Ad Hoc Networks 124 (2022) 102721

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Smart adaptive collision avoidance for IEEE 802.11

Yalda Edalat [a,*], Katia Obraczka [a], Jong-Suk Ahn [b]

[a] Department of Computer Science and Engineering, University of California, Santa Cruz, USA
[b] Department of Computer Engineering Dongguk University, Seoul, Republic of Korea

ARTICLE INFO

Keywords:
IEEE 802.11
Congestion avoidance
CSMA/CA
Machine learning
Enabling/disabling RTS/CTS

ABSTRACT

In this paper, we introduce a novel algorithm that uses machine learning to dynamically decide whether to enable or disable IEEE 802.11 DCF's RTS/CTS. Our algorithm continuously learns current networking conditions, namely air time, i.e. the ratio between the size of data/control information being transmitted and transmission rate, and network contention to compare the cost between using RTS/CTS or retransmitting data, and dynamically switches RTS/CTS on and off accordingly. Simulation results using a variety of WLAN- as well as wireless multi-hop ad-hoc network scenarios, including synthetic and real traffic traces, demonstrate that the proposed approach consistently outperforms current best practices, such as never enabling RTS/CTS or using a pre-specified threshold to decide whether to switch RTS/CTS on or off.

1. Introduction

IEEE 802.11's Distributed Coordination Function (DCF) is the most popular MAC protocol used in both wireless LANs and multihop wireless ad hoc networks (MANETs). It defines a basic Carrier Sense Multiple Access (CSMA) channel access method which uses physical carrier sensing and an optional link-layer acknowledgment (ACK) to confirm correct reception of transmitted data frames. DCF also specifies an optional mode which employs both physical- (i.e., CSMA) as well as virtual carrier sensing, or CSMA/CA [1]. CSMA/CA was proposed as a way to solve the so-called *hidden terminal* problem by allowing nodes to reserve the channel before engaging in data communication. They do so by exchanging short control frames, namely Request to Send (RTS) and Clear to Send (CTS) ahead of transmitting data. RTS/CTS has been part of the IEEE 802.11 standard since its early versions and has been in use since then, including in more recent variants such as 802.11n and 802.11ac [2]. Since collisions may occur only when the RTS frame is sent, and are detected by not receiving the CTS, the RTS/CTS handshake improves network performance by reducing the duration of a collision when long data messages (relative to the size of the RTS/CTS frames) are transmitted [3]. Motivated by that, IEEE 802.11 has defined a configurable parameter named *RTS Threshold (RT)*, or *RT*, which is used to enable and disable the RTS/CTS exchange. *RT* stipulates the minimum frame size that requires an RTS/CTS handshake before transmitting the frame. However, the IEEE 802.11 standard does not specify or recommend what *RT* value(s) to use.

A number of studies have explored techniques to dynamically set *RT*'s value based not only on frame size but also on other characteristics (e.g., transmission rate) and conditions (e.g., frame delivery ratio). Our work introduces a novel approach based machine learning to dynamically switch RTS/CTS on or off ahead of data transmission. The proposed Smart Adaptive Collision Avoidance technique, or SACA for short, considers a combination of: (1) "air time", i.e. the ratio between the size of data/control information being transmitted and transmission rate, and (2) network contention.

This paper builds on our preliminary work reported in [4] and makes the following additional contributions: (1) We propose a new, more general approach to evaluate network contention which is carried out at regular periods instead of measuring contention during the *learning period* at the beginning of a time slot, which required that RTS/CTS be switched off; (2) We use a more accurate cost model to calculate the overhead incurred by data and RTS/CTS collisions. These new cost functions are at the core of SACA since they determine whether RTS/CTS should be turned on or off on a per-frame basis; (3) We present new simulation results using the revised network contention measurement approach as well as the new cost functions to assess data and RTS/CTS collision overhead; (4) In addition to evaluating SACA in infrastructure-based scenarios, we also conducted simulations to assess SACA's performance in multihop wireless ad-hoc networks (MANETs).

The remainder of this paper is organized as follows. Section 2 provides a brief overview of IEEE 802.11 DCF and discusses RTS/CTS' performance characterization and trade offs. SACA, our network-aware, adaptive collision avoidance approach to dynamically enable or disable RTS/CTS is described in Section 3. Section 4 describes our performance evaluation methodology and Section 5 shows how SACA performs under different network scenarios when compared to IEEE 802.11. We

* Corresponding author.
E-mail address: yalda@soe.ucsc.edu (Y. Edalat).

https://doi.org/10.1016/j.adhoc.2021.102721
Received 30 November 2020; Received in revised form 14 September 2021; Accepted 11 October 2021
Available online 21 October 2021

discuss related work in Section 6. Finally, Section 7 concludes the paper with directions for future work.

2. IEEE 802.11 RTS/CTS handshake

IEEE 802.11 DCF provides two modes of operation, namely the *base mode* and the *collision avoidance* mode. In DCF's *base mode*, CSMA [5] is used by stations that have data to send to check whether the shared medium is being used. When a station wants to transmit a data frame, it first senses the channel to check whether it is idle for a DCF Inter-frame Space (DIFS) interval. If the channel is sensed idle, the station transmits the data. Otherwise, it defers transmission using a random backoff timer. After transmitting data, the station waits for an acknowledgement (ACK). If the ACK is received, the station considers that the data frame was successfully delivered. Otherwise, it will assume a collision occurred and runs a slotted Binary Exponential Backoff (BEB) scheme to retransmit the frame at a later time.

In *collision avoidance* mode, stations use CSMA/CA [1], which requires a station that has data to send, still performs carrier sensing to check if the channel is busy; and if the channel is idle, the station will then reserve it for its own transmission via a two-way handshake using small control frames, namely the Request to Send (RTS) and Clear To Send (CTS).

RTS/CTS benefits

The main goal of the RTS/CTS handshake is to combat the *hidden node* problem. While the RTS and the CTS frames are themselves still subject to collisions, the overhead incurred by an RTS or CTS collision is usually much lower than a data retransmission, especially for very long data frames, which has become a common trend in recent applications.

RTS/CTS's downsides

Although the RTS/CTS mechanism can mitigate collisions caused by hidden terminals, it also presents some drawbacks as described below.

Overhead. The RTS/CTS handshake incurs additional overhead by generating control traffic (i.e., RTS and CTS frames) as well as delaying data transmission. In the case of short data frames, the additional delay needed to perform the RTS/CTS exchange may not be worthwhile as the cost of a data frame and a RTS/CTS collision would be comparable. That was the motivation behind the *RTS Threshold* or *RT*, proposed by the IEEE 802.11 standard and commonly used in IEEE 802.11 implementations. *RT* specifies the minimum data frame size that will trigger the RTS/CTS handshake to reserve the channel ahead of a data frame transmission. If data frames are larger than *RT*, RTS/CTS is invoked for that specific frame. Otherwise, DCF's base mode is used.

Blocking non-interfering parallel transmission and False Blocking. RTS/CTS may block concurrent transmissions from other nodes that would not result in collisions. The RTS may block nodes within the sender's transmission range from transmitting even if their transmission would not interfere with the RTS sender's transmission. Similarly, the CTS may block nodes that receive it from receiving from other nodes.

RTS/CTS collisions. In crowded areas, where hidden terminals are prevalent, the RTS/CTS handshake is less effective as a collision avoidance technique [6]. This is because RTS and CTS frames are themselves subject to collision in the same way as data frames. When the traffic load is heavy and the number of hidden terminals is high, the chance of unsuccessful RTS/CTS handshake increases due to higher channel contention and thus higher collision probability. Besides the delay and overhead incurred by the retransmission of the RTS, the channel would be unusable for nodes who overhear the RTS and CTS for the time specified in the NAV.

Characterizing RTS/CTS's performance

In our previous work [4], we conducted an empirical characterization of RTS/CTS performance as a function of a number of factors including frame size, transmission rate (for both data and control frames), and network contention. We confirmed experimentally some analytical well-known results on the performance of RTS/CTS and showed that network contention, as well as frame size, and transmission rate must be collectively considered in order to decide whether to enable or disable 802.11's RTS/CTS mechanism. In the remainder of this section, we discuss each of these factors briefly.

Air time. The IEEE 802.11 standard suggests that the RTS/CTS access mode should be chosen based on the size of the data frame since the chance of a collision is higher for larger data frames. Additionally, the size of data frames should be large enough compared to the RTS/CTS frame size so that the overhead of transmitting RTS/CTS would be negligible.

Besides the size of the frame, the amount of time the channel is busy transmitting a frame, or the *air time*, is also dependent on the channel's transmission rate. The *air time* is thus given by Eq. (1) below.

$$air\ time = \frac{data/control\ \ frame\ size}{data/control\ \ frame\ rate} \tag{1}$$

A notable practical factor in the performance of RTS/CTS, which is frequently neglected, is that, in multi-rate WLANs, control frames such as ACK, RTS, and CTS are transmitted at a fixed basic rate regardless of the data rate. One of the main reasons is to enable interoperability and to accommodate legacy devices, since all devices in the network must be able to receive these frames. Consequently, control frame rate is another factor when studying RTS/CTS performance trade-offs.

Network contention. Network contention is another key factor to consider when deciding whether to use RTS/CTS. If collisions are not likely to occur (e.g., in low network load scenarios), there is no need to use RTS/CTS and incur additional overhead. There are mainly two situations that cause collisions: If two or more stations transmit at the same time (e.g., as a result of back off synchronization) or because of the existence of hidden terminals.

3. Adaptive collision avoidance

In this section, we describe our Smart Adaptive Collision Avoidance mechanism, or SACA for short, a simple yet efficient technique to dynamically enable or disable IEEE 802.11's collision avoidance in order to automatically adapt to a number of factors such as frame size, transmission rate (for both data and control frames), and network contention. The main idea behind SACA is to evaluate the cost-performance trade-off of using RTS/CTS to avoid data collisions. If RTS/CTS is deemed beneficial, i.e., it avoids collisions or reduces collision duration when the cost of a data collision is higher than the cost of the RTS/CTS exchange, then RTS/CTS is enabled; otherwise it is disabled. We discuss how these costs are defined and calculated in Section 3.2.

Algorithm 1 SACA

Collision estimation timeout:
  $Collision\_Estimation()$;
Frame transmission:
  If ($Data\_Retransmission\_Cost() \geq RTS/CTS\_Cost()$)
    then *enable* RTS/CTS
    else *disable* RTS/CTS

SACA is event-driven and handles two different events, namely: (1) *collision estimation timeout* and (2) *frame transmission*. As illustrated by SACA's pseudocode which is shown in Algorithm 1, when the collision estimation timer expires, a *collision estimation timeout* event is triggered. The $Collision\_Estimation()$ procedure is then invoked to measure the

current collision rate and to compute an estimate of the collision rate for future use (described in Section 3.1).

When a data frame is ready to be transmitted, a *frame transmission* event is triggered and the cost of using and not using RTS/CTS is calculated by $RTS/CTS\_Cost()$ and $Data\_Retransmission\_Cost()$ (described in Section 3.2), respectively. Based on how the cost of a data collision compares to the cost of the RTS/CTS exchange, a decision is made to enable or disable RTS/CTS. In the remainder of this section, we present SACA in more detail. We describe how SACA measures network contention, then, and how it uses its current network contention measurements to enable/disable RTS/CTS. Finally, we discuss SACA's implementation and overhead.

3.1. Adapting to network contention

As discussed in detail later in this section, network contention can be measured in a variety of ways. In this paper, we use the collision probability or *collision rate* as an indicator of network contention. Each node calculates its collision rate locally which captures the overall contention in the node's neighborhood. We measure it by dividing the number of failed transmissions, i.e., the number of unacknowledged frames at the transmitter by the total number of transmissions as shown in Eq. (2). Other ways to evaluate network contention in a node's neighborhood include the node's MAC queue length, mean time to access the medium, etc. In this work, because we are using a network simulator, more specifically *ns-3* [7], we use the expression in Eq. (2) because both its nominator ($number\ of\ failed\ transmissions$) and denominator ($total\ number\ of\ transm-issions$) are readily available in *ns-3* and provide fairly accurate collision measurement since losses that are not due to collisions are quite rare in the experimental scenarios we use. As we discuss in Section 3.3, this information is also readily available in real systems.

$$collision\ rate = \frac{number\ of\ failed\ transmissions}{total\ number\ of\ transmissions} \tag{2}$$

In order to continuously adapt to current network contention conditions, SACA measures collisions regularly. The frequency at which network contention is measured is one of SACA's parameters and deciding how often to measure collisions is discussed in . Network contention estimation is accomplished by the $Collision\_Estimation()$ procedure, which is invoked periodically triggered by the **collision estimation timeout** event. $Collision\_Estimation()$, whose pseudo-code is shown in Algorithm 2, measures current data- and RTS/CTS collision rates. Based on current collision measurements, $Collision\_Estimation()$ also estimates near-future collision rates using the SENSE estimator [8] described in Section 3.1 below.

Algorithm 2 Collision_Estimation

Initialization:
- $Data\ collision\ rate = 0$
- $RTS/CTS\ collision\ rate = 0$
- $Data\_collision\_estimation = 0$
- $RTS\_collision\_estimation = 0$

Collision rate measurement:
- $Compute\ Data\ collision\ rate$
- $Compute\ RTS/CTS\ collision\ rate$

Collision rate estimation:
- $SENSE\ computes\ Data\_collision\_estimation$
- $SENSE\ computes\ RTS/CTS\_collision\_estimation$

Collision rate estimation

Data and RTS/CTS frame sizes,[1] as well as data and signaling transmission rates are known at frame transmission time, which are used to calculate the *air time* for both data and RTS/CTS. However, current network contention conditions must be estimated on an ongoing basis. As previously discussed, in SACA, collision is continuously measured at fixed intervals and fed to the SENSE estimator [8] to estimate "near-future" contention based on previous collision history. SENSE uses a simple, yet effective, algorithm combining a machine learning approach known as Fixed-Share with Exponentially Weighted Moving Average (EWMA). SENSE is briefly described in the remaining of this section.

EWMA based predictors, calculate an exponentially weighted mean of the data [9]. Eq. (3) shows the EWMA equation where x_t is the estimated data and y_t is the data point that has been observed. α, the "smoothing factor", is set between 0 and 1 and controls how much weight is given to previous estimates (i.e., the "past") versus new samples (the "present"). The challenge raised by EWMA-based predictors is to find the appropriate value of α to use. Low values of α, favor the "past" over the "present", while with high α values, the "present" plays a more important role. Therefore, a-priori knowledge of the data's behavior is needed in order to choose an appropriate α.

$$x_t = \alpha \times y_t + (1 - \alpha) \times x_{t-1} \tag{3}$$

Fixed-Share algorithms [10] are a member of the Multiplicative Weight algorithmic family. In this family of algorithms, N experts are initialized with values within a target estimation range based on what they are trying to predict. For example, if the prediction is a value between 0 and 9, 10 experts can be used and initialized with values 0, 1, 2, ..., 9. Then, experts are assigned weights which are used to combine experts' values to compute the overall prediction. Every iteration of the algorithm, expert weights are adjusted based on a loss function, which measures the accuracy of the current prediction compared to the actual measurement of the variable being estimated. The impact of each expert on the overall prediction is determined by a weight associated with each expert. The weight of each expert is updated at the end of each trial based on the difference between its prediction and the real data. Although the Fixed-Share algorithm has been shown to perform well [11], it has some drawbacks. First, a fixed value within the range of possible values of the data is given to each expert as its prediction. As such, the data range should be known before using these predictors. Second, Fixed-Share cannot adjust to abrupt changes in data fast enough because it takes a long time for the weight of an expert to either shoot up or down when the expert's performance suddenly changes following an abrupt change in the data. Third, the accuracy of the algorithm is sensitive to the number of experts. More experts can cover more data from the data set. However, it may also introduce additional error.

SENSE is a variant of the Fixed-Share algorithm, where, instead of fixed-value experts, EWMA filters are employed as experts. Algorithm 3 shows SENSE's pseudo-code and Table 1 summarizes the notation used in Algorithm 3. During the **Initialization** phase, each expert is given a weight, $w_{i,1} = 1/N$, where N is the total number of experts. Each expert is also assigned an α value between 0 and 1. As shown in Algorithm 3, in the **EWMA Experts** step each experts' prediction is calculated as a weighted sum of the current data (y_t) and the previous prediction $x_{i,t-1}$. In the **Prediction** step, SENSE calculates the current prediction by adding the weighted predictions from N experts. The **Loss Function** step calculates the absolute difference between the actual data and each expert's forecast and normalizes this error with the maximum outcome y_{max}. The loss function is set to either the normalized error or the NULL function based on the normalized error. The **META Learning** step sets η, the "learning rate" which helps to adjust the experts' weights based on their recent performance. The less precise an expert's prediction, the more severe that expert is penalized. Finally, the **Restart Learning** step checks for significant changes in the mean of the observed data (level shift) and, if so, restarts its experts by only considering data after the level shift and resetting η of each expert.

[1] IEEE 802.11 RTS/CTS frame sizes are 34 bytes.

Algorithm 3 SENSE

Initialization:

$w_{1,1} = \ldots = w_{N,1} = \frac{1}{N}$

EWMA Experts:

$x_{i,t} = \alpha_i \times y_t + (1 - \alpha_i) \times x_{i,t-1}$

Prediction:

$\hat{y}_{i,t} = \frac{\sum_1^N w_{i,t} \times x_{i,t}}{\sum_1^N w_{i,t}}$

Loss Function:

$NE_{i,t} = \frac{|x_{i,t} - y_t|}{y_{max}}$

$$L(x_i, t)_{i,t} = \begin{cases} NULL & , NE_{i,t} \leq EL \\ NE_{i,t} & , Otherwise \end{cases}$$

META Learning:

$$\eta_{i,t} = \begin{cases} min(\eta_m ax, (\eta_{i,t-1} \times \beta)) \\ \qquad , NE_{i,t} > NE_{i,t-(j-1)} > NE_{i,t-j} \\ max(\eta_m in, (\frac{\eta_{i,t-1}}{\beta})) \\ \qquad , NE_{i,t} < NE_{i,t-(j-1)} < NE_{i,t-j} \\ \eta_{i,t-1} \\ \qquad , Otherwise \end{cases}$$

Weight Update:

$w_{i,t+1} = w_{i,t} \times e^{-\eta_{i,t} \times L(x_i,t)_{i,t}}$

Restart Learning:

If Level Shift is detected then,

$w_{i,t} = w_{i,t} \times e^{\sum_{t=-2T}^{t=-T} \eta_{i,t} \times L(x_i,t)_{i,t}}$

Table 1
SENSE's parameters.

Parameter	Description
$x_{i,t}$	Prediction of expert i at time t
y_t	Observed data at time t
$\hat{y}_t$	SENSE's prediction for time t
$w_{i,t}$	Weight of expert i at time t
NE_i	Normalized error of expert i
N	Total number of experts
$L(x_i, t)_{i,t}$	Loss of expert i at time t
y_{max}	Maximum data observed so far
$\eta_{i,t}$	Expert i's penalty at time t
β	Determines how much η_i should be increased or decreased
EL	Error limit (based on user's desired accuracy)
η_{min}, η_{max}	Limit experts' weight
j	Time window to evaluate expert's performance (used to update $\eta_{i,t}$)

Measuring network contention

The question of how to measure network contention is central to our approach. While we chose to use collision probability, or collision rate, as an indicator of network contention, there have been a number of proposals to estimate network contention, such as number of hidden terminals [12], mean medium access delay [13], frame delivery ratio [14], number of RTSs waiting for a CTS [15], to name a few. As part of our future work, we plan to evaluate how different approaches to measuring network contention impact the performance of SACA.

Another question that needs to be addressed is how often collision should be measured. We discuss some alternatives and their pros and cons below.

In SACA's preliminary version [4], time is divided into slots and during a *learning period* at the beginning of each slot, RTS/CTS is disabled and collision rate when data is transmitted is measured. Based on these measurements, SENSE estimates the data collision rate for the remainder of the slot. The advantage of using the learning period to measure collision rate is that, overall, it incurs less overhead. However, since collision rate is calculated only during learning periods, information about network contention may be out of date by the time the collision rate is used to decide whether to switch RTS/CTS on/off. Another problem is that, if contention is high, turning off RTS/CTS during the learning period may result in degraded performance. Additionally, sudden changes in network contention during the time slot will not be captured until the next learning period. Considering all the pros and cons discussed above, in SACA's current version, which is described in this paper, we decided not to include a learning period and measure the contention every t seconds. To validate SACA's new contention measurement approach, we ran simulation experiments comparing it against the approach used in [4] and confirmed its superior performance.

There is a clear trade-off in setting the value of t. Using a smaller t may allow SACA to better capture variations in contention conditions; however, it will incur higher overhead by invoking $Collision\_Estimation()$ more often. Another drawback of setting t too small is to capture short-lived variations in contention conditions. On the other hand, setting t too large results in less computation at the expense of running the risk of not adequately capturing network contention dynamics. But t also needs to be set large enough such that sufficient frame transmissions can be observed. Therefore, setting the value of t should also consider the underlying link speed and frame size. In order to set the value of t for our experimental evaluation of SACA, we ran several preliminary simulations with different values of t, namely 0.5, 1, 2, and 3 s and did not notice any significant differences in the results obtained. In the results reported in Section 5, we use t equal to 1 s.

3.2. RTS/CTS on–off

As shown in Algorithm 1, when a frame is ready for transmission, based on the collision estimation from $Collision\_Estimation()$ and the information from the frame itself, i.e., frame size and transmission rate which is used to calculate the frame's *air time*, the cost of retransmitting data and the cost of the RTS/CTS handshake are calculated by $Data\_Retransmission\_Cost()$ and $RTS/CTS\_Cost()$, respectively. RTS/CTS is enabled (by setting $RT = 0$) for that specific frame if the data retransmission cost is higher or equal to the RTS/CTS handshake cost; otherwise RTS/CTS is disabled (by setting RT to a large value). $Data\_Retransmission\_Cost()$ and $RTS/CTS\_Cost()$ are described in Section 3.2 below.

Data collision versus RTS/CTS handshake

In an ideal scenario when there is no contention, a node can avoid RTS/CTS handshake since there is no risk of collisions. But in congested environments, packets transmitted by different nodes which start to transmit at the same time may collide. Also, frames transmitted by hidden terminals may collide at the receiver even if the sender's transmission does not start exactly at the same time. In these situations, a node can decide to use RTS/CTS to reserve the channel and avoid data frame collisions. However, performing the RTS/CTS handshake ahead of data transmission will incur additional overhead. In order to determine if enabling RTS/CTS is beneficial, SACA compares the cost of the RTS/CTS exchange (calculated by $RTS/CTS\_Cost()$) against the cost of data retransmission in case of collision (computed by $Data\_Retransmission\_Cost()$). Basically, for every data frame transmission, the sender calculates the cost of retransmitting the frame in case of collision and compares it against the RTS/CTS overhead. The idea is to make this decision based on current network conditions as well as frame size and transmission rate.

Cost of data frame collision. The cost of a data frame collision, i.e., the cost to retransmit a frame until successfully received is calculated as:

$$Data\ Retransmission\ Cost = overhead\ of\ one\ frame\ retransmission \times average\ number\ of\ retransmissions \quad (4)$$

To calculate the overhead of data frame retransmission, we consider IEEE 802.11's *base mode* which uses only CSMA (no RTS/CTS). As illustrated in Fig. 1, which shows the transmission sequence of base mode according to the IEEE 802.11 standard, either the data frame

will be transmitted successfully and will not incur any retransmission overhead, or the frame will collide which will require retransmission. Fig. 1 illustrates a scenario with two hidden terminals, $Sender1$ and $Sender2$, in which Sender 1's first transmission to $Receiver$ is successful, but the second one collides with $Sender2$'s transmission.[2] When a collision happens, the total amount of time spent retransmitting the frame includes: the $DIFS$, the average backoff time (BO), the data frame retransmission time (i.e., the data frame's *air time*) ($Data$), plus the ACK timeout which is $SIFS$ plus ACK time. Therefore, the overhead of one frame retransmission is:

$$DIFS + BO + Data + SIFS + ACK \tag{5}$$

where the backoff time (BO) is calculated by:

$$\sum_{k=0}^{inf}(1/2) * min((2^k * (CW_{min} + 1) - 1, CW_{max}) * Pdc^k * (1 - Pdc)) \tag{6}$$

Note that in IEEE 802.11, $CWmin$ is typically set to $2^a - 1$, $a = 2, 3, 4$. As a result, the term $2^k * (CWmin + 1) - 1$ becomes $2^{(k+a)} - 1$.

Meanwhile, the average number of retransmissions is calculated by:

$$\sum_{k=0}^{inf} k * (P_{DC})^k * (1 - P_{DC}) = P_{DC}/(1 - P_{DC}) \tag{7}$$

where P_{DC} is the probability of data packet collision.

From Eqs. (4), (5), and (7), we derive Eq. (8):

$$Data\_Retransmission\_Cost() = (DIFS + BO + Data + SIFS + Ack) \times (\frac{P_{DC}}{1 - P_{DC}}) \tag{8}$$

Cost of RTS/CTS handshake. To calculate the RTS/CTS overhead, we consider IEEE 802.11's operation in congestion avoidance mode as illustrated in Fig. 2, which shows the transmission sequence of congestion avoidance mode according to the IEEE 802.11 standard. In this mode, either the RTS/CTS will be transmitted successfully and then will be followed by data and ACK frames or there is a possibility that the RTS frame collides with a frame from another node. In this figure, we show the case of RTS collision from two hidden nodes, $Sender1$ and $Sender2$, but RTS collisions can also happen between two non-hidden nodes if they start transmission at the same time. When RTS and CTS frames are successfully transmitted, the overhead incurred includes RTS air time (RTS), $SIFS$, CTS air time (CTS) and another $SIFS$. However, when an RTS collides, the overhead includes $DIFS$, BO, RTS, and CTS timeout which is $SIFS$ plus CTS. RTS collision cost is much less than data frame collision cost since RTS and CTS frames are typically much smaller than data frames. Similarly to previous work (e.g., [16]), we use the number of CTS timeouts as an indicator of the number of RTS collisions. We denote by P_{RC} the probability of an RTS collision. In the case of RTS/CTS transmission:

$$\begin{aligned} RTS/CTS\ Cost = {} & RTS/CTS\ successful\ transmission\ overhead + \\ & overhead\ of\ RTS/CTS\ retransmission \times \\ & average\ number\ of\ RTS/CTS\ retransmissions \end{aligned} \tag{9}$$

where the RTS/CTS successful transmission overhead is:

$$RTS + CTS + 2 \times SIFS \tag{10}$$

The overhead of RTS/CTS retransmission is:

$$DIFS + BO + RTS + SIFS + CTS \tag{11}$$

and the average number of RTS/CTS retransmissions is:

$$\sum_{k=0}^{inf} k * (P_{RC})^k * (1 - P_{RC}) = P_{RC}/(1 - P_{RC}) \tag{12}$$

From Eqs. (10), (11) and (12), we derive Eq. (13) for the overall RTS/CTS cost:

$$\begin{aligned} RTS/CTS\_Cost() = {} & (RTS + CTS + 2 \times SIFS) + \\ & [(DIFS + BO + RTS + SIFS + CTS) \times (\frac{P_{RC}}{1 - P_{RC}})] \end{aligned} \tag{13}$$

3.3. SACA's implementation and overhead

As illustrated in Algorithms 1, 2, and 3, simplicity and low overhead were among SACA's main design goals. Additionally, SACA was designed so that each node can run independently of other nodes and only needs information about local conditions, i.e., contention in its neighborhood. In other words, each node evaluates network contention it has been experiencing in recent past and, for each frame to be transmitted, decides to enable or disable RTS/CTS regardless of other nodes. That allows SACA to be able to coexist with legacy devices. SACA can be implemented as a separate module that is invoked by 802.11 on a per-frame basis.

SACA uses collision probability as an indicator of network contention in order to dynamically decide whether RTS/CTS should be used or not. To do that, each node keeps track of the total number of successful- and unsuccessful transmissions (see Eq. (2)), both of which are usually readily available in real systems, along with frame size and transmission rate (available from the frame's header). We should also point out that there is a tradeoff between how frequent network contention (in our case collision probability) is measured/estimated and the resulting overhead. In Sections 6 and 3.1, we note that, unlike our prior work [4] which measures collision probability only once at the beginning of a slot, our current approach estimates collision probability every t seconds. Clearly, the more frequent collision probability is measured, i.e., using lower values of t, the more up to date information on current conditions is, but the higher the overhead. In fact, the value of t can be automatically adjusted depending on the dynamics of the underlying network/system, which is one of the future work directions we plan to pursue.

In our prior work [17], we have implemented the Fixed-Share algorithm in the Linux kernel and ran "live" experiments in a testbed, confirming that SACA can be implemented and run on real devices. As part of future work, we use our Linux implementation of Fixed-Share to conduct SACA experiments in a real testbed.

4. Experimental methodology

We evaluate SACA through extensive simulations using a variety of infrastructure-based as well as ad-hoc network scenarios and show that it can automatically adjust to changes in network contention while accounting for airtime to decide whether to enable or disable the RTS/CTS handshake. In this section, we describe our experimental setup and performance metrics. Our results are presented in Section 5.

In our simulations, we used the *ns-3* network simulator [7] and its implementation of the IEEE 802.11n. We use *ns-3*'s *Matrix Propagation Loss Model* and set the propagation loss between each pair of nodes to make them hidden or not hidden from each other. According to *ns-3*'s channel model, if the propagation loss between two nodes is greater than 200 dB, they are considered hidden from one another. For example, if we set the propagation loss between nodes A and B to a very high value, then A becomes hidden from B, and vice-versa. For the infrastructure-based network simulations, 50 nodes were placed randomly within the transmission range of an Access Point (AP) in a 500 × 500 m area. The ad-hoc network simulation scenarios also used 50 nodes randomly placed in a 500 × 500 m area but without any

[2] Note that, as mentioned earlier, there is the possibility of collisions with non-hidden nodes as well which is not illustrated in this figure.

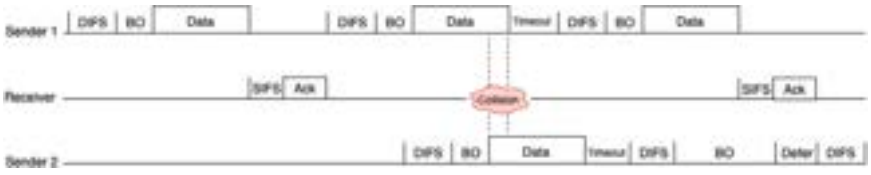


Fig. 1. Successful and unsuccessful data transmission in IEEE 802.11 base mode.

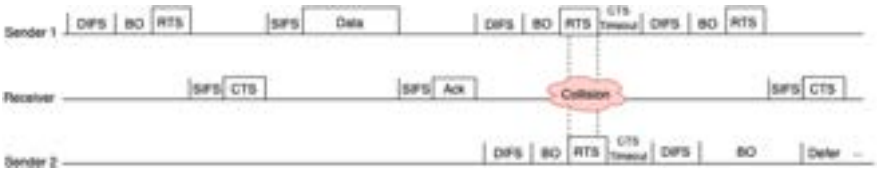


Fig. 2. Successful and unsuccessful data transmission in IEEE 802.11 collision avoidance mode.

Table 2
Simulation parameters.

Area	500 × 500 m^2
Total number of nodes	50
Routing protocol	AODV (ad-hoc simulations only)
Traffic type	CBR
DIFS and SIFS	50 and 10 μs
802.11 version	802.11 n
Propagation loss model	Two-ray ground

Table 3
Hot-spot and Company campus network traces.

Trace name	Hot-spot	Company campus network
Location	Coffee shop	Company campus
Time	Around noon	–
Number of flows	10	5
Duration	20 min	30 min
Frame size range	34–2150 bytes	34–11 000 bytes
802.11 version	802.11n	802.11n
Data rate range	6–300 Mbps	6–450 Mbps

communication infrastructure (e.g., APs). As such, the ad-hoc network scenarios used the AODV protocol [18] to perform routing and forwarding. We used the Two-Ray Ground channel propagation loss model in our simulations in order to account for multipath effects in wireless communication. Note that the effect of node location is captured by both the channel propagation model as well as the perceived contention in the node's neighborhood. Table 2 lists simulation parameters and their values used in our simulations.

4.1. Traffic traces

We use three different traffic traces to drive our simulations, namely a synthetic data trace as well as traces collected in real networks. In the synthetic data trace, we vary frame size and the number of senders, and consequently collision rate, every 5 s (total simulation time is 50 s) to evaluate how closely SACA is able to track network contention fluctuations. Table 4 summarizes the synthetic trace we use showing the sequence of frame sizes and number of senders as they vary every 5 s. In all cases, sender- and receiver nodes are selected randomly from the set of participating nodes.

We also drive SACA using real traffic traces captured in a public hot spot and a company campus using a wireless sniffer. Data rates and frame sizes provided in the Radiotap Header are used to calculate a frame's airtime. Table 3 summarizes the real traffic traces used in our simulations. For the hot spot trace, we captured 10 flows[3] between users and the access point (AP) for 20 min and feed the flows to the *ns-3* simulator. In the 10-sender scenario, each of the 10 flows is assigned to a single sender–destination pair, while in the 30- and 50-sender scenario, each flow is assigned to 3- and 5- sender–destination pairs, respectively. Each flow has a slightly different start time to avoid excessive contention at the start of the simulations. For the company campus trace, 5 individual flows were captured and each flow was assigned to 1, 2, 6, and 10 senders in the 5-, 10-, 30-, and 50-sender scenarios respectively. Each experiment is run 10 times using different seeds.

[3] A flow refers to traffic between the same source–destination pair.

4.2. Performance benchmarks and metrics

We evaluate SACA by comparing its performance against IEEE 802.11's base mode and congestion avoidance mode with different *RT* values. We should point out that we use these different comparison baselines to represent existing RTS/CTS control mechanisms, e.g., [12]. In our prior work [4], we show that network contention, as well as frame size, and transmission rate must be collectively considered in order to decide whether to enable or disable 802.11's RTS/CTS mechanism. As discussed in Section 6, SACA is the first approach that uses a combination of air time (which is the ratio between frame size and transmission rate) and information about local network congestion to dynamically enable or disable the RTS/CTS handshake.

As performance metrics, we use average throughput and average throughput improvement, which are calculated as follows. Average throughput is the number of bits per second received at each node (also known as "goodput"), then averaged over all nodes; average throughput improvement is calculated as the difference in throughput between SACA and the other approach divided by the other approach's throughput. For infrastructure-based scenarios, we also examine the contention an AP experiences when all nodes are hidden from each other as well as when all nodes are exposed to one another.

5. Results

Results from our performance evaluation study comparing SACA against IEEE 802.11 are presented in two parts, namely: infrastructure-based scenario and ad-hoc scenario results.

5.1. Infrastructure-based scenarios

Average throughput

In these simulation experiments, we compare SACA's average throughput against IEEE 802.11's DCF base mode (i.e., no RTS/CTS), IEEE 802.11's DCF congestion avoidance mode (i.e., RTS/CTS always enabled), and when using statically configured values for the RTS Threshold *RT*, namely 200-, 500-, 1000-, 1500-, and 2000 bytes. In all

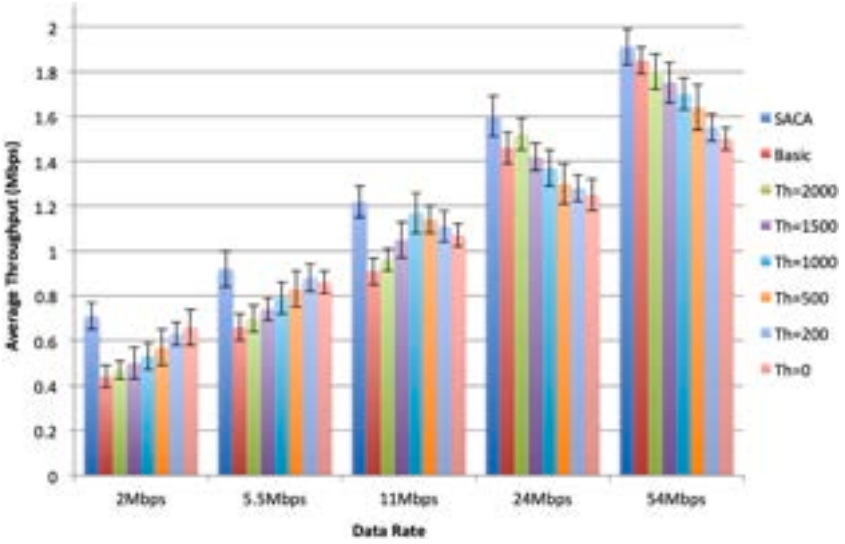


Fig. 3. Average throughput with the synthetic trace in infrastructure-based scenario. 95% confidence intervals are shown.

Table 4
SACA's slot-by-slot behavior using the synthetic trace with different data rates in infrastructure-based scenario illustrating how one of the sending nodes dynamically switches between Basic (B) and RTS/CTS (R/C) modes based on air time and perceived congestion.

Time (s)	Frame size (bytes)	Number of senders	Collision	54 Mbps	24 Mbps	11 Mbps	5.5 Mbps	2 Mbps
0 to 5	1500	5	5%	B	B	B	B	B
5 to 10	500	8	9%	B	B	B	B	R/C
10 to 15	2000	14	26%	B	B	R/C	R/C	R/C
15 to 20	200	20	38%	B	B	B	B	R/C
20 to 25	1000	24	47%	B	R/C	R/C	R/C	R/C
25 to 30	2000	30	59%	R/C	R/C	R/C	R/C	R/C
30 to 35	500	35	66%	B	R/C	R/C	R/C	R/C
35 to 40	200	38	71%	B	B	R/C	R/C	R/C
40 to 45	1500	43	78%	R/C	R/C	R/C	R/C	R/C
45 to 50	500	45	83%	B	R/C	R/C	R/C	R/C

simulations (unless otherwise specified), half of the nodes are hidden from other nodes. For example, in the 10-sender scenario, 5 senders are hidden, i.e., they can see the AP but they cannot see any other sender. The other senders, which are not hidden, can see the AP as well as the other senders. Each experiment is run for 10 times using different seeds and nodes are selected to be hidden or not hidden randomly.

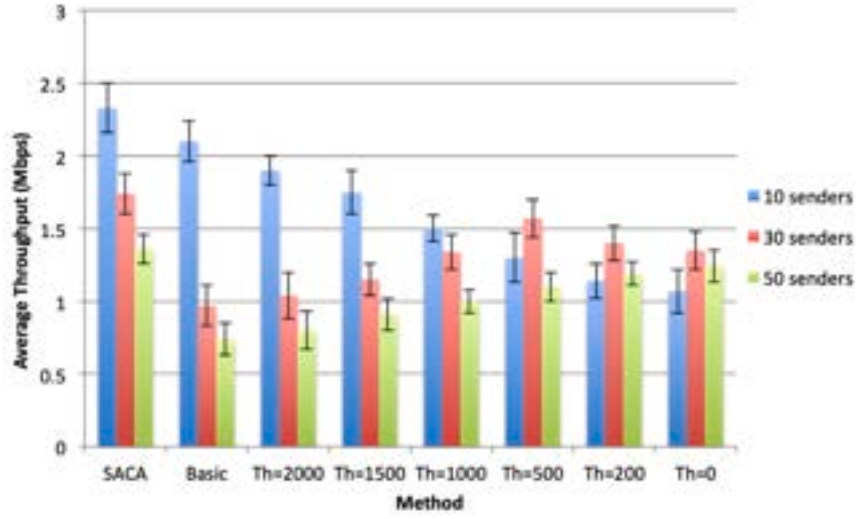
Synthetic trace. To show how SACA adjusts to network dynamics, we periodically changed the frame size and number of senders (see Table 4). Note that by varying the number of simultaneous transmitters, we vary network contention and consequently collision rate. In order to observe the impact of data rates on performance, we ran this experiment with data rates of 54-, 24-, 11-, 5.5-, or 2 Mbps, while signaling transmission rate is kept at 2 Mbps.

Fig. 3 shows SACA's average throughput compared against IEEE 802.11 DCF's base mode ("Basic") and IEEE 802.11's DCF congestion avoidance mode with different RT values. We observe that SACA outperforms all other approaches for all data rates. As expected, for lower data rates, "Basic" has the lowest performance because frame transmission takes longer and therefore collision rate is higher. At lower data rates, RTS/CTS enabled with lower RT values provides better performance. As the data rate increases, performance of $Th = 0$ (i.e., RTS/CTS enabled all the time) improves and eventually outperforms "Basic". For example, for 24 Mbps, RTS/CTS with mid-range RT values provide better performance than both "Basic" and RTS/CTS with larger RT values. The reason is that throughput improves when larger frames are "protected" by RTS/CTS at this data rate while, for smaller frames, throughput is higher without RTS/CTS. Our proposed algorithm performs consistently well because it can dynamically decide when to enable or disable RTS/CTS based on current conditions, i.e., frame size, transmission rate, and contention. Note that the 95% confidence intervals confirm that our simulations have reached stationary behavior.

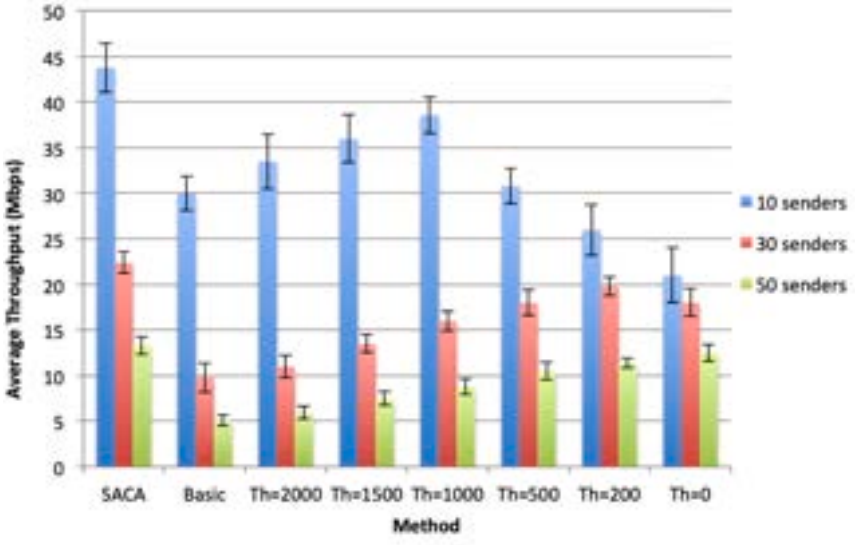
Table 4 shows how one of the senders uses RTS/CTS during the experiment as contention and frame size change every 5 s. These results confirm that, in addition to frame size and network contention, data transmission rate plays an important role in determining whether RTS/CTS should be used or not. For example, in seconds 10 to 15, where frames are 2000 bytes and collision rate is 26%, when data transmission rate is 54- and 24 Mbps, RTS/CTS is disabled. However, at 11-, 5.5-, and 2 Mbps, RTS/CTS is used. This is because, for lower data rates, using RTS/CTS is more advantageous.

Hot-spot trace. We ran similar simulations using the hot-spot trace with 10-, 30-, and 50 senders using *ns-3*'s IEEE 802.11n. Since there are 10 individual flows in our trace, each flow is assigned to 1-, 3-, and 5 senders in the 10-, 30-, and 50-sender scenarios, respectively, with each flow starting at slightly different start times. Similarly to the synthetic trace simulations, we compare SACA's average throughput against IEEE 802.11's base mode (no RTS/CTS), as well as statically configured RT values of 0 (RTS/CTS always enabled), 200-, 500-, 1000-, 1500-, and 2000 bytes. Fig. 4(a) show SACA's average throughput in infrastructure-based scenario with 10-, 30-, and 50 senders.

In the 10-sender scenario, since there is less contention, using IEEE 802.11's base mode or higher RT values is more beneficial. With 30 senders, which results in higher contention, using lower thresholds, e.g., 200- and 500 bytes, yields better performance. In the 50-sender scenario, RTS/CTS should be used all the time because of the high



(a) Hot-spot trace



(b) Company campus network trace

Fig. 4. Average throughput using hot-spot and company campus network traces in infrastructure-based scenario. 95% confidence intervals are shown.

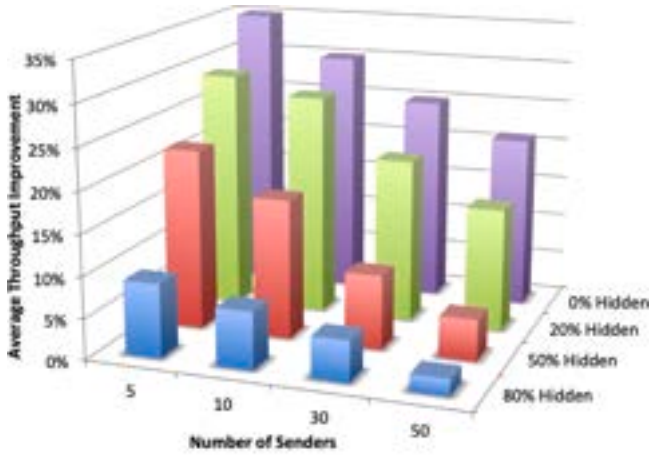
contention. In all cases, SACA outperforms all other methods because of its ability to automatically adjust to network contention and airtime. Note that, while in the 10-sender experiment, "Basic" yields similar throughput when compared to SACA, in the 50-sender scenario, "Basic" is the worst performer. In other words, SACA outperforms the best performer among the static methods in all cases.

Company campus trace. Similarly to the hot-spot trace, data rates provided in the Radiotap Header are used to calculate airtime. We ran simulations with 10-, 30-, and 50 senders by assigning each captured flow to 2-, 6- and 10 senders, respectively. Compared to the hot-spot trace, the average frame size and data rate are considerably higher. As shown in Fig. 4(b), SACA outperforms all other methods in all scenarios, which is consistent with the results observed in the hot-spot simulations. This is due to SACA's ability to dynamically adjust to frame size, transmission rate, and network contention. For instance, even though contention is not high in the 10-sender scenario, since the ratio of frame sizes to the data rates is larger on average, enabling RTS/CTS yields higher average throughput when compared to Basic. As the number of senders increases, the optimal *RT* value decreases. So in the 50-sender scenario, RTS/CTS should be used all the time.

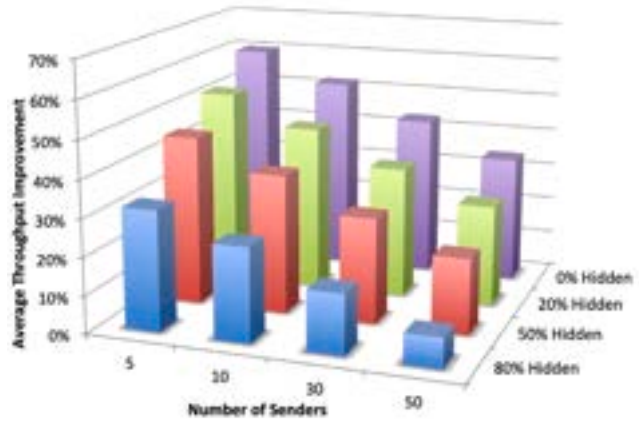
SACA's throughput improvement

To further explore SACA's performance, we ran simulations varying not only the number of senders but also the percentage of hidden terminals and measure SACA's throughput improvement.

Synthetic trace. The set of simulation experiments driven by the synthetic trace used two different data rates, lower and higher, namely, 2 Mbps and 54 Mbps. Fig. 5(a) and (b) show SACA's throughput improvement over IEEE 802.11 DCF's collision avoidance mode with RTS/CTS always on for 2- and 54 Mbps, respectively. As expected, when the number of senders and hidden nodes increase, the benefits of using RTS/CTS all the time increase. However, SACA is still performing better in all cases. For lower data rates (Fig. 5(a)), SACA's improvement

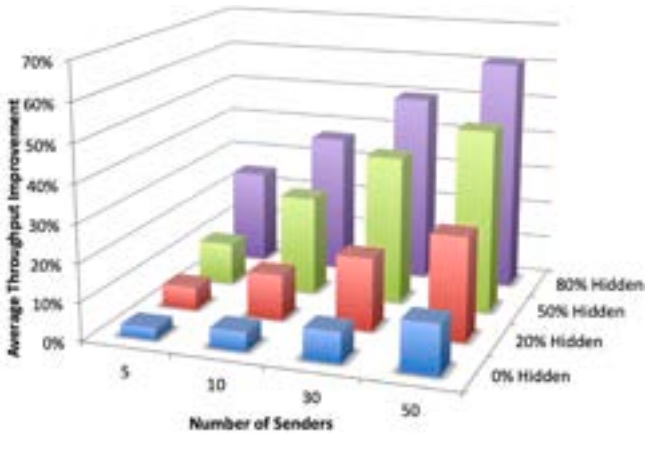


(a) 2 Mbps

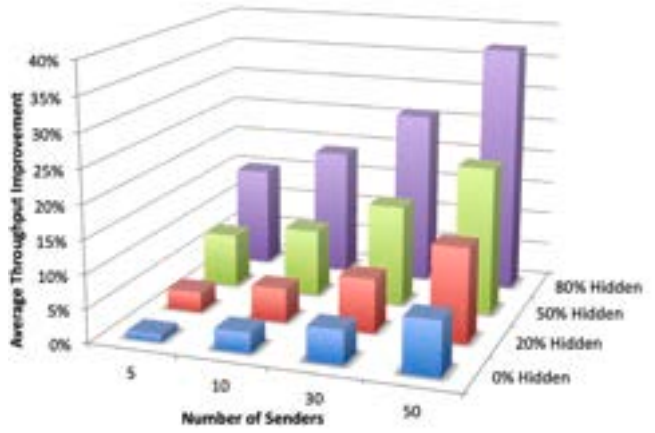


(b) 54 Mbps

Fig. 5. SACA's average throughput improvement compared to IEEE 802.11 DCF's congestion avoidance mode (RTS/CTS ON) for synthetic trace.



(a) 2 Mbps



(b) 54 Mbps

Fig. 6. SACA's average throughput improvement compared to IEEE 802.11 DCF's base mode (RTS/CTS OFF) for synthetic trace.

is not as pronounced compared to higher data rate. The reason is that, at higher data rates (Fig. 5(b)), the cost of using RTS/CTS all the time is relatively more expensive since control frames are sent at lower data rate (i.e., 2 Mbps).

We also evaluate SACA's throughput improvement over IEEE 802.11 DCF's base mode (i.e., no RTS/CTS) for data rates of 2- and 54 Mbps. is getting more by increasing the number of nodes and hidden nodes. Fig. 6(a) and (b) show almost the exact inverse behavior when compared to Fig. 5(a) and (b). In other words, SACA's throughput improvement is more accentuated at lower data rates when compared to higher data rates. This is because SACA, in some cases, enables RTS/CTS and RTS/CTS' cost is lower at lower data rates.

Hot-spot trace. We conducted similar simulations using the hot-spot trace and evaluated SACA's improvement over IEEE 802.11 DCF's congestion avoidance mode (RTS/CTS on) and IEEE 802.11 DCF's base mode (RTS/CTS off). From Fig. 7(a), we observe that SACA performs much better than IEEE 802.11 DCF's congestion avoidance mode when chance of collision is low. However, by increasing the number of senders and hidden nodes, SACA's throughput improvement is less pronounced. Fig. 7(b) shows SACA's throughput gain over IEEE 802.11 DCF's base mode (RTS/CTS off). We observe that as network contention and number of hidden nodes increase, so does SACA's average throughput improvement over RTS/CTS off. The reason is that SACA turns on the RTS/CTS dynamically when contention is high, which decreases the collision rate and, as a result, improves overall throughput performance. In all cases, SACA outperforms both IEEE 802.11 DCF's base- and congestion avoidance modes.

Company campus network trace. We observe similar behavior for the company campus network trace. As shown in Fig. 8(a), when there is less contention and no hidden nodes, there is no need to use RTS/CTS. Therefore, SACA disables RTS/CTS and, as a result, throughput goes up compared to having RTS/CTS on all the time. Although the performance of both techniques are closer in higher contention scenarios, SACA still performs better since it switches RTS/CTS off considering airtime in addition to contention. From Fig. 8(b), we observe that, since RTS/CTS is always off, in scenarios that exhibit higher contention and number of hidden nodes, SACA's throughput improvement is more pronounced, while SACA's throughput improvement decreases in scenarios with lower contention and lower number of hidden nodes.

SACA's network contention adaptation

In this section, we examine SACA's ability to adapt to network contention in more detail. More specifically, we investigate how well SENSE estimates contention based on its collision rate measurements and then how SACA uses that information to enable/disable RTS/CTS.

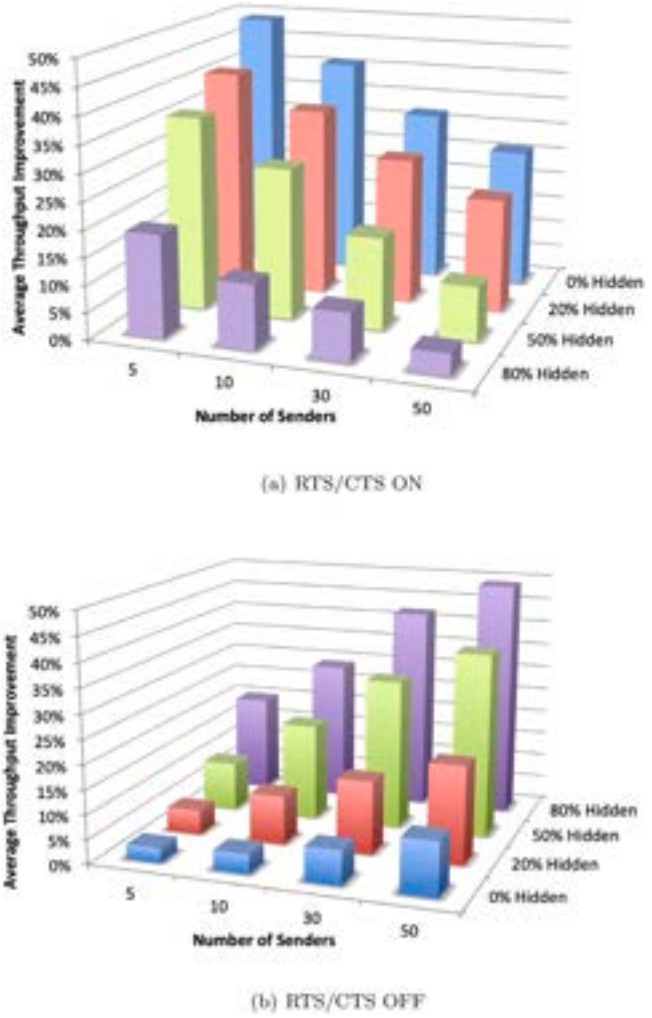


Fig. 7. SACA's average throughput improvement for hot-spot data.

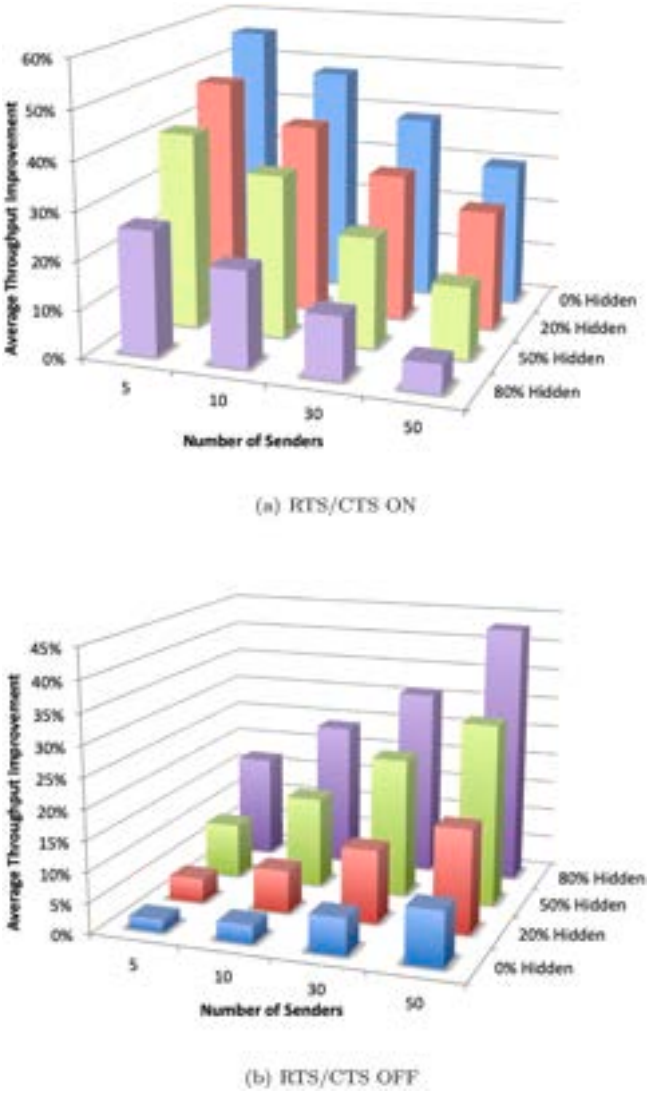


Fig. 8. SACA's average throughput improvement for company campus data.

To do that, we collect collision rate measurements as well as SENSE's collision rate estimates (see Section 3.1). We collect data at an AP with 20 senders transmitting according to the public hot-spot and company network datasets. Collision rates were collected with RTS/CTS disabled in order to measure "raw" network contention.

Based on SENSE's collision rate estimates, SACA decides whether to enable/disable RTS/CTS. Then, we also show the AP's RTS/CTS usage time series, where "1" indicates that RTS/CTS is enabled and "0" that RTS/CTS is disabled. We ran simulations for two scenarios, namely when all nodes are hidden from each other when all of them are exposed.

Company campus network trace. For the company campus trace, Fig. 9(a) shows the collision rate and SENSE's collision rate estimates at the AP when all nodes are hidden from each other. As expected, in the presence of hidden nodes, collision rates can be quite high (in this case, as high as 90%) and we observe that SENSE's estimates are able to follow real collision rate measurements quite closely with a mean squared error (MSE) of 0.2%. Because of the relatively high average collision rate, as shown in Fig. 9(b), RTS/CTS usage in the company network trace is triggered around 40% of the time.

In Figs. 10(a) and 10(b), respectively, we show collision rate measurements, SENSE's collision rate predictions, and RTS/CTS usage for the company campus trace where all nodes can see each other. Again, SENSE's predicted collision rate follows real collision rate measurements quite closely with a mean squared error (MSE) of 0.7%. As expected, collision rates are considerably lower when compared to the case where nodes are hidden (Fig. 9(a)) and, as a result, RTS/CTS is used less (around 8% of the time).

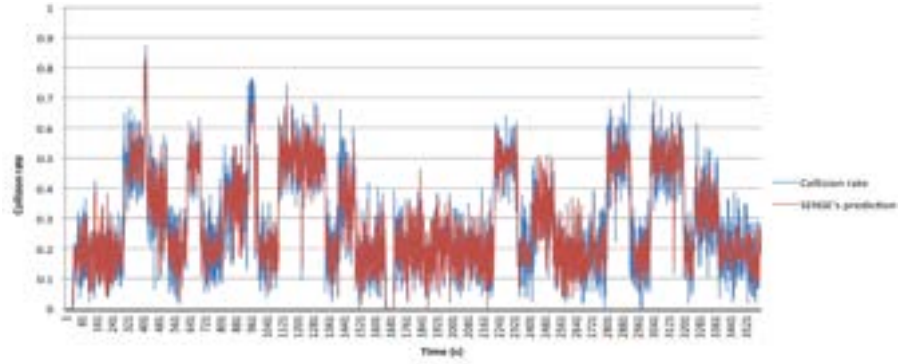
We should point out that we ran similar simulations using the hot-spot traffic trace and observed similar behavior to the company campus trace. These results are not shown due to space limitations.

5.2. Ad-Hoc network scenarios

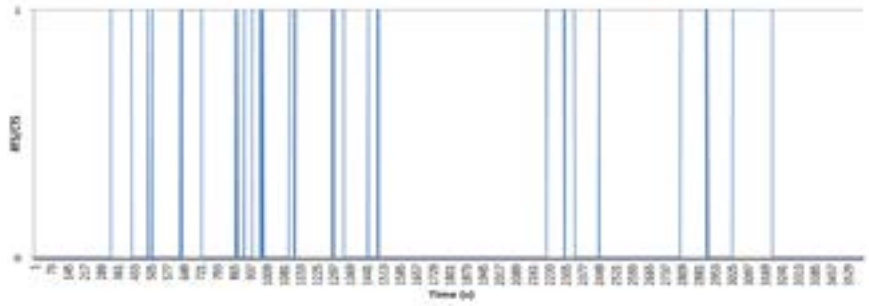
Simulation experiments similar to the ones conducted for infrastructure-based topologies were run for ad-hoc scenarios using the parameters summarized in Table 2. As previously noted, unlike the infrastructure-based simulations which need not use routing, ad-hoc scenarios employed the AODV routing protocol [18].

Average throughput

We compare SACA's average throughput against that of IEEE 802.11's DCF base mode (RTS/CTS always off) and congestion avoidance mode (RTS/CTS always on) in a variety of ad-hoc scenarios. We also show average throughput when using statically configured values for the RTS Threshold, RT, namely 200-, 500-, 1000-, 1500-, and 2000 bytes.

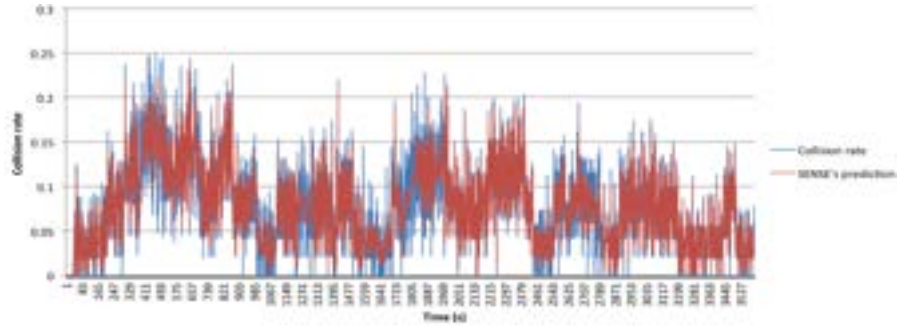


(a) Collision rate and SENSE's prediction with a mean squared error (MSE) of 0.2%)

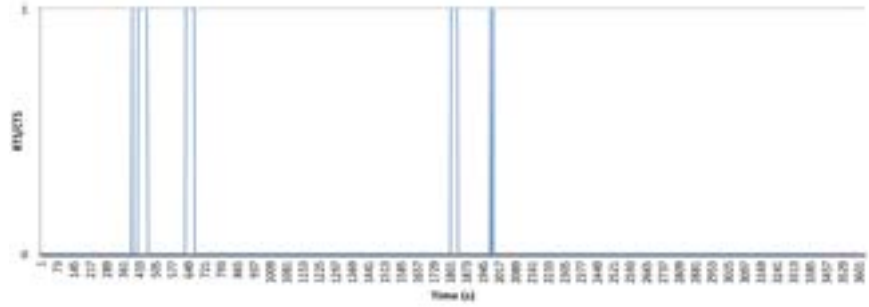


(b) AP RTS/CTS usage

Fig. 9. Collision rate measurements, SENSE's collision rate prediction, and RTS/CTS usage for AP in company campus network trace when all nodes are hidden from one another.



(a) Collision rate and SENSE's prediction with a mean squared error (MSE) of 0.7%)



(b) AP RTS/CTS usage

Fig. 10. Collision rate measurements, SENSE's collision rate prediction, and RTS/CTS usage for AP in company campus network trace when all nodes are exposed to one another.

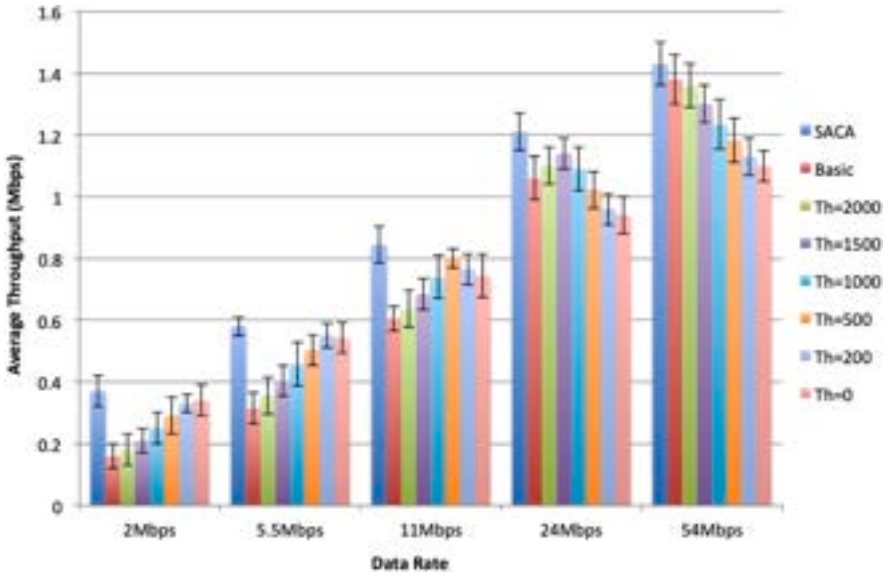


Fig. 11. Average throughput using the synthetic trace in ad-hoc scenario.95% confidence intervals are shown.

Synthetic trace. To show how our algorithm adjusts to network dynamics, we periodically changed frame size and number of nodes transmitting in our synthetic data trace. Note that by varying the number of simultaneous transmitters, we vary network contention and consequently collision rate. We ran each experiment with data transmission rates of 54-, 24-, 11-, 5.5-, and 2 Mbps, while signaling transmission rate is kept at 2 Mbps. We ran each experiment 10 times randomly selecting senders and their receivers.

Fig. 11 shows SACA's average throughput as well as that of IEEE 802.11 DCF's base mode ("Basic"), and RTS/CTS enabled based on different *RT* values. Consistent with results from infrastructure-based scenarios, we observe that our algorithm outperforms all other methods tested for all data rates used. As expected, for lower data rates, "Basic" has the lowest performance because frame transmission takes longer and therefore collision rate is higher. In the low data rate cases, lower *RT* values provide better performance. As the data rate increases, performance of "Basic" (RTS/CTS disabled all the time) improves and eventually outperforms "Th = 0". For example, for 24 Mbps, RTS/CTS with mid-range *RT* values provide better performance than both "Basic" and RTS/CTS with larger *RT* values. The reason is that throughput improves when larger frames are "protected" by RTS/CTS at this data rate while, for smaller frames, throughput is higher without RTS/CTS. SACA always performs well because it can dynamically decide when to enable or disable RTS/CTS based on current conditions, i.e., frame size, transmission rate, and contention.

Table 5 shows the usage of RTS/CTS at a specific node as contention and frame size changes every 5 s. It confirms that, in addition to frame size and network contention, data transmission rate plays an important role as well. For example, in seconds 10 to 15, when frames are 2000 bytes and collision rate is 21%, at 54 and 24 Mbps, RTS/CTS is disabled. However, in 11, 5.5 and 2 Mbps, RTS/CTS is used. This is because, for lower data rates, using RTS/CTS turns out to be more advantageous.

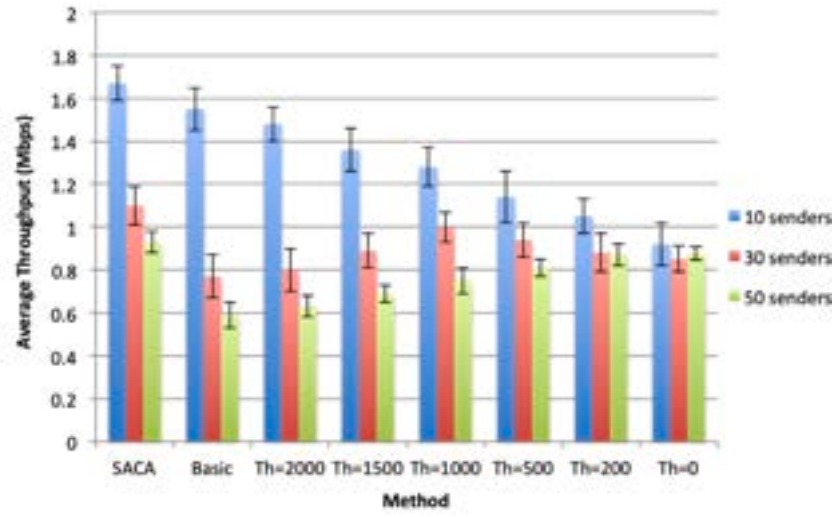
Hot-spot trace. For the hot-spot trace, we employed a similar strategy as in the infrastructure-based scenarios. In order to vary network contention, we used 10-, 30-, and 50 senders and assigned each flow to 1-, 3-, and 5 senders, respectively. Flows have slightly different start times. Similarly to the synthetic trace simulations, we compare the average throughput when using SACA against IEEE 802.11's base mode (no RTS/CTS), as well as statically configured *RT* values of 0- (RTS/CTS always enabled), 200-, 500-, 1000-, 1500-, and 2000 bytes.

Table 5
SACA's slot-by-slot behavior using the synthetic trace with different data rates in the ad-hoc scenario illustrating how one of the sending nodes dynamically switches between Basic (B) and RTS/CTS (R/C) modes based on air time and perceived congestion.

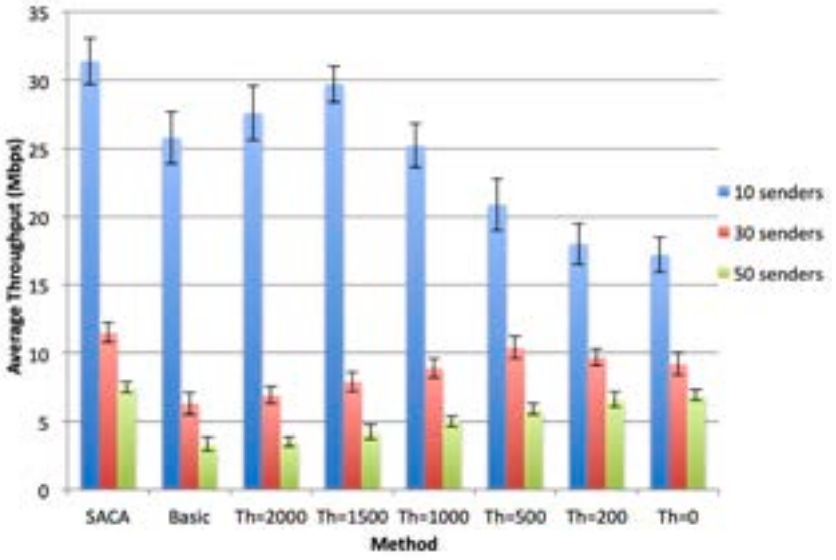
Time (s)	Frame size (bytes)	Collision	54 Mbps	24 Mbps	11 Mbps	5.5 Mbps	2 Mbps
0 to 5	1500	2%	B	B	B	B	B
5 to 10	500	7%	B	B	B	B	R/C
10 to 15	2000	21%	B	B	R/C	R/C	R/C
15 to 20	200	33%	B	B	B	B	R/C
20 to 25	1000	39%	B	B	R/C	R/C	R/C
25 to 30	2000	52%	R/C	R/C	R/C	R/C	R/C
30 to 35	500	59%	B	B	R/C	R/C	R/C
35 to 40	200	63%	B	B	B	R/C	R/C
40 to 45	1500	67%	R/C	R/C	R/C	R/C	R/C
45 to 50	500	71%	B	B	R/C	R/C	R/C

As shown in Fig. 12(a), in the 10-sender scenario, since there is less contention, using the base mode ("Basic") or higher *RT* values is more beneficial. By adding more nodes and, as a result, increasing contention, lower RTS thresholds like 200- and 500 bytes perform better. In the 50-sender scenario, RTS/CTS should be used all the time because of the high contention. In all cases, SACA outperforms all other methods because of its ability to adjust to network contention and airtime. While in the 10-sender experiment, "Basic" yields similar throughput when compared to our approach, in the 50-sender scenario, "Basic" is the worst performer. In other words, SACA can beat the best static method in all cases.

Company campus network trace. Similarly to the hot-spot trace, data rates provided in the Radiotap Header are used to calculate the airtime in the company campus network trace. We ran simulation experiments with 10-, 30-, and 50 senders by assigning each captured flow to 2-, 6- and 10 senders, respectively. Compared to the public hot-spot trace, average frame sizes and data rates are considerably higher in this dataset (see Table 3). As shown in Fig. 12(b), in all scenarios, SACA outperforms all other methods since it adjusts dynamically to frame size, transmission rate, and network contention. For instance, even though contention is not high in the 10-node scenario, since frames are larger on average, enabling RTS/CTS yields higher average throughput when compared to "Basic". As the number of nodes increases, the



(a) Hot-spot trace



(b) Company campus trace

Fig. 12. Average throughput using hot-spot and company campus traces in ad-hoc scenario. 95% confidence intervals are shown.

optimal *RT* value decreases. So in the 50-node scenario RTS/CTS should be used all the time.

6. Related work

Improving the performance of IEEE 802.11 has been the subject of a considerable body of work from both academia and industry. Due to space limitations, in this section we present a brief overview of research efforts directly related to our work on dynamically enabling/disabling IEEE 802.11's congestion avoidance mechanism.

We categorize these efforts in two groups. The first group investigates throughput performance when using or not using RTS/CTS and argue that the problems introduced by the RTS/CTS mechanism tend to counterbalance its benefits. For example, the work reported in [19] shows that in some situations, the interference range is much larger than the transmission range, which prevents the RTS/CTS mechanism to completely prevent interference. In [20], the maximum throughput in IEEE 802.11 DCF networks with RTS/CTS enabled is evaluated and argues that as the number of RTS/CTS control frames increases, RTS/CTS collisions occur more frequently. In [13] and [21], RTS/CTS performance was evaluated under different data rates and concluded that RTS/CTS does not show much benefit at higher data rates.

The second group examines performance under different values of the RTS Threshold, *RT*. For instance, in [22], it was shown that, when the network is under "stress", e.g., high node density, high traffic load, the value of *RT* can significantly impact performance in terms of end-to-end delay, medium access delay, retransmission attempts, network load, and throughput. Other efforts explore how to enable/disable RTS/CTS dynamically. The work in [15] proposes a mechanism to count the number of "Waiting for CTS" timeouts. If that number goes

above a certain threshold, RTS/CTS is enabled; otherwise, it is disabled. In [12], an adaptive RTS/CTS control method based on the existence of hidden terminals is introduced and uses a fixed threshold to determine whether to switch RTS/CTS on or off. The algorithm proposed in [14] dynamically adjusts *RT* based on frame delivery ratio and shows performance improvements over both IEEE 802.11 base mode and collision avoidance mode. In [23], *RT* is set based on recently observed frame size distribution in the network. *RT* is set to a value such that some specific percentage of frames size's fall below its value. In [13], *RT* is adjusted dynamically according to the data rate and number of stations. The work reported in [24] proposes an analytical expression deriving the optimal RTS threshold based on 5 parameters, namely the number of nodes, the transmission rate, the basic rate, the initial backoff window size, and finally the cutoff phase. The latter four parameters were assumed to be constant while the number of nodes which would be reported by the Access Point.

To the best of our knowledge, SACA is the first approach to use the combination of air time and local network congestion to automatically and dynamically enable or disable the RTS/CTS handshake. In our prior work [4], time is divided into slots and network contention is evaluated only at the beginning of a slot. Then, decision to disable/enable RTS/CTS is made based on the estimated collision and transmission air time. In SACA, we use a more general contention assessment approach which measures collisions periodically over time, allowing us to capture network contention fluctuations more accurately. Additionally, unlike our previous approach, which only considers the cost of data collisions, SACA also accounts for the overhead incurred by RTS/CTS collisions.

7. Conclusion

This work proposed and evaluated a novel algorithm that employs machine learning to dynamically decide whether to enable or disable the RTS/CTS handshake used by IEEE 802.11 DCF's congestion avoidance mode to reserve the communication channel ahead of data transmission. The proposed Smart Adaptive Collision Avoidance algorithm, or SACA for short, continuously learns current networking conditions, namely air time, i.e. the ratio between the size of data/control information being transmitted and transmission rate, and network contention to compare the cost between using RTS/CTS or retransmitting data, and dynamically switches RTS/CTS on and off accordingly. This work builds on our previous work [4] and goes a step further by making three main contributions as follows. The first one is that SACA's current approach continuously tracks network conditions rather than relearn them periodically from scratch. The second one is that SACA's new learning mechanism employs a refined cost function that reflects the overhead of data and RTS/CTS transissions more accurately. Finally, to validate SACA, we evaluate it on a variety of WLAN as well as wireless multi-hop ad-hoc network scenarios, driven by synthetic and real traffic traces. Our simulation results demonstrated that SACA consistently outperforms current best practices, such as never enabling RTS/CTS or using a pre-specified threshold to decide whether to switch RTS/CTS on or off.

As part of future work, we will evaluate SACA using data rate adaptation algorithms, in particular the ones currently available in the *ns-3* simulator (e.g., ARF, AARF, CARA, and RRAA). We also plan to explore how SACA can be applied to improve the performance of other aspects and variants of the IEEE 802.11, as well as protocols at other layers of the network stack, including routing and transport.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been partially supported by the US National Science Foundation under grant CNS 1321151.

References

[1] P. Karn, Maca-a new channel access method for packet radio, in: ARRL/CRRL Amateur Radio 9th Computer Networking Conference, Vol. 140, 1990, pp. 134–140.
[2] E. Perahia, R. Stacey, Next Generation Wireless LANs: 802.11 N and 802.11 Ac, Cambridge University Press, 2013.
[3] G. Bianchi, Performance analysis of the IEEE 802.11 distributed coordination function, IEEE J. Sel. Areas Commun. 18 (3) (2000) 535–547.
[4] Y. Edalat, K. Obraczka, B. Amiri, A machine learning approach for dynamic control of RTS/CTS in WLANs, in: Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, ACM, 2018, pp. 432–442.
[5] L. Kleinrock, F. Tobagi, Packet switching in radio channels: Part I-carrier sense multiple-access modes and their throughput-delay characteristics, IEEE Trans. Commun. 23 (12) (1975) 1400–1416.
[6] Y. Wang, J.J. Garcia-Luna-Aceves, Collision avoidance in multi-hop ad hoc networks, in: Proceedings. 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, IEEE, 2002, pp. 145–154.
[7] G.F. Riley, T.R. Henderson, The ns-3 network simulator, in: Modeling and Tools for Network Simulation, Springer, Berlin, Heidelberg, 2010, pp. 15–34.
[8] Y. Edalat, J.S. Ahn, K. Obraczka, Smart experts for network state estimation, IEEE Trans. Netw. Serv. Manag. 13 (3) (2016) 622–635.
[9] J.S. Hunter, The exponentially weighted moving average, J. Qual. Technol. 18 (4) (1986) 203–210.
[10] M. Herbster, M.K. Warmuth, Tracking the best expert, Mach. Learn. 32 (2) (1998) 151–178.
[11] B.A.A. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, K. Obraczka, A machine learning framework for TCP round-trip time estimation, EURASIP J. Wireless Commun. Networking (1) (2014) 47.
[12] T. Shigeyasu, M. Akimoto, H. Matsuno, Throughput improvement of ieee 802.11 dcf with adaptive rts/cts control on the basis of existence of hidden terminals, in: 2011 International Conference on Complex, Intelligent, and Software Intensive Systems, IEEE, 0000, pp. 46–52.
[13] Z.N. Kong, D.H. Tsang, B. Bensaou, Adaptive RTS/CTS mechanism for IEEE 802.11 WLANs to achieve optimal performance, in: 2004 IEEE International Conference on Communications. (IEEE Cat. No. 04CH37577), Vol. 1, 2004, pp. 185–190.
[14] M. Mjidi, D. Chakraborty, N. Nakamura, K. Koide, A. Takeda, N. Shiratori, A new dynamic scheme for efficient RTS threshold handling in wireless networks, in: 22nd International Conference on Advanced Information Networking and Applications (AINA 2008), IEEE, pp. 734–740.
[15] L.H.J. Ju, I. Rubin, The effect of disengaging RTS/CTS dialogue in IEEE 802.11 MAC protocol, in: International Conference on Wireless Networks, 2003, pp. 632–638.
[16] B. Kalantari-Sabet, M. Mjeku, N.J. Gomes, J.E. Mitchell, Performance impairments in single-mode radio-over-fiber systems due to MAC constraints, J. Lightwave Technol. 26 (15) (2008) 2540–2548.
[17] B. Nunes, K. Veenstra, W. Ballenthin, S. Lukin, K. Obraczka, A machine learning framework for TCP round-trip time estimation, EURASIP J. Wireless Commun. Networking 2014 (1) (2014) 47.
[18] C. Perkins, E. Belding-Royer, S. Das, Ad hoc on-demand distance vector (AODV) routing, in: IETF RFC 3561, 2003.
[19] K. Xu, M. Gerla, S. Bae, How effective is the IEEE 802.11 RTS/CTS handshake in ad hoc networks, in: GlobeCom, Vol. 2, 2002, pp. 72–76.
[20] T. Sugimoto, N. Komuro, H. Sekiya, S. Sakata, K. Yagyu, Maximum throughput analysis for RTS/CTS-used IEEE 802.11 DCF in wireless multi-hop networks, in: International Conference on Computer and Communication Engineering (ICCCE'10), IEEE, 0000, pp. 1–6.
[21] I. Tinnirello, S. Choi, Y. Kim, Revisit of RTS/CTS exchange in high-speed IEEE 802.11 networks, in: Sixth IEEE International Symposium on a World of Wireless Mobile and Multimedia Networks, IEEE, 2005, pp. 240–248.
[22] H. Singh, A. Kaur, A. Sharma, V. Sharma, Performance optimization of DCF-MAC standard using enhanced RTS threshold under impact of IEEE 802.11 n WLAN, in: 2015 Fifth International Conference on Advanced Computing & Communication Technologies, IEEE, 2015, pp. 421–424.

[23] S.R. Ahsan, M.S. Islam, N. Hassan, A. Rahman, Packet distribution based tuning of RTS threshold in IEEE 802.11, in: The IEEE Symposium on Computers and Communications, 2010, pp. 1–6.

[24] Y. Yin, Y. Gao, S. Manzoor, X. Hei, Optimal RTS Threshold for IEEE 802.11 WLANs: Basic or RTS/CTS?, in: 2019 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation, 0000.

Yalda Edalat received Ph.D. and M.Sc. from San Jose State University and University of California Santa Cruz, in 2010 and 2019, respectively. Her research interests center around the application of computational intelligence techniques to computer networks and in particular wireless networks. She is currently teaching computer engineering courses at San Jose State University.

Katia Obraczka is a Professor of Computer Science and Engineering at the University of California, Santa Cruz. Prof. Obraczka's research interests span the areas of computer networks, distributed systems, and Internet information systems. Her lab, the Internetwork Research Group (i-NRG) at UCSC, conducts research on designing and developing protocol architectures motivated by the internets of the future. She has been a PI and a co-PI in a number of projects sponsored by government agencies (e.g., NSF, DARPA, NASA, ARO, DoE, AFOSR) as well as industry (e.g., Cisco, Google, Nokia). She is a Fellow of the IEEE.

Jong-Suk Ahn received the Ph.D. and M.S. degrees in Electrical Engineering from the University of Southern California in 1995 and from the Korean Advanced Institute of Science and Technology in 1985, respectively. He also received the B.S. degree from the Seoul National University in 1983. He was a visiting researcher in ISI/USC in 2001 and in the University of California in Santa Cruz (UCSC) in 2013. He was awarded the Gaheon Prize for the best paper in 2003 from the Korean Information Science Society. He is currently a professor at Dongguk University in Korea. His major interests are sensor networks, dynamic error control and flow control algorithms over the wireless Internet, network simulation techniques, IEEE 802.11, and IEEE 802.15.4.