J Braz Comput Soc (2013) 19:493–510
DOI 10.1007/s13173-013-0113-y

ORIGINAL PAPER

CogTRA: a deployable mechanism for cognitive transmission rate adaptation in IEEE 802.11 networks

Luciano J. Chaves · Edmundo R. M. Madeira · Islene C. Garcia

Received: 20 May 2012 / Accepted: 27 May 2013 / Published online: 12 June 2013
© The Brazilian Computer Society 2013

Abstract Wireless local area networks have become vastly popular, and IEEE 802.11 is the chosen standard for almost all wireless devices. This standard specifies several modulation and channel coding techniques that must be implemented by all wireless interfaces to adapt to changes in wireless channels. As a result, these interfaces support multiple transmission data rates. However, this standard does not define how to dynamically select the appropriate data rate; instead, manufacturers can design and implement their own algorithms. Although several solutions have been proposed in the literature, only a few are used in practice. Moreover, their performance is still limited to specific conditions, such as highly dynamic environments. To tackle these challenges, this paper introduces CogTRA, which is a deployable mechanism built upon an existing cognitive framework called CogProt. Due to its self-adjustment functionality, CogTRA can work not only in stable but also in dynamic environments. It was implemented in the OpenWrt Linux distribution for embedded devices and evaluated through experiments using real network equipment. The results underline performance benefits with respect to existing data rate adaptation algorithms, with CogTRA exhibiting better performance especially in such dynamic networks.

Keywords IEEE 802.11 networks · Rate adaptation · Self-configuration · Cognitive algorithms

L. J. Chaves (✉) · E. R. M. Madeira · I. C. Garcia
Institute of Computing, University of Campinas (Unicamp),
Av. Albert Einstein, 1251, Campinas, SP 13083-852, Brazil
e-mail: lchaves@ic.unicamp.br

1 Introduction

Wireless local area networks have emerged as key building blocks in broadband Internet access networks, and IEEE 802.11 [1] is the chosen standard for almost all wireless devices. Among the factors that have contributed to the success of this standard, one key aspect has been the development of sophisticated modulation and channel coding techniques, which convert data streams into suitable form for transmission over wireless channels. It is possible to achieve higher data rates using techniques that efficiently take advantage of good channel conditions. However, these techniques are more sensitive to signal degradation and do not perform well either for long-range transmissions or in environments with high interference or noise. In these cases, it is appropriate to use robust solutions, which lead to resilient connections but reduce the data rate due to redundancy overhead.

To address unstable environments, a group of several modulation and channel coding techniques were defined as mandatory by this wireless standard. For instance, IEEE 802.11g supports up to 12 combinations of techniques to achieve nominal data rates. However, the standard does not specify how to select an appropriate data rate for current conditions to optimize performance. This issue is known as the transmission data rate adaptation problem.

Because the IEEE 802.11 standard offers no support for choosing the transmission data rate, the general approach used by existing rate adaptation mechanisms consists in assessing link conditions first and then selecting the most appropriate rate based on link quality estimation [3]. For this, information must be gathered at the sender (and possibly also at the receiver) to evaluate it in accordance with the goals of the algorithm. The most common indicators include throughput of probe frames sent at different rates, the number of consecutive successes or failures in frame transmissions,

the frame error rate (FER), and signal quality indicators like signal-to-noise ratio (SNR).

The absence of a standard solution has motivated the development of many rate adaptation algorithms, some of which are used on off-the-shelf devices [4,20,25,36]. However, they still present limitations. Mechanisms based on signal quality indicators suffer from a lack of a strong correlation between the indicator and the delivery probability at a given data rate. In addition, data rate configuration is performed at the sender. Meanwhile, the signal information is available at the receiver, which leads to communication overhead [12]. Moreover, the use of statistics like throughput of probing frames or FER is affected by the difficulty of finding proper thresholds for optimal rate selection. Additionally, such mechanisms present long convergence time, which leads to performance degradation in dynamic environments [39].

An effective rate adaptation algorithm should be able to dynamically find the proper data rate in both stable and dynamic environments. In the former, the algorithm is especially useful at initial set-up, as it helps to identify which rate should be used during network operation. In the latter, the signal quality varies over time, and therefore, the algorithm must suit the environment and dynamically adjust the data rate to match current link conditions. Among existing solutions, there is a cognitive mechanism called CORA [6], which implements a cognitive self-configuration functionality to adapt the data rate in accordance to the CogProt framework [28]. However, CORA is not ready to be deployed in embedded devices, and it only performs well in stable environments with low channel fluctuations. Another important solution is Minstrel [36], which is a high-performance mechanism mainly used in devices with open-source systems. Minstrel presents better results in dynamic environments, but its performance is still poorly explored in the current literature.

To address some of the aforementioned issues, as well as to increase the performance of previous solutions, this paper introduces a cognitive transmission rate adaptation algorithm (CogTRA). The first main contribution of this paper is the development of a novel cognitive data rate adaptation mechanism, which is built upon the CogProt framework but enhanced with cutting-edge features specifically designed to make CogTRA a deployable solution (this involves lower computational resource usage and the self-adjustment of configurable parameters to match both static and dynamic environments). The second main contribution is the implementation of this mechanism into the OpenWrt Linux distribution for embedded devices equipped with Atheros wireless chipsets [29]; and a comparison of the performance evaluation of this algorithm with the results from the Minstrel and other typical rate adaptation mechanisms.

Because CogTRA is a deployable mechanism, it can be implemented in a straightforward way and put to use on existing wireless embedded devices. It requires no changes on previously established standards. Additionally, there are advantages regarding computational resource usage. CogTRA reduces frame loss by quickly reacting to changes in channel conditions, and it achieves long-term stability when there are no link quality fluctuations. In this regard, CogTRA self-adjusts some of its configurable parameters to suit the environment. Relatedly, Minstrel is one of the most important existing algorithms, and its performance is significant compared to other solutions. To the best of the authors knowledge, [40,41] are the only works that evaluate Minstrel's performance in detail. Thus, this paper also compares CogTRA and Minstrel, highlighting the benefits of both approaches.

The rest of this paper is organized as follows. Section 2 presents the rate adaptation problem. Section 3 summarizes related work, including the CogProt framework and the cognitive rate adaptation (CORA) algorithm. Section 4 details the CogTRA mechanism, and it presents the set of improvements offered by this novel solution. Each improvement is discussed separately, and the complete algorithm is described at the end. Section 5 provides a performance evaluation and comparison of CogTRA using the typical transmission rate adaptation mechanisms. Finally, Sect. 6 presents the conclusion of this work.

2 The rate adaptation problem

The IEEE 802.11 standard specifies support for multiple transmission data rates at the physical layer so that networks can perform properly in environments with different levels of interference and noise. Each data rate is derived from a coding algorithm for error correction, a digital modulation process, and a spread spectrum technique. Table 1 summarizes the techniques used in 802.11abg networks.

Each data rate is calculated using some form of forward error correction with a coding rate expressed by k/n, where n coded bits are transmitted for every k bits of data. The coded bits are translated into symbols through the digital modulation process, and these symbols are transmitted over the radio interface using a spread spectrum technique. Each nominal data rate is found by multiplying the coding rate, bits per symbol, and the number of symbols per second. However, the nominal data rate cannot be achieved in practice due to protocol overhead.

Higher data rates demand high-quality signals to perform proper demodulation with a small bit error rate (BER). This is a necessary condition for bit error correction when using high coding rates. Because the signal quality is not always good and because the distance between nodes can increase, it may be necessary to use lower rates to maintain the connection alive and, therefore, recover more bit errors successfully. These rates are considered to be robust because they are

Table 1 A summary of the IEEE 802.11 data rates [4]

Rate (Mbps)	IEEE 802.11 standards	Spread spectrum	Digital modulation	Coding rate	Bits per symbol	Symbols per second
1	b/g	DSSS-BK	BPSK	1/11	1	11×10^6
2	b/g	DSSS-BK	QPSK	1/11	2	11×10^6
5.5	b/g	DSSS-CCK	QPSK	4/8	1	11×10^6
11	b/g	DSSS-CCK	QPSK	4/8	2	11×10^6
6	a/g	OFDM	BPSK	1/2	1	12×10^6
9	a/g	OFDM	BPSK	3/4	1	12×10^6
12	a/g	OFDM	QPSK	1/2	2	12×10^6
18	a/g	OFDM	QPSK	3/4	2	12×10^6
24	a/g	OFDM	QAM-16	1/2	4	12×10^6
36	a/g	OFDM	QAM-16	3/4	4	12×10^6
48	a/g	OFDM	QAM-64	2/3	6	12×10^6
54	a/g	OFDM	QAM-64	3/4	6	12×10^6

immune to small variations in channel conditions. Because of that, it is possible to increase the coverage area and perform well not only for long-range transmission but also in environments with high levels of interference [14].

Appropriate rate selection is essential for optimal network utilization [15,18,22]. According to [18], for a given link condition, there exists a rate that maximizes performance (usually in terms of throughput). If a higher rate is used, then the throughput may be reduced due to retransmissions, as BER generally increases along with the rate for a constant SNR. This is only a general trend that has exceptions: the 9 Mbps data rate has a higher BER than 12 Mbps in almost every case, as noted by [14,25]. However, the use of a lower rate limits the performance because transmissions will require more time to complete. Thus, an effective rate adaptation algorithm should be able to dynamically find the optimal data rate by itself in both stable and dynamic environments. In the former, the algorithm can be useful at initial set-up, identifying which rate should be used during network operations. In the latter, where the signal quality varies over time, the algorithm must dynamically adjust the data rate to match current link conditions. In this case, the goal is to boost performance when possible and to always sustain the connection between network nodes alive when link quality decreases.

A relevant problem that arises when using multiple data rates is the performance anomaly effect, which was first investigated in [13]. This problem occurs because the carrier sense multiple access with collision avoidance (CSMA/CA) protocol, which is used at medium access control (MAC) layer, only guarantees that the long-term channel access probability is equal for all nodes. However, it does not restrict the amount of time during which a node can reserve the channel. Because of that, when a node using a lower data rate captures the channel, it takes too long to release it, thereby penalizing all other nodes that use a higher rate during smaller periods of time. According to [13], if there is a node using a lower rate, than the throughput of other nodes transmitting at higher rates can be degraded roughly to level of this lower rate in specific conditions, such as in highly congested environments. This anomaly should be considered in the design of rate adaptation algorithms to avoid transmission at lower data rates when it is not necessary [27].

3 Related work

A fundamental problem in wireless networks is to design communication protocols capable of achieving high throughput in the face of noise, interference, and fading, all of which vary with time. An ideal promising solution to this problem is the implementation of rateless wireless systems [10]. In a rateless network, the sender encodes data without any explicit estimation or adaptation, implicitly adapting to the vagaries introduced by noise, interference, or fading. The receiver processes the symbols as it arrives, decoding it, until the message is received successfully. In this case, the network can perform well without incurring the complexities and challenges of implementing multiple fixed data rate together with a rate adaptation algorithm to select between them. Existing rateless coding schemes, such as Strider [11] and spinal codes [31], allow near Shannon capacity [35], and improved communication performance when compared to state of the art rate adaptation mechanisms.

However, these rateless solutions cannot be implemented in existing off-the-shelf hardware or without modifications in the 802.11 standard. For this reason, there are still efforts in the design and implementation of more efficient rate

adaptation algorithms to proper handle multiple data rates. In Sect. 3.1, we highlight some of the most relevant solutions to the rate adaptation problem. Specifically, because our proposed cognitive mechanism CogTRA was built based upon the CogProt framework, Sect. 3.2 presents CogProt, and Sect. 3.3 details CORA, which is the plain CogProt implementation for the rate adaptation problem.

3.1 Existing rate adaptation algorithms

The rate adaptation process involves two steps: first, assess wireless link conditions, and then select the most appropriate rate based on quality information. Existing algorithms can be classified according to the approach used to assess link conditions [2,3], including (1) history-based algorithms, which use information from previous transmissions to infer future conditions; (2) signal-strength-based algorithms, which rely on signal quality measurements as link indicators; and (3) hybrid algorithms, which combine the first two categories. The major rate adaptation solutions are presented as follows.

One of the early history-based solutions was the auto rate fallback (ARF) algorithm [20]. ARF defines fixed thresholds to increase or decrease the data rate according to the number of successes or failures on consecutive transmission attempts, respectively. It is a simple solution that attempts to use the highest effective data rate at each moment. However, it suffers from instability, as it tries to increase the rate even when it reaches the optimal one. Adaptive auto rate fallback (AARF) [25] was proposed as an improvement over ARF. It uses a binary exponential back off mechanism to dynamically adjust increase and decrease thresholds to mitigate instabilities resulting from unnecessary changes on the data rate.

The authors of AARF also proposed the adaptive multi-rate retry (AMRR) algorithm [25]. AMRR is based on the same ideas as AARF, although it is designed to work in high latency systems like those using Atheros chipsets. AMRR uses a set of four pairs of rate and transmission counts (r_0/c_0, r_1/c_1, r_2/c_2, r_3/c_3) per frame. Originally the frame is transmitted with the rate r_0. If this transmission fails (as determined by the absence of an ACK response), the hardware makes $c_0 - 1$ attempts to retransmit the frame. If the transmission is still unsuccessful, AMRR makes c_1 attempts with the rate r_1; then c_2 attempts with rate r_2; and, finally, c_3 attempts with rate r_3. When the transmission fails $c_0 + c_1 + c_2 + c_3$ times, the frame is discarded. In AMRR, the first rate r_0 is selected using the AARF algorithm's principle. Then, rates r_1 and r_2 are set to the next lower available rates, following a decreasing order. Finally, rate r_3 is always set to the lowest available rate. Counters c_0 to c_3 are always set to 1. This approach is called multi-rate retry (MRR) and is used to handle short-term channel variations. In fact, MRR can almost serve as patch to the limitation imposed by the implementation of the rate adaptation as part of the wireless interface driver. Due to the fairly short timing issue facing frame retransmissions at the MAC layer, the rate adaptation mechanisms implemented in the driver are not able to react in a per frame basis. So, MRR is a workaround for this problem.

All theses solutions provide acceptable performance on scenarios with a minimum number of clients and access points (APs). However, while they adapt the data rate based on frame acknowledgment, they are unable to identify the cause of frame loss. Because of that, in an environment with APs and clients transmitting on the same channel, there is a high probability of experiencing a collision if two simultaneously transmitted frames interfere at the receiver. To avoid this, the IEEE 802.11 CSMA/CA employs the carrier sense, a random back off, and the RTS/CTS mechanisms. Nevertheless, in hidden-terminal or congested environments, collisions will occur anyway. This increase in collision-based packet errors leads the ARF, AARF, and AMRR algorithms to unnecessarily decrease the rate as observed in [33].

To overcome these difficulties, the collision-aware rate adaptation (CARA) algorithm [21] was developed. It is similar to ARF, but it uses the RTS/CTS mechanism to identify the cause of frame loss. When the collision is the cause, the algorithm can prevent unnecessary rate decreases. This approach reduces misleading link quality information due to collisions, but it introduces overhead and can lead to instability by alternating between using or not using the RTS/CTS control frames.

Another history-based solution is the SampleRate algorithm [4]. It periodically sends single frames at rates other than the current one to estimate the expected transmission time per frame at that rate. Then, it adjusts the rate to the best-known one. The limitations include the small number of probing packets, which can be misleading and trigger incorrect rate changes.

Using a similar principle, there is the Minstrel [36] algorithm, which aims to increase the throughput based on larger sample performance measurements. As with AMRR, Minstrel uses MRR with four rate-count pairs. The rate order is based on the measured throughput and probability of success for each rate. These metrics are recalculated every 100 ms, and Minstrel uses 10 % of frames to randomly try other rates to collect statistics. Therefore, frames can be transmitted during normal or "look around" phases. Table 2 details

Table 2 Minstrel multi-rate retry chain table

Rate	Normal	Look around	
		Random < best	Random > best
r_0	Best tp.	Best tp.	Random rate
r_1	2nd best tp.	Random rate	Best tp.
r_2	Best prob.	Best prob.	Best prob.
r_3	Lowest rate	Lowest rate	Lowest rate

the Minstrel MRR table. If a frame is to be transmitted in the normal phase (as with 90 % of frames), then the MRR retry chain table r_0 is the best throughput; r_1 is the next best throughput; r_2 is the best probability; and r_3 is the lowest data rate. During the "look around" phase (as with 10 % of frames), if the randomly selected rate is slower than the current rate with the best throughput (random < best column), then it is placed second in the chain; otherwise, it is placed first. Thus, slower random rates are only sampled during this "look around" phase when the first frame transmission attempts fail. Consequently, if the link is ideal, all frames will be sent at the best rate. The retry counters (c_0 to c_3) are adjusted to ensure the packet is sent or dropped in at most 26 ms, to avoid delaying the next data packet due to a TCP congestion control mechanism.

All history-based algorithms presented so far rely on prior statistical information to adapt the rate. When using a throughput-driven history-based algorithm (like SampleRate and Minstrel), it is expected that the collision probability remains independent of the rate choice and that the collisions should cancel each other out when comparing different rates. Based on this assumption, these algorithms should be resilient to congestion. However, in extremely congested environments, theses solutions are also affected by throughput degradation, as with other previous approaches [33].

Using a different approach, the receiver-based auto rate (RBAR) algorithm [14] is a signal-strength-based solution that gets feedback on link quality from the receiver node to determine the optimal rate at the sender. This algorithm uses RTS/CTS control frames to piggyback this information. This solution is not affected by collisions and can perform well even in highly unstable environments. However, measuring the SNR and mapping it onto a specific rate is a complex task [2,4]. In addition, this solution requires changes in frame formats and introduces the RTS/CTS overhead, which is generally not used in practice.

Channel-aware rate adaptation (CHARM) [19] is a signal-strength-based mechanism that leverages link symmetry to obtain signal information at the transmitter without incurring RTS/CTS overhead. It also uses the SNR collected by wireless cards to help rate selection. However, link asymmetry and high fluctuation are two salient characteristics of wireless channels, especially in mobile environments. Based on that, a more recently developed mechanism is the rate adaptation in mobile environments (RAM) [9]. RAM allows the receiver to convey the feedback link information to the transmitter via ACK transmission rate variation. It requires some small changes for the protocol implementation, but RAM adopts a scheme to guarantee interoperability with legacy IEEE 802.11 devices.

The hybrid rate control algorithm [12] combines these techniques to reduce network delay and jitter. It uses history-based information on probe frames as the basis for operation. Moreover, it also uses the signal indicator SNR from ACK frames to infer the signal quality at the receiver, eliminating rates that definitely cannot improve performance and allowing quick adaptation in unstable environments.

Concerning vehicular networks, there is the context-aware rate selection (CARS) [34]. Vehicular networks face key challenges such as the rapid variations in link quality caused by mobility at vehicular speeds and sparse bursty transmissions, which forces the rate adaptation scheme to estimate link quality with few or no transmitted frames. CARS uses context information (e.g., vehicle speed and distance from neighbor) to systematically address these challenges while maximizing the link throughput using MRR capability during frame transmissions.

To overcome some issues associated with the aforementioned solutions, the cognitive rate adaptation (CORA) algorithm was developed [6]. It implements the cognitive approach proposed by the CogProt framework to perform rate adaptation. CogProt and CORA will be reviewed in the following subsections.

3.2 The CogProt framework

The CogProt cognitive framework [7,23,28] considers the principle of cognitive networks. This is a promising paradigm to address performance degradation resulting from changes in network conditions. It relies on cognitive algorithms to provide a dynamic reconfiguration of protocol parameters, through learning and reasoning, to optimize system-wide performance [37].

The proposed framework introduces the cognitive network node architecture presented in Fig. 1. It considers a cognitive plane in parallel to the protocol layers. This plane is capable of monitoring protocol parameters as well as controlling them by issuing configuration commands. The cognitive self-configuration process involves a quality feedback loop assisted by a knowledge base and remote cognitive

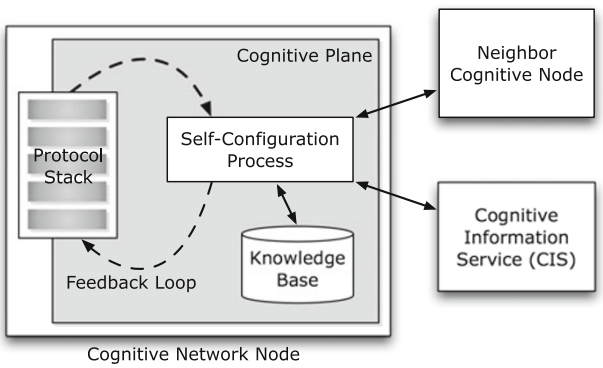

Fig. 1 Cognitive network node

information (i.e., from neighbor nodes and from a centralized CIS server).

Such quality feedback loop consists of three conceptual phases: data analysis, decision-making, and action. Consider the adjustment of a parameter of interest P within its operational range $P_i \in [P_{\min}, P_{\max}]$. Performance information is analyzed at the end of each iteration interval τ, according to a predefined quality metric associated with the parameter being adjusted. In the *data analysis* phase, the mechanism updates the knowledge base with average performance for the current value of P. In the *decision-making* phase, the mechanism analyzes both the knowledge base and the cognitive remote information to find the value of P that provides the best performance. The corresponding value is assigned to the mean of the normal distribution employed for the selection of the next value of P. In the *action* phase, a new random value is randomly generated according to the previous configured normal distribution and assigned to parameter P.

This loop continuously adjusts the mean of the normal distribution to the value of P that provides the best performance under current network conditions. Thus, the mean converges to the optimal value for P. As a result, that optimal value is chosen with a higher frequency because it is the mean of the normal distribution. Meanwhile, the mechanism will choose values nearby the mean, which allow it to react to changes on the network state [23].

3.3 The cognitive rate adaptation algorithm

The CORA rate adaptation algorithm was built upon the CogProt framework. CORA aims at maximizing the throughput (T) at the MAC level by periodically reconfiguring the data rate (R) based on past experience. To this end, CORA implements the CogProt's quality feedback loop as illustrated in Fig. 2. This loop follows a slightly different phase division known as Observe, Orient, Decide, and Act, as proposed in [5]. Each phase of this self-optimization process is explained below.

Observe Performance information on each data rate $R_i \in [R_{\min}, R_{\max}]$ is stored in a local knowledge base (KB). Let R_c be the current transmission data rate. In this first phase, the mechanism monitors the system performance measuring the throughput T_c obtained from the use of the current data rate R_c during the last time interval. Then, this information is averaged with an exponentially weighted moving average (EWMA) as follows:

$$KB_c = (1 - \alpha) * KB_c + (\alpha) * T_c \quad (1)$$

where KB_c is the stored performance information for the rate R_c, and α is the weight assigned to the currently measured performance T_c. In other words, the measured throughput for the current rate is used to calculate and update the average performance.

Orient During this phase, the mechanism identifies the rate with the best performance. It looks up the knowledge base for the data rate R_b that provides the highest throughput. The index b of the corresponding data rate is assigned to the mean μ of a normal distribution $\mathcal{N}(\mu, \sigma^2)$ that will be used in subsequent calculations.

Decide In this phase, the algorithm decides which rate R_r will be used during the next cycle iteration. This rate is selected from a random $r \in [\min, \max]$, generated according to the normal distribution previously configured. Later, this r is approximated to the integer index $r = \lfloor r + 0.5 \rfloor$.

Act Finally, the mechanism assigns the random rate R_r to the MAC layer, which completes the loop. All of the following frame transmissions up to the next quality feedback loop iteration will use this R_r rate.

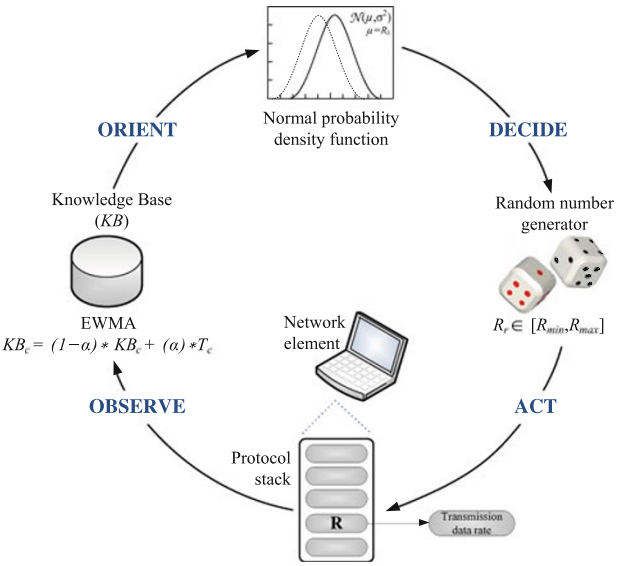

Fig. 2 Quality feedback loop implemented by CORA

This quality feedback loop is independently performed by each network node at the end of a sample interval $\tau = 100$ ms. It continuously adjusts the mean of the normal distribution to the index b of the rate R_b that provides the best throughput according to the knowledge information base. As a result, the mean of the normal distribution converges to the optimal value. Consequently, most of the randomly chosen values for R_r are optimal under current network conditions. Meanwhile, the cognitive process explores alternative rates "around" the best rate R_b to track eventual changes in the wireless environment. CORA algorithm does not make use of any remote cognitive information; this means it returns a

decentralized solution that relies only on information available at the network element itself.

Preliminary simulation results have demonstrated that CORA outperforms two classical rate adaptation algorithms (namely, ARF and RBAR), and it gets close to the best theoretically achievable performance in ad hoc scenarios with a single pair of nodes [6]. In an attempt to confirm these simulation results, the CORA mechanism was implemented into the OpenWrt Linux distribution and evaluated through experiments conducted on real network equipment. However, this validation revealed some shortcomings of the mechanism, including high computational usage due to unnecessary quality feedback loop iterations, frame loss due to poor rate selection, slow convergence to the appropriate transmission data rate, and reduced performance in stable environments (where the signal quality remains approximately constant over time but does not allow for the use of the highest available data rate).

To address aforementioned problems and increase the cognitive solution's performance, this paper introduces CogTRA, which is a deployable Cognitive Transmission Rate Adaptation mechanism. CogTRA was designed upon this same CogProt framework, but it is enhanced with novel features that were specifically designed to overcome some previous issues, making this a deployable mechanism that can be used in embedded network equipment. The following section introduces the CogTRA mechanism and discusses its characteristics in detail.

4 Proposed mechanism

CogTRA is a cognitive mechanism built upon the same foundation as CORA: the CogProt framework. However, to make it a high-performance deployable solution for real network equipment, it was necessary to incorporate some improvements. Before presenting the CogTRA solution, Sects. 4.1–4.4 discuss some problems that are addressed by this mechanism (specially those related to the algorithm's deployment) as well as the improvements that the CogTRA mechanism offers. Finally, Sect. 4.5 presents the entire algorithm, describing in detail all phases of the CogProt's quality feedback loop implementation.

4.1 Reducing computational overhead

To design a rate adaptation mechanism that can be implemented and used in embedded systems, such as home wireless network equipment, the algorithm should demand low computational resources. So, instead of a time-based interval as proposed by the CogProt framework (and also adopted by CORA), the CogTRA mechanism implements a packet-based interval to perform subsequent rate adaptations. In fact, we have observed that a packet-based approach reduces computational cost because the number of feedback loop iterations is reduced when the network is not in intense use.

Table 3 CogTRA packet-based rate adaptation approach

Scenario	Average traffic (kbps)	CogTRA (interval)
Quake3	72	148 (405 ms)
Heavy UDP	27,632	651 (92.2 ms)
YouTube	323	17 (3,567.9 ms)

A packet-based approach is also used by other data rate adaptation algorithms such as ARF, AARF, and AMRR. Some authors in the literature had already established that this approach has little influence on the behavior of the algorithms, particularly when the number of packets (Pkt_n) is suitably chosen [14,32]. In CogTRA, the default value of $Pkt_n = 150$ was selected for the interval between subsequent quality feedback iterations. This choice was made to get the packet-based interval close to the same 100 ms used by other algorithms, such as CORA and Minstrel. The average transmission time for a single packet of 1200 bytes in IEEE 802.11a networks, across all eight available rates, is 673 μs (where 673 μs × 150 ≅ 100 ms).

As a proof of concept, three different experiments were conducted to validate this improvement in the number of rate adaptations and the average time between them. All experiments are 60-s long and were performed on a high-quality signal environment, which allowed for the use of a data rate of 54 Mbps. CORA behavior is the same on all experiments: 600 rate adaptations using a 100 ms time-based interval. Table 3 shows the results for CogTRA.

The first experiment, called Quake3, used a traffic pattern similar to those generated by the online game Quake3[1]. It is possible to observe that for a small average UDP traffic with packets ranging from 50 to 150 bytes, the number of rate adaptations was reduced from 600 (under CORA) to 148 loop iterations using the packet-based approach. This is an improvement of 75 % in computational resource usage, increasing the time between consecutive rate adaptations to 405 ms with a 95 % confidence interval of 0.9 ms. The second experiment, called Heavy UDP, considers continuous UDP traffic with packets of 1,500 bytes saturating the network capacity. The average time between two consecutive rate adaptations is 92.2 ms with a 95 % confidence interval of 1.4 ms. Note that this is very close to the predicted value of 100 ms, with a computational cost only 8.45 % superior to CORA under extreme traffic load like this. The third experiment, called YouTube, considers a more realistic traffic of a YouTube 720p video download over a 1 Mbps ADSL

[1] This traffic was generated by the D-ITG application [38].

connection. In this case, the average time between consecutive rate adaptations was increased to 3,567.9 ms with a 95 % confidence interval of 3.7 ms. Only 17 rate adaptations were performed. This represents an improvement of $\approxeq 35.6$ times, mainly due to large TCP packets and low traffic.

Based on these experiments, it is possible to conclude that a packet-based approach incurs a higher computational cost (loop iterations) compared to CORA only in extreme cases. It is worth to emphasize that the CORA (and for that matter, Minstrel) time-based approach can be seamlessly executed in commercial devices. However, the improvement offered by CogTRA can help to avoid resource waste whenever possible.

4.2 Preventing dropping frames

Several high-latency devices support multi-rate retry (MRR), such as those equipped with Atheros chipsets. With MRR, it is possible to define a retry chain table with four segments. Each segment is an advisement to send the current packet at some rate, with a fixed number of retry attempts. Once the packet is successfully transmitted, the remainder of the retry chain table is ignored. A recent study has shown that MRR usage is very effective, especially in a non-congested environment [24].

In contrast to CORA (following the same principle used by AMRR and Minstrel), CogTRA also makes use of MRR's capability to optimize performance when transmitting frames. Table 4 shows how this retry chain table is populated by the CogTRA mechanism. With this approach, the first two transmission attempts are always performed with the random rate R_r, which is obtained from the normal distribution as specified by the CogProt framework. If the use of this rate R_r fails, then the next two attempts are performed using the best-known rate R_b (defined as the normal curve mean). Because R_b is the best rate for current link conditions, there is a high probability of successful transmission using this rate. If the previous transmission attempts fail, two new attempts are performed with the highest delivery probability rate R_p. Finally, the last two attempts are performed with the lowest rate R_l, which is the same rate used for management and control frames transmissions.

The decision to use two tries for each segment acknowledges two potential pitfalls: a single attempt can be badly affected by collisions or by some transient interference, while more than two attempts on each segment can significantly increase the overall retry limit (currently set to 7), inducing latency on the network.

Table 4 CogTRA's multi-rate retry chain table

Try	Rate	Attempts
r_0	Random rate (R_r)	$c_0 = 2$
r_1	Best performance rate (R_b)	$c_1 = 2$
r_2	Best probability rate (R_p)	$c_2 = 2$
r_3	Lowest rate (R_l)	$c_3 = 2$

4.3 Speeding up the convergence process

One of the main CogProt parameters is the standard deviation σ of the normal distribution, which controls the aggressiveness with which the algorithm tests random values other than the best-known one. To illustrate this concept, Fig. 3 exposes three normal curves with the same mean $\mu = R_\mathrm{o}$ (representing the optimal data rate) and different values for the standard deviation σ.

As shown in Sect. 3.3, any random value $r \in [R_\mathrm{o} - 0.5, R_\mathrm{o} + 0.5)$ will reflect in the use of data rate R_o. The area under the curve reflects the probability of finding a number within this interval. For the normal curve `B`, the hatched area graphically represents this probability. Decreasing σ, as in curve `A`, increases this probability. However, the chances of selecting a data rate other than R_o decreases, reducing the algorithm's aggressiveness. The opposite occurs in the case of curve `C`, which has a higher σ value.

This parameter imposes a trade-off between stability and convergence time during network operation. Using a low-level of aggressiveness results in good performance in stable networks, but it takes too long to identify some change and move the mean to a better value. A high-level aggressiveness results in quick convergence to the best rate, but it decreases performance when the link remains stable because it constantly checks other rates beyond the best-known one.

To overcome this issue, CogTRA introduces an aggressiveness self-adjustment (ASA) improvement. The key idea is to increase the value of σ whenever changes occur in link quality and decrease it during periods of stability. Thus, it is possible to keep the standard deviation always set to a value

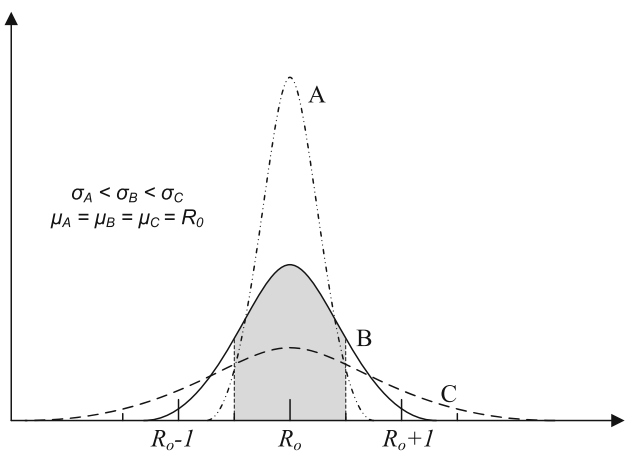

Fig. 3 Normal curves with $\mu = R_\mathrm{o}$ and different σ values

close to the ideal value, allowing the mechanism to quickly converge when needed but remain stable otherwise.

To identify link quality fluctuations, ASA checks whether the actual performance (T_r) differs from the previous one (T_o) by more than a δ_T%. When such a change occurs, it is possible to infer some link quality fluctuation, in which case ASA increases the current standard deviation σ by 0.1 U. Otherwise, the standard deviation is decreased by the same amount. Algorithm 1 describes this improvement. The initial value of σ is set to 1.5; δ_T is always 10 %; and the algorithm keeps the new standard deviation within the operating range $\sigma \in [0.4, 1.5]$. These parameters values are discussed in the following paragraphs.

To validate this improvement, experiments in an IEEE 802.11a network were conducted in a stable environment with two stationary nodes and a high-quality link between them. Figure 4 presents both the average throughput and the throughput over time for CogTRA using a fixed standard deviation $\sigma \in \{0.4, 0.8, 1.5\}$. When σ is controlled by ASA improvement, this figures also includes the values of σ over time.

At start-up, there is no information about data rates, and so frames are transmitted using the lowest base rate (6 Mbps). However, the link quality is sufficiently good so that the network nodes can communicate using the highest rate. When the algorithm performs rate adaptation for the first time, the initial convergence process starts. The analysis of the throughput over time indicates that $\sigma = 1.5$ accelerates the convergence process. However, this high value of σ reduces the maximum throughput achieved during stability. The opposite occurs when using a small value, such as $\sigma = 0.4$ or $\sigma = 0.8$. These lower values sustain high throughput while the network remains stable, but they are slower in reaching the stability.

It is possible to observe that CogTRA with ASA improvement makes the convergence faster, starting with $\sigma = 1.5$, and it maintains high throughput during stability, thanks to a small $\sigma = 0.4$. In this experiment, CogTRA with ASA resulted in a gain of 19.2 % compared to fixed $\sigma = 0.4$, 8.7 % compared to fixed $\sigma = 0.8$, and 13.9 % compared to fixed $\sigma = 1.5$. Notably, these gains were obtained from a single initial convergence process, and any other link condition change may trigger a new convergence process, which can also take advantage of this improvement.

To properly identify lower and upper bounds for the operational range of σ, the same previous scenario was used. CogTRA was configured without ASA improvement, but with fixed $\sigma \in \{0.2, 0.3, \ldots, 2.0\}$. Figure 5 shows the experiment results in terms of throughput, the time spent during the convergence process, and the rate stability after reaching the stable phase. It is considered that the algorithm reached stability when the same data rate is identified as the best rate twice consecutively.

When using extremely low σ values (i.e., 0.2 and 0.3), CogTRA spends too much time in the convergence phase, and it is even unable to reach the best data rate of 54 Mbps in some

Algorithm 1 Aggressiveness Self-Adjustment

Require: T_r, T_o, σ;
Ensure: σ;
1: $\delta_T \leftarrow 0.1 * T_o$;
2: **if** $|T_r - T_o| > \delta_T$ **then**
3: $\quad\sigma \leftarrow \min(1.5, \sigma + 0.1)$;
4: **else**
5: $\quad\sigma \leftarrow \max(0.4, \sigma - 0.1)$;
6: **return** σ;

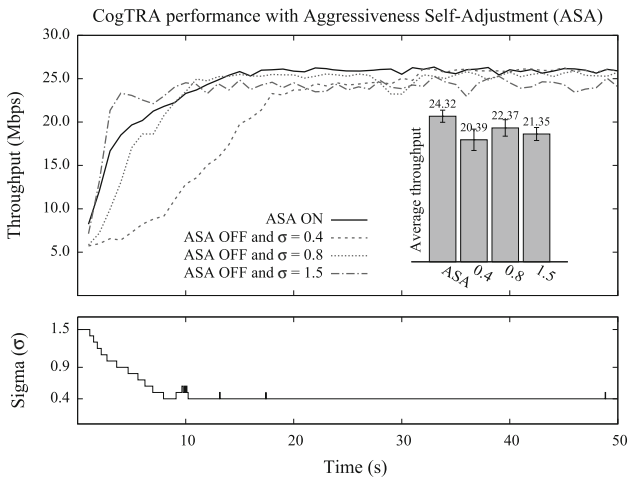

Fig. 4 CogTRA with Aggressiveness Self-Adjustment

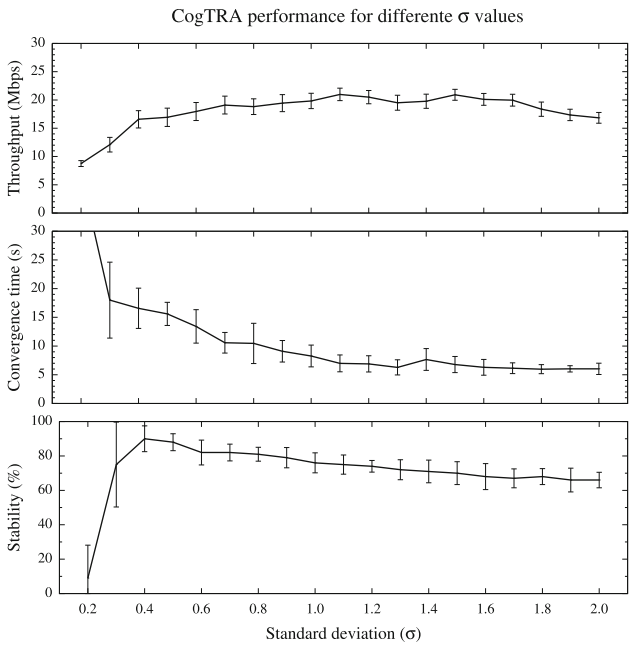

Fig. 5 CogTRA performance for different σ values

Table 5 ASA performances for different δ_T values

δ_T (%)	Throughput (Mbps)	95 % confidence
5	18.32	2.20
10	20.66	2.24
20	20.63	2.55

of the experiments for $\sigma = 0.2$. Moreover, the stability at 54 Mbps data rate is extremely low (9 %). However, $\sigma = 0.4$ provides the highest observed stability (90 %), an acceptable throughput, and modest convergence time. Because of that, $\sigma = 0.4$ was chosen as the lower bound, with the caveat that this value should only be used during periods of stability. As σ increases, it is possible to observe that the convergence time decreases and the throughput increases, improving overall performance. However, the stability decreases. Nevertheless, because high values of σ are generally not used during stable periods, this does not affect the performance. An upper bound of $\sigma = 1.5$ was selected because it reduces the convergence time to 6.78 s, which is close to the smallest observed value of 6.02 s using $\sigma = 2.0$. Furthermore, it was observed that after $\sigma = 1.5$, the throughput decreases, indicating that this value is an appropriate upper bound.

With respect to δ_T, the same scenario was used, now with the ASA improvement enabled. Table 5 shows the result in terms of the average throughput for experiments for $\delta_T \in \{5, 10, 20\ \%\}$. It is possible to observe that $\delta_T = 5\ \%$ reduces the throughput because it does not allow σ to stabilize; this is because 5 % is a small variation that can occur even when using the same rate over time. $\delta_T = 10\ \%$ and $\delta_T = 20\ \%$ both provide statistically identical results, and because of that, 10 % was arbitrarily selected as the default value.

4.4 Improving performance in stable environments

According to the CogProt framework, even when environmental conditions remain stable, the algorithm must randomly select and use different data rates other than the best-known one. This strategy is fundamental to the self-configuration functionality, as it allows the algorithm to check for link fluctuations and possibly adjust the normal curve. However, when there is a high-quality link, too much time is spent sampling slower rates. Thus, the throughput is lower than that derived from a higher fixed rate. The general idea is that CogTRA should sample less at lower rates if the link is working well. However, if the link deteriorates, CogTRA should immediately recover.

During CogTRA's rate adaptation process, when a higher rate is randomly selected, the transmission success probability tends to decrease. If the transmission succeeds at first attempt, then there is no performance loss. If the first two attempts with r_0 fail, then the next attempt will be performed using rate r_1 defined in the MRR retry chain table. As rate r_1 is precisely the best-known rate R_b, the impact on the network will be only affected by those first two unsuccessful attempts. The major problem arises when the randomly selected rate R_r is slower than the current best one (R_b). In this case, it is very likely that the frame transmission will succeed at first try, which initially seems to result in good performance. However, this lower rate spends more time during frame transmission, reducing network throughput and inducing the performance anomaly effect discussed in Sect. 2.

In an attempt to solve this problem, CogTRA features an interval self-adjustment (ISA) improvement. The working principle behind this improvement is the self-adjustment of the Pkt_n parameter, which reduces the number of packets that will be transmitted using the lower random rate R_r during the next interval to $\text{Pkt}_n = 20$. This approach minimizes the performance anomaly effect and mitigates throughput losses caused by the use of lower rates. At the end of this short interval, once the next rate adaptation is performed, the value is restored to its default $\text{Pkt}_n = 150$ unless another random $R_r < R_b$ is obtained for use. Algorithm 2 describes this feature.

To evaluate this ISA improvement, experiments in an IEEE 802.11a network were conducted in a stable environment with two stationary nodes and a high-quality link between them. Figure 6 shows the throughput over time and the average throughput for the CogTRA algorithm with and without ISA. It is possible to observe that there is no performance improvement during the first 10 s, when the mechanism is converging from the initial lowest 6 Mbps to the best rate of 54 Mbps. When CogTRA reaches stability, ISA starts improving performance by avoiding the unnecessary use of lower rates during long intervals. In this experiment, ISA improved network throughput by 6.6 %. This experiment was conducted without ASA improvement (as in Sect. 4.3). ASA reduces this effect somewhat by decreasing the aggressiveness of the algorithm while the environment remains stable. Still, lower rates need to be assessed continuously, and ISA takes action to prevent performance degradation when it occurs.

Algorithm 2 Interval Self-Adjustment

Require: R_b, R_r;
Ensure: Pkt_n;
1: **if** $R_r < R_b$ **then**
2: $\quad \text{Pkt}_n \leftarrow 20$;
3: **else**
4: $\quad \text{Pkt}_n \leftarrow 150$;
5: **return** Pkt_n;

The value of $\text{Pkt}_n = 20$ was chosen as a small value to minimize possible losses in network performance, but it

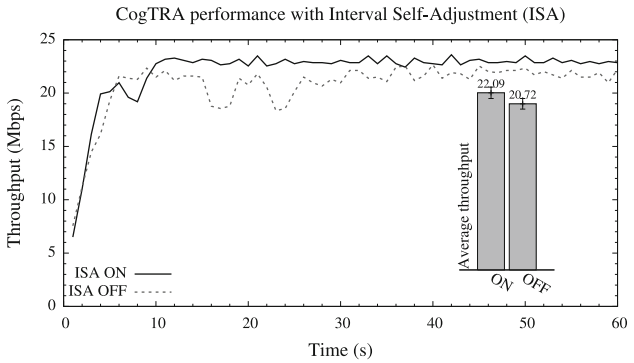

Fig. 6 CogTRA with Interval Self-Adjustment

Table 6 ISA performances for different Pkt_n values

Pkt_n	Throughput (Mbps)	95 % confidence
10	22.15	2.41
20	22.09	2.38
30	21.85	1.92

remains representative for throughput and delivery probability evaluations. To provide a better analysis of this parameter, this same experiment was executed with Pkt_n values of 10, 20, and 30. Table 6 shows the result in terms of the average throughput. It is possible to observe that there is no significant difference between the results, and because of that, a value of 20 was chosen as the default.

4.5 The CogTRA algorithm

CogTRA is a groundbreaking algorithm for transmission rate adaptation in IEEE 802.11 networks. It is built upon the CogProt framework and equipped with the improvements described in the previous subsections. The core of the algorithm is the quality feedback loop implementation. Each phase of this OODA loop is explained below:

Observe Performance information on each data rate $R_i \in [R_{\min}, R_{\max}]$ is stored in two local knowledge bases: the throughput base (KtB) and the probability base (KpB). CogTRA monitors the system performance by measuring both the probability of success in frame transmission P_i and throughput T_i obtained by each rate $R_i \in \{R_\text{r}, R_\text{b}, R_\text{p}, R_\text{l}\}$ that was used during the last interval. Equation 2 [36] is used to compute P_i and T_i:

$$P_i = \frac{\text{Suc}_i}{\text{Att}_i}; \qquad T_i = \frac{P_i * \text{Mb}_i}{Tx_i}; \tag{2}$$

where Att_i and Suc_i are the number of frame transmission attempts and successes, respectively; Mb_i is the total of Megabits transmitted; and Tx_i is the time for one try of one frame to be sent using rate R_i. T_i and P_i are averaged using the same EWMA Equation 1, and both KtB_i and KpB_i are updated with this current information. At this point, the ASA improvement compares the current updated throughput information for the random rate KtB_r against the previous KtB_r value for the purpose of adjusting the standard deviation σ of the normal distribution $\mathcal{N}(\mu, \sigma^2)$.

Orient At this point, CogTRA looks up KtB and KpB to derive rates R_b and R_p that provide the best throughput and the best delivery probability, respectively. The index b of the corresponding rate R_b is assigned to the mean μ of the normal distribution $\mathcal{N}(\mu, \sigma^2)$.

Decide In this phase, CogTRA decides which rate R_r will be used during the next cycle iteration. This rate is obtained from a random $r \in [\min, \max]$, which is generated according to the normal distribution previously configured. Later, this r is approximated to the integer index $r = \lfloor r + 0.5 \rfloor$. At this point, ISA improvement compares the current best rate R_b against the random rate R_r to check for $R_\text{r} < R_\text{b}$ and possibly reduce Pkt_n for the next interval.

Act In the action phase, the mechanism assigns the rates R_r, R_b, R_p, and R_l to the MRR retry chain table, completing the quality feedback loop.

Algorithm 3 formalizes this entire process. Lines 3–7 are responsible for updating knowledge bases with recent information. During the orientation phase, function `best_idx` (*KB*) at lines 10 and 11 looks up both knowledge bases for the indices corresponding to the rates with the best

Algorithm 3 The CogTRA quality feedback loop

1: ***{Observation phase}***
2: $T_\text{o} \leftarrow \text{KtB}_\text{r}$;
3: **for all** $i \in \{r, b, p, l\}$ **do**
4: $\quad P_i \leftarrow \text{Suc}_i/\text{Att}_i$;
5: $\quad T_i \leftarrow (P_i * \text{Mb}_i)/Tx_i$;
6: $\quad \text{KpB}_i \leftarrow (1-\alpha) * \text{KpB}_i + \alpha * P_i$;
7: $\quad \text{KtB}_i \leftarrow (1-\alpha) * \text{KtB}_i + \alpha * T_i$;
8: $\sigma \leftarrow$ Aggressiveness Self-Adjustment $(T_\text{r}, T_\text{o}, \sigma)$;
9: ***{Orientation phase}***
10: $b \leftarrow$ `best_idx` (KtB);
11: $p \leftarrow$ `best_idx` (KpB);
12: $\mu \leftarrow b$;
13: ***{Decision phase}***
14: $r \leftarrow \lfloor$ `random` $\left(\mathcal{N}(\mu, \sigma^2)\right) + 0.5 \rfloor$;
15: $\text{Pkt}_n \leftarrow$ Interval Self-Adjustment (R_b, R_r);
16: ***{Action phase}***
17: $r_0 \leftarrow R_\text{r}$;
18: $r_1 \leftarrow R_\text{b}$;
19: $r_2 \leftarrow R_p$;
20: $r_3 \leftarrow R_l$;
21: $c_0 \leftarrow c_1 \leftarrow c_2 \leftarrow c_3 \leftarrow 2$;
22: `sleep` (Pkt_n packets);

stored values. In the decision phase, a function called `random` ($\mathcal{N}(\mu, \sigma^2)$) derives a new random data rate index according to the normal distribution. This loops repeats at every Pkt_n packets transmission, as represented by a function called `sleep`.

Before each frame (f) transmission, Algorithm 4 identifies the frame type and attempt number used to select the appropriate rate (R_f). The frame type is obtained by the function `type` (f) in line 1. To ensure interoperability, the standard defines the control frames and any multicast or broadcast frame must be transmitted using the mandatory lowest rate R_l. In other cases, function `attempt`(f) returns the attempt number for frame f, which is used by function `mrr_rate`(att) to select the proper rate from the MRR chain table.

Algorithm 4 Send frame

Require: f;
Ensure: R_f;
1: $typ \leftarrow$ `type`(f);
2: **if** typ is (control **or** *multicast* **or** *broadcast*) **then**
3: $R_f \leftarrow R_l$;
4: **else**
5: $att \leftarrow$ `attempt`(f);
6: $R_f \leftarrow$ `mrr_rate`(att);
7: **return** R_f;

5 Performance evaluation

To evaluate performance in real environments, we have used the OpenWrt Linux distribution for network embedded devices, with a fully writable file system with package management. OpenWrt (version 10.03.1 backfire) was extended with CogTRA functionality [8] using the ath5k wireless driver. The modified firmware was compiled and installed into a Ubitiqui RouterStation Pro equipped with a MiniPCI Engenius 2.4/5 GHz NL-5354MP PLUS ARIES 2. This hardware was used as the AP in all experiments. The network clients were notebooks equipped with Ubuntu and Atheros wireless cards. The available bandwidth was estimated using Iperf, which is a common network-testing tool that generates TCP or UDP data streams on the network and measures the throughput of these streams [16].

Wireless network cards were configured according to the IEEE 802.11a standard at channel 36 (5,180 MHz). This setup was adopted because there was no other wireless network using the same spectrum nearby, allowing us to experiment in scenarios with controlled interference. There are 8 available nominal data rates in IEEE 802.11a: 6, 9, 12, 18, 24, 36, 48 and 54 Mbps. An integer index $i \in [0, 7]$ was mapped to each rate $R_i \in [R_{\min}, R_{\max}]$. At start-up, both knowledge bases were empty, and the standard deviation was set to $\sigma = 1.5$, the packet interval $\text{Pkt}_n = 150$, EWMA $\alpha = 0.75$ (Eq. 1), and initial rates to $R_\text{r} = R_\text{b} = R_\text{p} = R_\text{l} = 6$ Mbps (as a conservative approach). Experiments were 120 s long, with the first and last 10 s discarded as transient intervals. The results were obtained from 20 iterations, and average values are presented with 95 % confidence interval.

The rest of this section is organized as follows. Section 5.1 compares the proposed CogTRA mechanism with CORA. Section 5.2 presents results from experiments conducted in stable environments, while Sect. 5.3 shows CogTRA performance in dynamic environments. An interfering traffic was included in experiments at Sect. 5.4, and a scenario to evaluate the performance anomaly effect is considered in Sect. 5.5. Section 5.6 finalizes the performance evaluation, summarizing the results of all experiments.

5.1 CORA and CogTRA performance comparison

Early experiments were intended to demonstrate that the set of improvements comprising CogTRA mechanism had been indeed effective in addressing some pointed difficulties in the existing cognitive algorithm CORA. For that purpose, this first experiment considers only two network elements: an AP and a mobile client, with a single downloading TCP traffic between them. During the initial 60 s, the client stands near the AP, creating a high-quality wireless link that allows for the use of the highest available data rate. At this moment, it is possible to achieve the maximum network performance as well as to evaluate both algorithms in stable environments. At $t = 60$ s, the mobile client starts moving away from the AP, until it reaches an intermediate distance of 7 meters away from the AP (in an indoor environment). The client remains at this position until $t = 90$ s, when it starts moving again to reach 15 meters away from the AP. This is the farthest distance where network can sustain a satisfactory connection in the experimental environment.

Figure 7 shows experimental results for both cognitive algorithms CORA and CogTRA in terms of TCP throughput over time (top), average aggregate throughput (bar graph), and data rate usage over time (bottom). The data rate usage was obtained by sampling at constant intervals of 1 s, using the debug system available on OpenWrt. It is possible to observe that CORA performs well only in stable environments with a high-quality link (i.e., the first 60 s of the experiment). During this interval, the performance of both algorithms is practically the same, specially because in this experiment CORA was configured with $\sigma = 0.8$, which is a fixed value with good performance, according to Figs. 4 and 5.

When the mobile client moves away from the AP, decreasing link quality, CORA performance is drastically reduced.

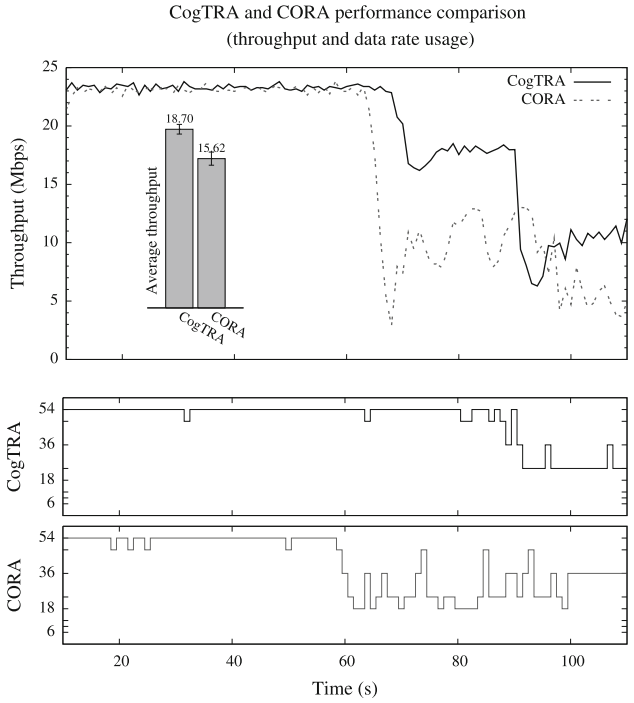

Fig. 7 CogTRA and CORA performance comparison

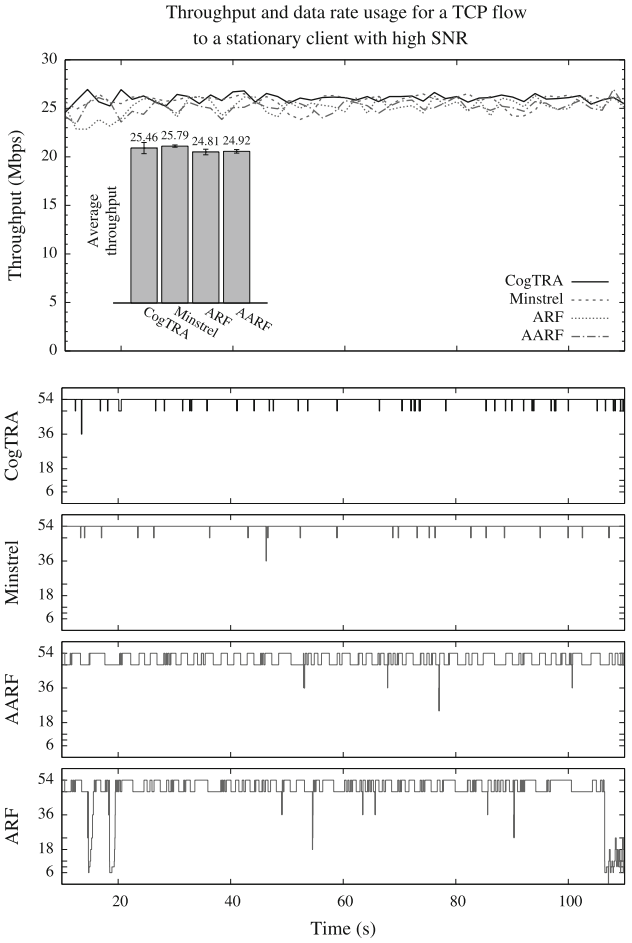

Fig. 8 CogTRA performance in high SNR scenario

Because CORA does not make use of MRR capabilities, after retrieving a random data rate R_r, this rate will be used by all frame transmissions up to the next quality feedback loop iteration. This means that, for each packet, the seven frame transmission attempts (that is, retry count) will be performed using the same R_r rate. If this rate does not work (as with a high rate in a low-quality link), then all frame transmission attempts will fail, and the packet will be reported as discarded to the upper layers. In this case, the TCP protocol will interpret this as network congestion and will reduce the throughput. This is the main reason for CORA's poor performance, especially just after the link-quality decreases (note that after $t = 70$ s, CORA's throughput increases somewhat). On the contrary, CogTRA uses MRR. If the first four attempts fail (two for the random R_r rate, and two for best rate so far), then the rate with the best success probability is used, possibly resulting in a frame transmission success that avoids all performance degradation caused by the TCP protocol. Another CORA behavior that can be observed involves the instability in selecting random rates under average-quality signal. This is an immediate consequence of the high standard deviation value, which is automatically adjusted by the ASA improvement in CogTRA.

In this experiments, CogTRA outperforms CORA by 19.7 % in average throughput, mostly in situations where the link quality is changing or stable at lower levels. It is possible to conclude that CORA, as it was originally proposed, is not suitable to be used in real environments. Most of these problems can be addressed by CogTRA, which will is evaluated and compared against the other rate adaptation algorithms in the following subsections.

5.2 Performance in stable scenarios

CogTRA was compared against the typical rate adaptation mechanisms, namely ARF, AARF, and Minstrel. Although there are other related, newer algorithms in the literature (such as RAM, CHARM, and CARS), CogTRA was compared here only with these traditional solutions and the widely used Minstrel. To the best of authors' knowledge, [40,41] are the only works that evaluate the performance of Minstrel by comparing it with other rate control mechanisms implemented in the madwifi and ath5k Linux drivers, such as SampleRate, Onoe, AMRR, and PID. The evaluations presented in these papers were carried out in a platform that provides a controllable and repeatable environment. The ultimate conclusion is that the Minstrel performance is superior, achieving more than fourfold the throughput of SampleRate in certain experiments. Therefore, we considered it suitable

to compare CogTRA with Minstrel, as the latter is a practical, high-performance solution that has been sparsely explored in the literature until now.

The first scenario for this experiment involves a single stationary client close (0.5 m) to the AP. There is a single downloading TCP traffic from the AP to the client. Because network nodes are close to each other, there is a high-quality link between them. In this configuration, the average SNR was measured in 35 dB, which allows for the use of the highest data rate (54 Mbps). Figure 8 shows the results in terms of throughput and data rate usage over time, and it also includes the average aggregate throughput for all mechanisms. To improve the analysis of the data rate usage, a custom debug mechanism was implemented, and the graph now shows the actual choice made by the algorithms.

It is possible to observe that all mechanisms perform well in this situation by always selecting the best data rate over time. Because ARF and AARF are solutions that attempt to use the highest effective data rate at each moment, this can be considered the perfect scenario for them. Nevertheless, it is possible to observe that these solutions often alternate between rates of 54 and 48 Mbps, indicating that there is some difficult in stabilizing. In practice, Minstrel and CogTRA also select the best data rate throughout the entire experiment. Moreover, ASA and ISA improvements are primordial in this scenario. Without them, CogTRA could not reach the highest performance in stable environments because it constantly selects random rates other than the best known one. Using both improvements, this becomes less frequent (ASA) and incurs a lower penalty (ISA). In terms of average throughput, there is no statistical difference that may indicate better performance for either algorithm. The average aggregate throughput for this experiment is 25.24 Mbps.

A second scenario was also evaluated that considers a stationary client not so close to the AP, which leads to a stable link with moderate quality. In this scenario, the SNR was measured in 12 dB on average, allowing for the use of an intermediate rate. Figure 9 shows the experiment results for this configuration. Here, it is possible to observe some significant differences. The most striking of them is the substantial inferior performance of ARF and AARF mechanisms. Because these two solutions are always trying to increase the data rate (ARF is more aggressive in this aspect compared to AARF), they become unstable, as can be observed by the data rate usage plot. Even when these two algorithms are using the rate with the best performance, they still attempt to use the next (not feasible) data rate. This induces frame loss, which decreases the throughput.

Average throughput for ARF is 2.90 Mbps, and for AARF, it is 4.73 Mbps. Is this scenario, CogTRA outperforms Minstrel mainly due to its stability; the former uses the proper 36 Mbps data rate 82 % of the time. Minstrel switches constantly between 36, 48 and 54 Mbps data rates, and it stays

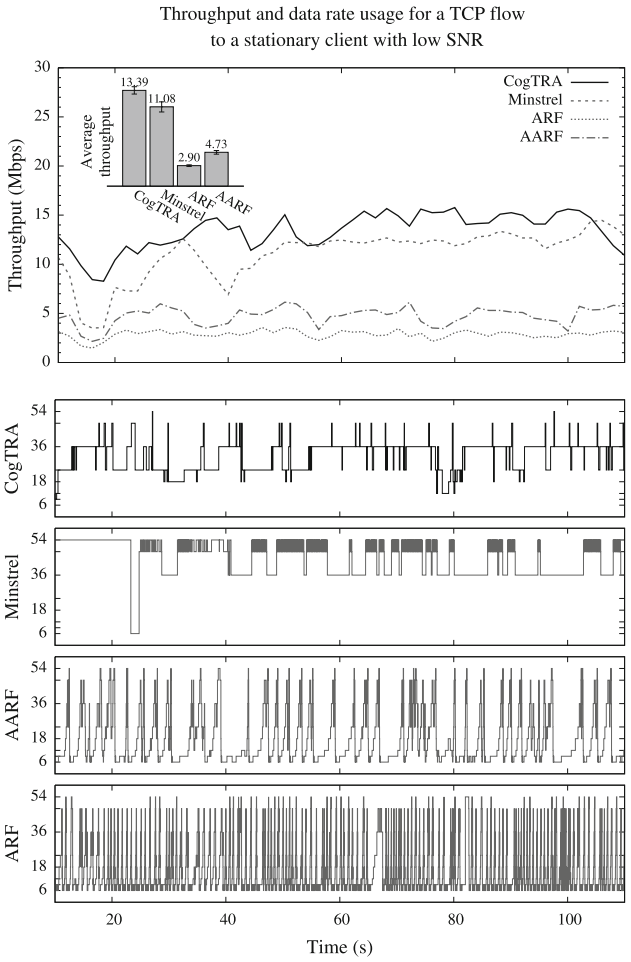

Fig. 9 CogTRA performance in low SNR scenario

on the proper 36 Mbps data rate only 45 % of the time. CogTRA average throughput is 13.39 Mbps, representing gains of 20.8 % compared to Minstrel and ≈ 250 % compared to ARF and AARF.

5.3 Performance in dynamic scenarios

For the purpose of evaluating CogTRA under dynamic environments, a scenario with a stationary AP and a mobile client was used for indoor experiments. During the first 25 s, the client remains side-by-side with the AP, with 0.5 m between them. Then, it starts moving away at ≈ 0.5 m/s until it reaches a distance of ≈ 12 m at $t = 50$ s. At $t = 70$ s, the client returns, reaching its initial position at $t = 95$ s. There is a single TCP traffic from the AP to the mobile client. Figure 10 shows the results, also including the average SNR value over time within a 95 % confidence interval.

As observed in previous experiments, the performance of all algorithms is satisfactory during the first and last seconds, when the signal quality remains stable at high SNR. At the moment the client starts moving away, the throughput for all

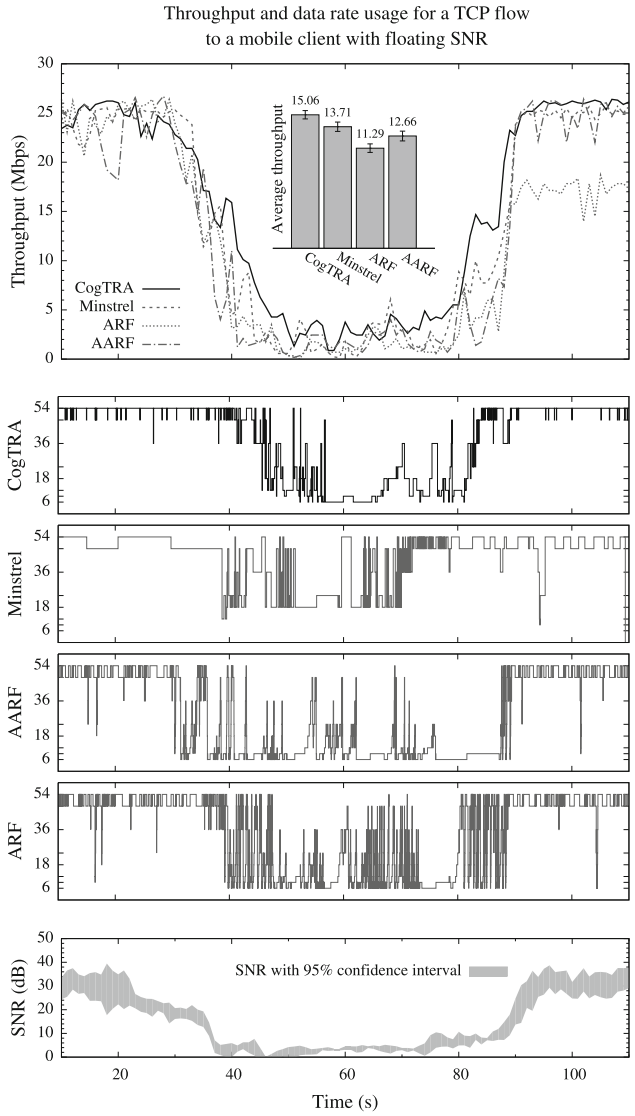

Fig. 10 CogTRA performance in dynamic scenario

mechanisms drops. At this moment ARF and AARF start to become unstable, switching between lower and higher data rates very often, as can be observed in the data rate usage graph. In contrast, Minstrel is resistant to decreasing the rate, and it always tries to use high rates even in moments of low signal strength. Because of this, many packets are dropped, affecting the performance. During mobility, CogTRA shows better performance compared to other algorithms. In dynamic environments, it is possible to see that CogTRA is fine-grained in terms of data rate usage due to Pkt_n self-adjustment (that is, ISA) and aggressive in unstable environments (i.e., ASA). As a consequence, CogTRA achieves better performance during the mobility period. Average throughput for CogTRA in this experiment is 15.06 Mbps, representing gains of 9.8, 33.3, and 18.9 % compared to Minstrel, ARF, and AARF, respectively.

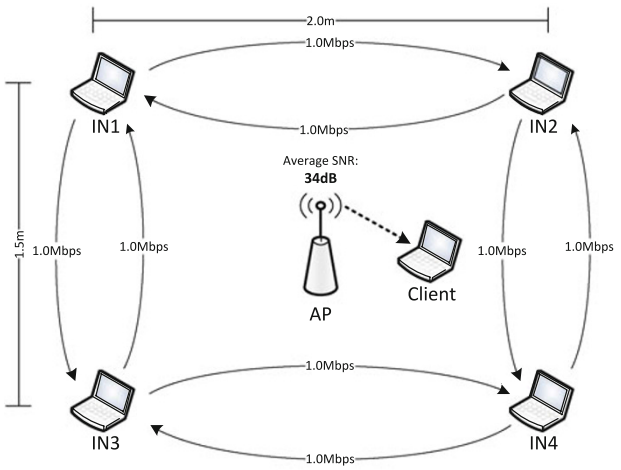

Fig. 11 Scenario used during experiments

5.4 Performance under interfering traffic

A problem related to every history-based rate adaptation algorithm is frame loss differentiation, which is needed to avoid unnecessary rate decreases due to collisions. To evaluate the performance of all algorithms in such situations, we extended the previous stable scenario with a high-quality link between the AP and the client (presented on Sect. 5.2) by adding four interfering nodes (INs) to generate eight interfering UDP flows at a constant throughput of 1 Mbps each. Figure 11 illustrates this topology. All nodes (i.e., the AP, the stationary client, and the four interfering nodes INs) are close to each other, allowing for the use of the highest 54 Mbps data rate during the entire experiment. The average SNR perceived by the AP was measured at 34 dB.

Figure 12 shows the results for this scenario with interfering traffic with collisions. An average FER of 16 % was observed during the experiment, indicating the presence of collisions, as all wireless links have high SNR, and the other parameters are the same as that for the first experiment of Sect. 5.2. It is possible to observe that ARF and AARF are highly affected by the interfering traffic, reducing the data rate and mitigating throughput performance. Minstrel and CogTRA are immune to this problem, sustaining higher rates and practically achieving the same performance during experiments. This happens because Minstrel and CogTRA do not rely on single frame transmission to infer rate performance, thereby avoiding unnecessary rate changes. Moreover, CogTRA sometimes selects the 48 Mbps data rate as a consequence of its cognitive approach in looking for network changes. Regardless, its performance is not severely affected thanks to ISA improvement that adjusts the Pkt_n parameter. There is no statistical gain of CogTRA over Minstrel, with an average throughput of 10.48 and 10.64 Mbps, respectively. CogTRA outperforms ARF by 185.2 % and AARF by 84.1 %.

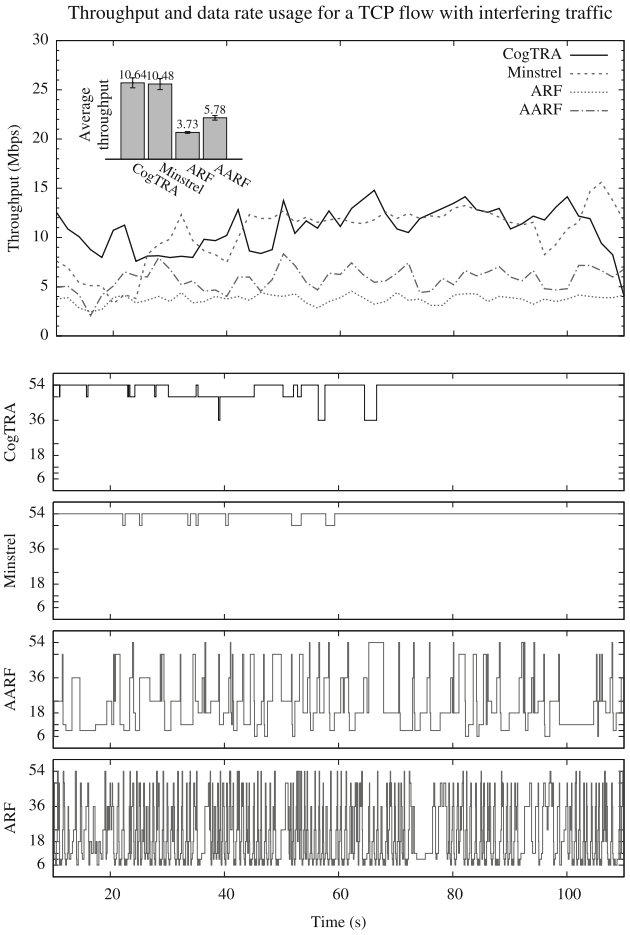

Fig. 12 CogTRA performance with interfering traffic

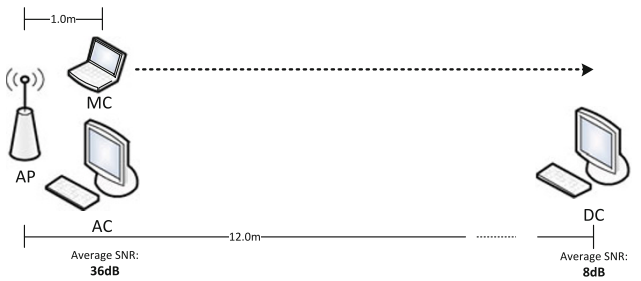

Fig. 13 Scenario used during experiments

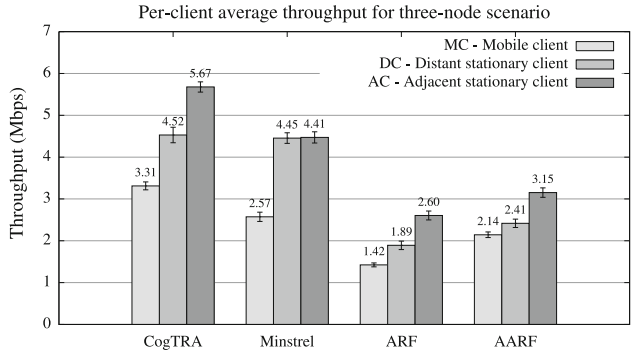

Fig. 14 CogTRA performance under anomaly effect

5.5 Performance under the anomaly effect

A final experiment was conducted to verify CogTRA behavior in environments with multiple clients associated with the same AP to analyze performance under the anomaly effect discussed in Sect. 2. This scenario included the AP and three clients: a distant stationary client DC, an adjacent stationary client AC, and a mobile client MC that moves away from the AP during experiment. There is a single uploading TCP traffic from each client to the AP. Figure 13 shows the topology of this experiment, including information about the average SNR at each location.

Figure 14 shows the per-client average throughput information for each TCP flow. It is possible to observe that the mobile client has the lowest performance compared to both stationary clients for all algorithms. This is expected because this mobile node demands more rate adaptations due to the unstable signal strength. As shown in Sect. 5.3, CogTRA outperforms other mechanisms in dynamic environments, and this behavior can also be observed here for this mobile node. With respect to both stationary clients, CogTRA allows higher TCP throughput when compared to other algorithms, especially for the closest stationary client.

Jain's fairness index [17] was used to quantify the difference between the throughputs achieved by each client, and it is represented by a percentage, whereby 100 % means that all clients have the same throughput. This index yields 95.6 % for CogTRA, 94.8 % for Minstrel, 94.3 % for ARF, and 97.3 % for AARF. The important conclusion is that CogTRA is affected by the anomaly effect in the same proportion as the other algorithms. In terms of absolute results, CogTRA has the higher throughput, reaching an average performance of 4.5 Mbps. This represents gains of 18.1, 128.4, and 75.4 % compared to Minstrel, ARF, and AARF, respectively.

5.6 Performance evaluation analysis

After analyzing the results of all experiments, it is possible to conclude that ARF and AARF provide competitive results in stable scenarios with high-quality links. In other cases, the instability of these solutions may affect their performance, reducing the throughput in dynamic environments.

Regarding Minstrel and CogTRA, we can observe that they provide high performance in stable environments. The absence of significant changes in the environment simply leads the algorithms to work during the initial network setup. However, in regard to dynamic scenarios, the CogTRA mechanism seems to be superior, performing a fine-grained

adaptation that improved results by almost 10 % in the conducted experiments. With respect to scenarios including some interfering traffic, the results indicate that CogTRA and Minstrel are not affected by frame collisions, effectively exhibiting the same performance in terms of throughput. However, the evaluated scenario is simple, and a more detailed analysis on the behavior of these mechanism must be conducted to draw more definitive conclusions. In the evaluation of performance anomaly effect, CogTRA was affected in the same proportion as the other algorithms and presented better results, surpassing Minstrel by 18.1 %.

6 Conclusion

This paper proposes CogTRA, which is a novel cognitive transmission rate adaptation mechanism enhanced with cutting-edge features to overcome the limitations of existing solutions. This mechanism was designed to work not only in stable but also in dynamic environments, as it quickly reacts to changes in channel conditions and achieves long-term stability in other conditions. Moreover, the mechanism self-adjusts some of its configurable parameters to properly suit the environment. Additionally, the use of MRR capability contributes to avoid dropping frames when a bad rate is selected. Finally, this is a deployable solution that does not require any changes to the IEEE 802.11 MAC protocol, and it can be implemented even on devices with limited resources due to its low processing requirements.

CogTRA was implemented on the OpenWrt Linux distribution and evaluated through experiments using real network equipment. The improvements that comprise this novel solution were discussed and experimentally evaluated independently. A complete performance evaluation was also performed to compare CogTRA with Minstrel and other typical rate adaptation solutions (namely, ARF and AARF). The results demonstrated that CogTRA is able to dynamically adjust the transmission data rate to always match current network conditions. The proposed mechanism outperforms Minstrel up to 20.8 % in specific scenarios.

Future work will focus on integrating the cognitive mechanism CogTRA into the latest IEEE 802.11n networks. This standard (1) relies on multiple antenna elements; (2) employs algorithms to leverage the consequent spatial multiplexing and diversity benefits that such antenna arrays can provide; and (3) offers up to 77 data rates. This process requires special attention, mainly in relation to the appropriate adjustment of the normal distribution's parameters and the large number of rates and solutions to avoid sampling at rates that cannot result in better throughput [26]. Furthermore, CogTRA performance should be evaluated in scenarios with interfering traffic and collisions using controlled environments as the ORBIT testbed [30].

Acknowledgments The authors would like to thank both CNPq (Grant number 132321/2008-4) and FAPESP (Grant number 2008/07770-8) for supporting this work.

References

1. 802.11-2007 (2007) 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. Technical report. IEEE Computer Society, New York. doi:10.1109/IEEESTD.2007.373646. Revision of IEEE Std. 802.11-1999
2. Ancillotti E, Bruno R, Conti M (2008) Experimentation and performance evaluation of rate adaptation algorithms in wireless mesh networks. In: PE-WASUN: proceedings of the ACM symposium on performance evaluation of wireless ad hoc, sensor, and ubiquitous, networks, pp 7–14. doi:10.1145/1454609.1454612
3. Biaz S, Wu S (2008) Rate adaptation algorithms for IEEE 802.11 networks: a survey and comparison. In: ISCC: proceedings of the IEEE symposium on computers and, communications, pp 130–136. doi:10.1109/ISCC.2008.4625680
4. Bicket JC (2005) Bit-rate selection in wireless networks. Master's thesis. Institute of Technology (MIT), Department of Electrical Engineering and Computer Science, Massachusetts. http://pdos.csail.mit.edu/papers/jbicket-ms.pdf
5. Boyd JR (1995) The essence of wining and losing. http://pogoarchives.org/m/dni/john_boyd_compendium/essence_of_winning_losing.pdf
6. Chaves L, Malheiros N, Madeira E, Garcia I, Kliazovich D (2009) A cognitive rate adaptation mechanism for wireless networks. In: MACE: proceedings of the IEEE international workshop on modelling autonomic communication environments. Springer, Berlin, pp 58–71. doi:10.1007/978-3-642-05006-0_5
7. CogProt—the cognitive framework. http://www.lrc.ic.unicamp.br/cogprot
8. CogTRA—cognitive transmission rate adaptation for OpenWrt. http://code.google.com/p/cogtra
9. Chen X, Gangwal P, Qiao D (2012) Ram: rate adaptation in mobile environments. IEEE Trans Mobile Comput 11(3):464–477. doi:10.1109/TMC.2011.91
10. Erez U, Trott MD, Wornell GW (2012) Rateless coding for Gaussian channels. IEEE Trans Inf Theory 58(2):530–547. doi:10.1109/TIT.2011.2173242
11. Gudipati A, Katti S (2011) Strider: automatic rate adaptation and collision handling. ACM SIGCOMM. Comput Commun Rev 41(4):158–169. doi:10.1145/2043164.2018455
12. Haratcherev I, Langendoen K, Lagendijk R, Sips H (2004) Hybrid rate control for IEEE 802.11. In: MobiWac: proceedings of the international workshop on mobility management and wireless access protocols, pp 10–18. doi:10.1145/1023783.1023787
13. Heusse M, Rousseau F, Berger-Sabbatel G, Duda A (2003) Performance anomaly of 802.11b. In: INFOCOM: proceedings of the international conference on computer communications, vol 2, pp 836–843
14. Holland G, Vaidya N, Bahl P (2001) A rate-adaptive MAC protocol for multi-hop wireless networks. In: MobiCom: proceedings of the international conference on mobile computing and networking, pp 236–251. doi:10.1145/381677.381700
15. Hou JC, Park KJ, Kim TS, Kung LC (2008) Medium access control and routing protocols for wireless mesh networks. In: Hossain E, Leung K (eds) Wireless mesh networks: architectures and protocols, chap. 4. Springer, Berlin, pp 77–111
16. Iperf. http://iperf.sourceforge.net

17. Jain RK, Chiu DMW, Hawe WR (1984) A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, vol 301, Digital Equipment Corporation. http://arxiv.org/abs/cs.NI/9809099
18. Joshi T, Ahuja D, Singh D, Agrawal DP (2008) SARA: stochastic automata rate adaptation for IEEE 802.11 networks. IEEE Trans Parallel Distrib Syst 19(11):1579–1590. doi:10.1109/TPDS.2007.70814
19. Judd G, Wang X, Steenkiste P (2008) Efficient channel-aware rate adaptation in dynamic environments. In: MobSys: proceedings of the international conference on mobile systems, applications, and services, pp 118–131. doi:10.1145/1378600.1378615
20. Kamerman A, Monteban L (1997) Wavelan-II: a high-performance wireless LAN for the unlicensed band. Bell Labs Tech J 2(3):118–133. doi:10.1002/bltj.2069
21. Kim J, Kim S, Choi S, Qiao D (2006) CARA: collision-aware rate adaptation for IEEE 802.11 WLANs. In: INFOCOM: proceedings of the international conference on computer communications, pp 1–11. doi:10.1109/INFOCOM.2006.316
22. Kim TS, Lim H, Hou JC (2006) Improving spatial reuse through tuning transmit power, carrier sense threshold, and data rate in multihop wireless networks. In: MobiCom: proceedings of the international conference on mobile computing and networking, pp 366–377. doi:10.1145/1161089.1161131
23. Kliazovich D, Malheiros N, Fonseca NLS, Granelli F, Madeira E (2009) CogProt: a framework for cognitive configuration and optimization of communication protocols. In: Mobilight: proceedings of the international conference on mobile lightweight wireless systems, pp 280–291. doi:10.1007/978-3-642-16644-0_25
24. Koci N, Marina M (2009) Understanding the role of multi-rate retry mechanism for effective rate control in 802.11 wireless lans. In: LCN: proceedings of IEEE conference on local, computer networks, pp 305–308. doi:10.1109/LCN.2009.5355094
25. Lacage M, Manshaei MH, Turletti T (2004) IEEE 802.11 rate adaptation: a practical approach. In: MSWiM: proceedings of the international symposium on modeling, analysis and simulation of wireless and mobile systems, pp 126–134. doi:10.1145/1023663.1023687
26. Lakshmanan S, Sanadhya S, Sivakumar R (2011) On link rate adaptation in 802.11n WLANs. In: INFOCOM: proceedings of the international conference on computer communications, pp 366–370. doi:10.1109/INFCOM.2011.5935183
27. Loiacono M, Rosca J, Trappe W (2007) The snowball effect: detailing performance anomalies of 802.11 rate adaptation. In: GLOBECOM: proceedings of the IEEE global telecommunications conference, pp 5117–5122. doi:10.1109/GLOCOM.2007.970
28. Malheiros N, Kliazovich D, Granello F, Madeira E, Fonseca N (2010) A cognitive approach for cross-layer performance management. In: GLOBECOM: proceedings of the IEEE global telecommunications conference, pp 1–5. doi:10.1109/GLOCOM.2010.5684313
29. OpenWrt Wireless Freedom. http://openwrt.org
30. ORBIT: open-access research testbed for next-generation wireless networks. http://www.orbit-lab.org
31. Perry J, Balakrishnan H, Shah D (2011) Rateless spinal codes. In: HotNets-X: proceedings of the ACM workshop on hot topics in networks, pp 6:1–6:6. doi:10.1145/2070562.2070568
32. Qiao D, Choi S, Shin K (2002) Goodput analysis and link adaptation for IEEE 802.11a wireless LANs. IEEE Trans Mobile Comput 1(4):278–292. doi:10.1109/TMC.2002.1175541
33. Ramachandran K, Kremo H, Gruteser M, Spasojevic P, Seskar I (2007) Scalability analysis of rate adaptation techniques in congested IEEE 802.11 networks: an ORBIT testbed comparative study. In: WoWMoM: proceedings of the IEEE international symposium on a world of wireless, mobile and multimedia, networks, pp 1–12. doi:10.1109/WOWMOM.2007.4351717
34. Shankar P, Nadeem T, Rosca J, Iftode L (2008) Cars: context-aware rate selection for vehicular networks. In: ICNP: proceedings of the IEEE international conference on network protocols, pp 1–12. doi:10.1109/ICNP.2008.4697019
35. Shannon CE (1949) Communication in the presence of noise. IRE proceedings of the Institute of Radio Engineers, vol 37, no. 1, pp 10–21
36. Smithies D (2005) Minstrel rate control algorithm. http://linuxwireless.org/en/developers/Documentation/mac80211/RateControl/minstrel
37. Thomas RW, Friend DH, Dasilva LA, Mackenzie AB (2006) Cognitive networks: adaptation and learning to achieve end-to-end performance objectives. IEEE Commun Mag 44(12):51–57. doi:10.1109/MCOM.2006.273099
38. Universita' degli Studi di Napoli "Federico II" (2011) D-ITG, distributed internet traffic generator. http://www.grid.unina.it/software/ITG/
39. Xia Q, Hamdi M (2008) Smart sender: a practical rate adaptation algorithm for multirate IEEE 802.11 WLANs. IEEE Trans Wirel Commun 7(5):1764–1775. doi:10.1109/TWC.2008.061047
40. Yin W, Bialkowski K, Indulska J, Hu P (2010) Evaluations of madwifi mac layer rate control mechanisms. In: IWQoS: proceedings of the international workshop on quality of service, pp 1–9. doi:10.1109/IWQoS.2010.5542745
41. Yin W, Hu P, Indulska J, Bialkowski K (2011) Performance of mac80211 rate control mechanisms. In: MSWiM: proceedings of the international conference on modeling, analysis and simulation of wireless and mobile systems, pp 427–436 (2011). doi:10.1145/2068897.2068970