 802.11 WLANs: A control-theoretic approach. In: *IEEE international conference on communications, ICC'06, 9*.

Author Biographies

Jorge Lima de Oliveira Filho (jlima@ic.unicamp.br) is an assistant professor at University of Santa Cruz (UESC), Brazil. He is a Ph.D. candidate in Computer Science at University of Campinas (UNICAMP), Brazil. His main research interests include network management and cloud computing.

Dzmitry Kliazovich is a Research Fellow at the Faculty of Science, Technology, and Communication of the University of Luxembourg. He holds an award-winning Ph.D. in Information and Telecommunication Technologies from the University of Trento, Italy. Prior to joining the University of Luxembourg he was an ERCIM Research Fellow at the VTT Technical Research Centre of Finland and a Scientific Advisor for Wireless Communications at the Create-Net Research Centre, Italy. In 2005 he was a Visiting Researcher at the Computer Science Department of the University of California at Los Angeles (UCLA). A year later he joined Nokia Siemens Networks with the responsibility of starting up a research direction focusing on 3G Long-Term Evolution (LTE). Dr. Kliazovich is a holder of several scientific awards including fellowship grants provided by the European Research Consortium for Informatics and Mathematics (ERCIM), the IEEE Communication Society, Italian Ministry of Education, and the University of Trento. His work on energy-efficient scheduling in cloud computing environments received Best Paper Award at the IEEE/ACM International Conference on Green Computing and Communications (GreenCom) in 2010. Dr. Kliazovich is the author of more than 70 research papers, Editorial Board Member of the IEEE Communications Surveys and Tutorials, Features Editor of the ICaST magazine and contributing member of the IEEE ComSoc Technical Committee on Communication Systems

Integration and Modeling (CSIM). His main research activities are in the field of energy efficient communications, cloud computing, and next-generation networking.

Fabrizio Granelli is IEEE ComSoc Distinguished Lecturer for 2012–2013, and Associate Professor at the Dept. of Information Engineering and Computer Science (DISI) of the University of Trento (Italy). From 2008, he is deputy head of the academic council in Information Engineering. He received the "Laurea" (M.Sc.) degree in Electronic Engineering and the Ph.D. in Telecommunications Engineering from the University of Genoa, Italy, in 1997 and 2001, respectively. In August 2004 and August 2010, he was visiting professor at the State University of Campinas (Brasil). He is author or co-author of more than 130 papers with topics related to networking, with focus on performance modeling, wireless communications and networks, cognitive radios and networks, green networking and smart grid communications. Dr. Granelli was guest-editor of ACM Journal on Mobile Networks and Applications, ACM Transactions on Modeling and Computer Simulation, and Hindawi Journal of Computer Systems, Networks and Communications. He is Founder and General Vice-Chair of the First International Conference on Wireless Internet (WICON'05) and General Chair of the 11th and 15th IEEE Workshop on Computer-Aided Modeling, Analysis, and Design of Communication Links and Networks (CAMAD'06 and IEEE CAMAD'10). He is TPC Co-Chair of IEEE GLOBECOM Symposium on "Communications QoS, Reliability and Performance Modeling" in the years 2007, 2008, 2009 and 2012. He was officer (Secretary 2005–2006, Vice-Chair 2007–2008, Chair 2009–2010) of the IEEE ComSoc Technical Committee on Communication Systems Integration and Modeling (CSIM), and Associate Editor of IEEE Communications Letters (2007–2011).

Edmundo Madeira (edmundo@ic.unicamp.br) is a full professor at Institute of Computing (IC) of University of Campinas (UNICAMP), Brazil. He received his Ph.D. in electrical engineering from UNICAMP in 1991. He has published over 140 papers in national and international conferences and journals. He was General Chair of the 7th Latin American Network Operation and Management Symposium (LANOMS'11), and Technical Program Co-Chair of the IEEE Latin-Cloud '12. His research interests include network management, future Internet, and cloud computing.

Nelson L. S. da Fonseca received his Ph.D. degree in Computer Engineering from The University of Southern California in 1994. He is a Full Professor at Institute of Computing of The University of Campinas, Campinas, Brazil. He has published 300+ papers and supervised 50+ graduate students. As Visiting Professor, he lectured at the University of Trento, the University of Pisa and the University of Basque Country. He is the recipient of the 2012 IEEE Communications Society (ComSoc) Joseph LoCicero Award for Exemplary Service to Publications and the 2011 ComSoc Latin America Service award. He received the Medal of the Chancelor of the University of Pisa (2007). He is also the recipient of the Elsevier Computer Network Journal Editor of Year 2001 award. He is past EiC of the IEEE Communications Surveys and Tutorials, past EiC of ComSoc Electronic Newsletter and past Editor of the Global Communications Newsletter. He is Senior Editor for the IEEE Communications Surveys and Tutorials and Senior Editor for the IEEE Communications Magazine, a member of the editorial board of Computer Networks, Peer-to-Peer Networking and Applications, Journal of Internet Services and Applications and International Journal of Communication Systems. He served in the editorial board of IEEE Transactions on Multimedia. He founded the IEEE Latin America Conference on Communications (LATINCOM) and the Latin America Conference on Cloud Computing and Communications (LATINCLOUD). He was technical chair of over 10 IEEE conferences. Nelson is an active volunteer of the IEEE Communications Society. Currently, He is ComSoc Vice Chair Member Relations. He served as: Member-at-Large in ComSoc Board of Governors, Director of Latin America Region and Director of on-line Services.