Testbed Evaluation of Optimized REACT over Multi-hop Paths

Matthew J. Mellott[1](✉), Charles J. Colbourn[1], Violet R. Syrotiuk[1], and Ilenia Tinnirello[2]

[1] School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, P.O. Box 878809, Tempe, AZ 85287, USA
`{mmellott,colbourn,syrotiuk}@asu.edu`

[2] Department of Electrical Engineering, University of Palermo, Viale delle Scienze, Parco D'Orleans, 90128 Palermo, Italy
`ilenia.tinnirello@unipa.it`

Abstract. REACT is a distributed resource allocation protocol that computes a max-min allocation of airtime for mesh networks. The allocation adapts automatically to changes in local traffic load and in local network views. SALT, a new contention window tuning algorithm, ensures that each node secures the airtime allocated to it by REACT. REACT and SALT are extended to the multi-hop flow scenario with the introduction of a new airtime reservation algorithm. With a reservation in place, multi-hop TCP flows show increased throughput when running over SALT and REACT compared to running over 802.11 DCF. All results are obtained from experimentation on the `w-iLab.t` wireless network testbed in Belgium.

1 Introduction

Wi-Fi network performance is known to degrade dramatically when node density is high, and when flows are sustained over multiple hops. These conditions arise when multiple networks coexist [10,21] and when large access infrastructure is deployed [3,17]. The degradation results from the starvation and unfairness associated with carrier sense multiple access (CSMA) based protocols. This is attributed to a mismatch in the local views of the wireless medium among the nodes, and due to high levels of contention when the network is congested [12].

To mitigate such problems several approaches have been proposed. These include adopting rate limiters on nodes [4,7], using multi-hop reservations [16], using different access priorities for data and control traffic [8], and exploiting admission control [22].

Airtime measures the channel time in which a link is sensed busy because of frame transmissions. Airtime measurements are position-dependent, because channel attenuation differs for each transmitter-receiver pair. Airtime has been applied in routing in mesh networks [7,9], and in admission control [18].

This paper makes two contributions. First, we use a measurement driven approach to control the airtime allocated to each node in a wireless network

https://doi.org/10.1007/978-3-030-02931-9_11

computed by the REACT protocol [19]. REACT negotiates airtime allocations among the nodes on the basis of traffic requirements and local views of the network. In order to achieve its allocated airtime we develop SALT, a mechanism for dynamically tuning the contention window of each node.

There has been much work on contention window tuning but with different objectives. Among the first may be MACAW [1], replacing the binary exponential backoff with a multiplicative increase and linear decrease (MILD) of the contention window (CW) size to improve fairness. Other work tunes the CW to achieve a theoretical throughput limit [6]. Our goal here is different: Our aim is to tune the contention window at each node to realize a specific airtime allocation.

REACT and SALT are both compatible with the 802.11 standard [15] and have been implemented on legacy devices. Extensive experimentation in the `w.i-Lab.t` wireless network testbed [5] shows that the tuning approach is able to align allocations to those negotiated.

The second contribution is an extension of REACT to reserve an allocation along the path of a multi-hop TCP flow. This requires that neighbours of the nodes along the forwarding path take the reservation into account as part of their own allocations so that they do not interfere with the multi-hop flow. Multi-hop wireless mesh networks present many challenges for TCP. In addition to the unreliable wireless transmission at each hop, contention from hidden and exposed nodes in a wireless network constrain the TCP throughput achievable over a multi-hop path [11]. To the best of our knowledge, Gupta et al. [14] is one of the few works conducting experimentation with TCP in a physical wireless network. However, their emphasis is on how TCP throughput is affected by routing, user mobility, and the number of hops in the network rather than on the impact of the MAC protocol. While no performance advantages for TCP were found in their results [14], we have achieved higher TCP throughput in running the flow over REACT combined with SALT, than over 802.11.

The rest of this paper is organized as follows. In Sect. 2, we describe the REACT protocol for negotiating channel airtime. Section 3 presents SALT, a new tuning algorithm, its implementation in legacy commercial Wi-Fi cards, and an evaluation of how well it achieves the airtime allocation. Section 4 provides an algorithm to reserve airtime for a multi-hop flow over the REACT/SALT framework and evaluates the algorithm using a multi-hop TCP flow. Finally, we summarize and propose future work in Sect. 5.

2 Realizing a REACT Allocation

REACT is a distributed resource allocation protocol that uses the metaphor of an auction [19]. When used in the context of mesh wireless networks, the resource being allocated (or put up for "auction") is *airtime*, the percentage of time a node controls the medium over a given period. Each node runs an auctioneer and a bidder algorithm concurrently; auctioneers offer capacity while bidders claim capacity at adjacent auctions to satisfy their own airtime demand. Auctioneers update their offers to satisfy all nodes bidding at their auction while also ensuring that all nodes receive a fair allocation of the resource. Bidders update their claims to ensure that they are not consuming any more airtime than can be offered at any adjacent auction. Nodes participating in REACT converge to a lexicographic max-min airtime allocation [19]. A change in the local network view or traffic load triggers REACT to run and adapt the allocation.

Lutz et al. [19] realize the REACT allocation in a schedule-based MAC protocol, in which a number of transmission slots at each node are selected at random to correspond to its allocation. A node can recompute its schedule immediately upon receiving a new allocation from REACT, rather than waiting for the end of a frame, making it competitive with contention-based protocols. Their initial evaluation was conducted in simulation where synchronization is not a challenge.

Hence, Garlisi et al. [13] instead realize the REACT allocation in a contention-based MAC protocol. Because the channel access probability depends on the average contention window (CW) size [2], a node's REACT allocation is realized by tuning the CW size. A legacy Wi-Fi node i can estimate its current allocation s_i as a function of the total number n_i of channel accesses it makes, the total time F_i that its backoff is frozen, and the total airtime A_i in an observation interval C:

$$s_i = \frac{A_i}{C} = \frac{A_i}{A_i + F_i + W_i/2 \cdot \sigma \cdot n_i} \tag{1}$$

where $W_i/2 \cdot \sigma \cdot n_i$ is an approximation of the total time required for the backoff countdown. This suggests how the contention window can be tuned.

3 REACT Implementation with SALT Tuning

Different from [13], rather than modify 802.11 packet headers, we send control messages to implement REACT periodically. The time period must be longer than the amount of time it takes to update bids and offers. With this method the overhead for control traffic does not increase when the data rate increases.

Our implementation of REACT is paired with a new contention window tuning approach, described next.

3.1 Smoothed Airtime Linear Tuning (SALT)

Smoothed Airtime Linear Tuning (SALT) is a new contention window tuning technique. As in [13], the contention window size is *fixed* unless and until a new airtime is allocated to node i by REACT. The intuition is that a node's channel access behaviour should depend on its allocation, not on the packet outcome.

SALT measures airtime a_i^t at node i over observation interval t and uses it to set W_i^{t+1}, the contention window size for the next interval $t+1$. However a_i^t is not passed directly to the tuning component of SALT. Its value is first smoothed, using an exponentially weighted moving average in Eq. (2) with parameter

$0 \leq \beta \leq 1$, to produce S_i^t which is then used for tuning. Smoothing is done to reduce the effect of random background noise.

$$S_i^t = \begin{cases} a_i^1 & \text{if } t = 1 \\ \beta a_i^t + (1-\beta) S_i^{t-1} & \text{if } t > 1 \end{cases} \tag{2}$$

SALT's tuning component is given in Eq. (3); s_i^t is the airtime allocation for node i from REACT, k is a constant scaling factor, and the maximum CW size is 1024. The difference between S_i^t and s_i^t is scaled by k to convert the difference of unit-less airtime ratios into a contention window size value.

$$W_i^t = \begin{cases} 0 & \text{if } t = 0 \\ \lfloor S_i^t - s_i^t \rfloor k + W_i^{t-1} & \text{if } t > 0 \text{ and } 0 \leq \lfloor S_i^t - s_i^t \rfloor k + W_i^{t-1} < 1024 \\ 1023 & \text{if } t > 0 \text{ and } \lfloor S_i^t - s_i^t \rfloor k + W_i^{t-1} \geq 1024 \\ 0 & \text{if } t > 0 \text{ and } \lfloor S_i^t - s_i^t \rfloor k + W_i^{t-1} < 0 \end{cases} \tag{3}$$

3.2 SALT Implementation

SALT is implemented as a Python program running at user level on each Linux testbed node in an experiment. It is part of the same program that is running REACT, and the airtime allocation is passed from the thread running REACT to the thread running SALT. The airtime measurement interval used is one second. SALT is invoked after each of these one second intervals and uses data collected by the networking subsystem to determine the airtime during the last interval.

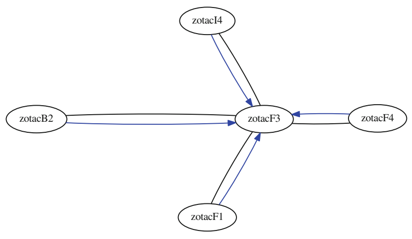
The Linux kernel is modified to expose the minimum W_{min} and maximum W_{max} contention window size to user level programs. The interface allows a user level program to set these parameters; we set $W_{min} = W_{max} = W_i^t$. The wireless subsystem is also patched to accept CW sizes that are not powers of two.

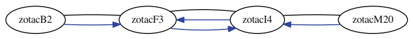
3.3 SALT Evaluation

To evaluate SALT, we use the IMEC advanced `w-iLab.t` testbed, located in Zwijnaarde, Belgium [5]. It is pseudo-shielded from external interference and is equipped with various wireless technologies, including IEEE 802.11, IEEE 802.15.4, Bluetooth dongles, Software Defined Radios (SDRs), LTE femto cells, among others. The `w-iLab.t` testbed uses the cOntrol Management Framework (OMF) for resource allocation, hardware and software configuration, and the orchestration of experiments. Measurement data are collected and stored in a central database over a wired control network for further processing.

Configuring wireless topologies in such an indoor controlled environment is important for benchmarking and for the reproducibility of the results. However, it is a non-trivial task, because the distance at which nodes are able to interfere can be much farther than the transmission range. In order to limit the physical visibility of the nodes, we use the 802.11a PHY, at the central frequency of 5180 MHz with a transmission power of 1 dBm.

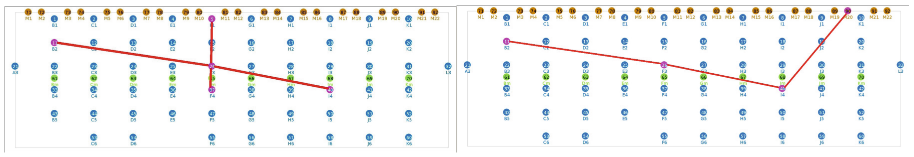
Tests are conducted to identify "zotac" nodes among the more than 90 available to match the logical topologies in Fig. 1. Each node is programmed to send





(a) Star topology. (b) Line topology.

Fig. 1. Logical testbed topologies used to evaluate SALT. Black lines correspond to bidirectional links while blue arrows denote single-hop flows. (Color figure online)



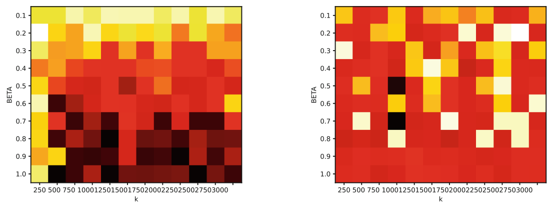
(a) Star topology. (b) Line topology.

Fig. 2. Logical topologies from Fig. 1 mapped onto physical nodes in `w-iLab.t`.

broadcast `ping`s in a dedicated time interval when all the other nodes are silent. Nodes that can decode the `ping` (`ICMP Echo Request` and `Response`) are neighbours connected by a bidirectional link. At the end of these tests, we build the network topologies in Fig. 1; see [20] for more topologies. The physical location of the nodes in `w-iLab.t` is shown in Fig. 2.

In each topology, each experiment conducted uses greedy UDP flows. These flows are set up with a target 1 Gbps UDP bandwidth, far beyond the capabilities of the wireless link (i.e., the channel is saturated).

Experiments to Select the Values of β and k. In order to determine values for β and k in Eqs. 2 and 3, we conduct an experiment where we varied their values over their range; we vary β from 0.1 to 1.0 in steps of 0.1 and k from 250 to 5000 in steps of 250, making for 120 trials on each of the topologies. Each trial lasts 15 s and we measure airtime for each node in the trial over that time. The heat maps in Fig. 3 show the convergence results for each topology and each combination of β and k. The darker the square in the heat map the faster the trial converged. We average the convergence time for each trial on each topology for the same β and k and select the lowest average. This results in a $\beta = 0.6$ and $k = 500$, for the best average convergence time of 7.44 s; these values of β and k are used in all subsequent experiments.



(a) Star topology. (b) Line topology.

Fig. 3. β versus k "heat maps" by topology. The darker the square in the heat map the faster the trial converged.

3.4 Experiment to Compare Tuning Algorithms

We conduct an experiment to compare SALT and the original tuning algorithm [13], running along with REACT; 802.11 DCF is included as a control. The flows in this experiment are greedy UDP flows set up as described in Sect. 3.3.

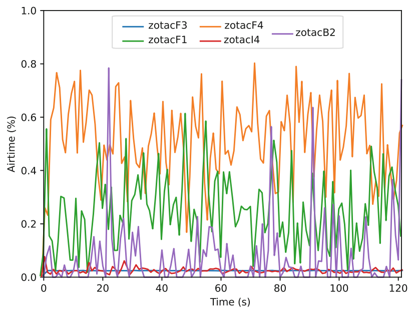
We present the results for the star topology in Figs. 4 and 5 because they are representative; see [20] for more detail. Figure 4 plots airtime as a function of time. As expected, the airtime of nodes running 802.11 is unequal and highly variable for each node. In this topology, the allocation computed by REACT for each node is 20%. The original tuning algorithm converges more quickly than SALT but not as tightly. Moreover, each node running SALT oscillates around the allocated airtime whereas in the original tuning algorithm some nodes are unable to reach their allocated airtime of 20%.

Figure 5 plots per-node throughput, jitter, drop rate, and aggregate throughput for the star topology. The high variability in airtime for 802.11 is reflected in the per-node results: The throughput, jitter, and drop rates from node to node are also highly variable. The tighter convergence of SALT leads to the higher throughput and lower jitter than the original tuning algorithm. Both tuning algorithms have a near zero drop rate while 802.11's drop rate is extremely high, well above 90% for three of the four nodes. Despite the high drop rate for 802.11, it still leads in aggregate throughput of 67.96 MiB; one node (zotacF4), which obtains around 60% of the airtime, is almost solely responsible for the higher aggregate throughput achieved. SALT has throughput of 55.57 MiB with the original tuning algorithm achieving 49.72 MiB.

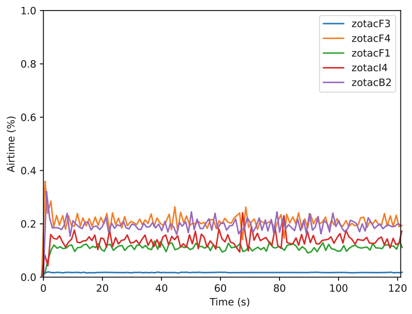
We now examine a more challenging multi-hop scenario.

4 Multi-hop REACT and Multi-hop Reservations

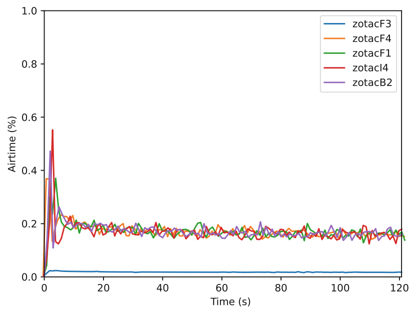
Until now, nodes running REACT have only taken into account their own traffic needs. In the auction a node's bid secures airtime for itself, but if there are multi-hop flows in a network this is insufficient because it does not take into account the fact that a node might need to forward traffic ultimately destined to other nodes. We present a multi-hop airtime reservation protocol that addresses this issue.



(a) 802.11.



(b) Original tuning [13].



(c) SALT.

Fig. 4. Airtime versus time for the star topology for 802.11, the original tuning algorithm, and SALT.

Without a reservation algorithm a node could try to predict how much airtime to reserve for multi-hop flows passing through it. Nodes store each of their neighbour's claims and could speculate how their demand should reflect the possibility of multi-hop flows. Unfortunately claims provide no information on the directionality of flows, multi-hop or not. A claim is sent to every node within broadcast range and only informs the receiver that the sender is currently expecting to utilize the amount of airtime claimed. Claims also do not tell a node anything about the demands of nodes beyond their one-hop neighbourhood, where the multi-hop flow could be originating. Multi-hop reservations allow the originators of multi-hop flows to inform nodes of the additional traffic they are expected to forward. Nodes along the reservation path can also inform the originator of resource saturation. The REACT auction itself is a convenient mechanism that can be used for the purpose of making these reservations.

Each auction in the REACT protocol allocates the channel capacity. In our multi-hop reservation algorithm, a reservation is made by reducing this capacity by the reservation amount at nodes along the path and at their neighbours. Once the reservation is placed nodes along the reservation path they each increase their

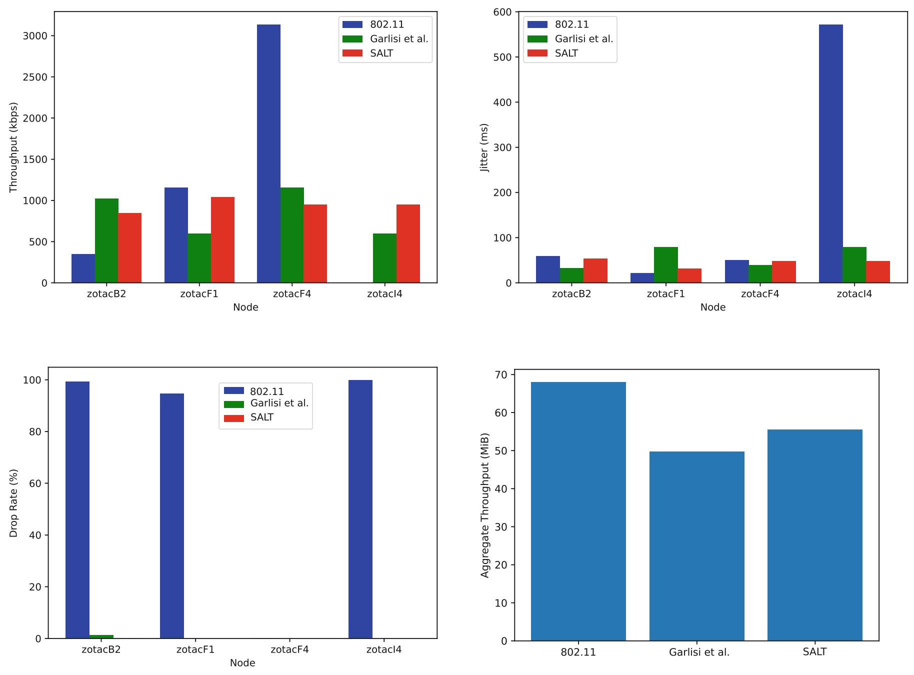


Fig. 5. Per-node throughput, jitter, and drop rate for each of 802.11 (in blue), the original tuning algorithm (in green), and SALT (in red), in addition to aggregate throughput, for the star topology. (Color figure online)

allocation by the reservation amount. This secures airtime for the flows that will be passing through the node while still maintaining the standard REACT auction for allocating airtime in the neighbourhood. Section 4.1 provides a more precise description of this process and Sect. 4.2 presents our evaluation of it.

4.1 Multi-hop Reservation Algorithm

The Reserve algorithm is recursive, with parameters s, r and d, where s and d are the source and destination of the multi-hop flow, requiring a reservation of r. First, the source reserves r for the multi-hop flow. If it is unable, the reservation fails. If it is successful, then it requests each of its one-hop neighbours, except the next-hop along the multi-hop path, to reserve r for the multi-hop flow. If any neighbour cannot reserve r it returns failure, causing the reservation to fail and recursively release the reserved resources. However, if all neighbours succeed in making a reservation, then Reserve is called recursively with the source equal to the next-hop node along the multi-hop path. If the next hop is the destination, then the reservation has succeeded at each node along the path, and the success propagates back to the source of the multi-hop flow. The pseudocode for the Reserve algorithm is provided in Algorithm 1.

Algorithm 1. RESERVE(s, r, d)

```
1:  if (capacity$_s$ − $r$) $\leq 0$ then
2:      return false
3:  end if
4:  capacity$_s$ $\leftarrow$ capacity$_s$ − $r$
5:  if ($s == d$) then
6:      return true
7:  else
8:      status $\leftarrow$ true
9:      for each $n \in$ (NEIGHBOURS($s$) \ NEXT_HOP($s$)) do
10:         status $\leftarrow$ status and NEIGHBOUR_HAS_CAPACITY($n, r$)
11:     end for
12:     if (status == true) then
13:         return RESERVE(NEXT_HOP($s$), $r$, $d$)
14:     else
15:         tear down any completed reservations made by neighbours of $s$ and $s$ itself
            for this multi-hop flow
16:     end if
17: end if
```

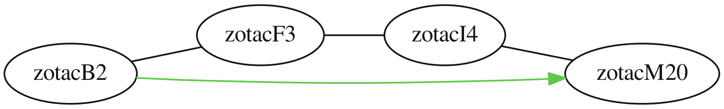
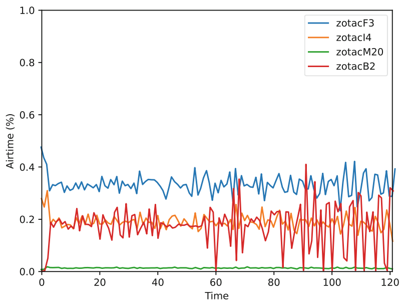


Fig. 6. A three-hop line topology used in the multi-hop reservation experiment. (Color figure online)

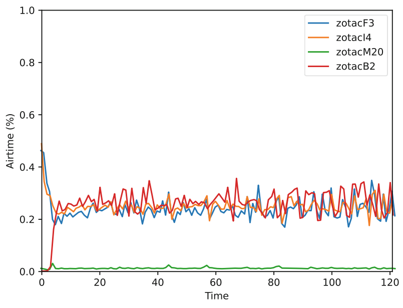
4.2 Multi-hop Reservation Evaluation

To evaluate the reservation process, experiments were conducted on the line topology with a multi-hop TCP flow. Figure 6 shows the line topology used with the multi-hop flow indicated in green. In the trials that used REACT, the TCP flow is started only after successful reservation along the path. In our experiments, we auction 80% of the channel, not 100%, leaving 20% for the control traffic; this is the same data/control traffic split used in [13]. The interior nodes of the line (e.g., `zotacF4`), must split 80% of the channel among three nodes (itself, and its previous and its next hop along the line). The reservation is placed for $\frac{80}{3} \approx 26.6\%$ airtime, which is close to the maximum amount of airtime that can be reserved in this scenario, assuming no other traffic flows. The TCP flow lasts for 120 s in both the REACT and 802.11 trials. Multi-hop routing is static with each node having neighbour information before the flow starts.

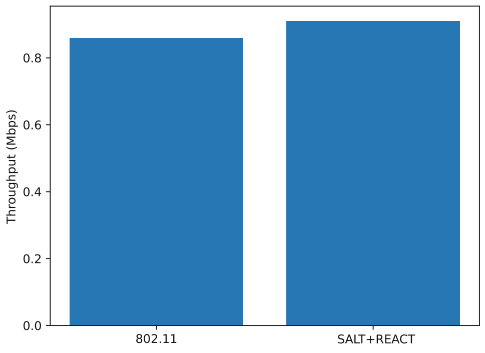
Figure 7 plots the airtime graphs for both REACT and 802.11 in the multi-hop scenario. The reservation of 26.6% airtime was placed after 0.9063 s and REACT converged after 9.041 s. Line topologies suffer both hidden and exposed node problems, with the airtime of 802.11 being highly unstable; in a multi-hop



(a) 802.11



(b) REACT with reservation.



(c) Multi-hop TCP throughput comparison.

Fig. 7. Multi-hop TCP results: The airtime of 802.11 and of REACT with reservation over the life of the multi-hop flow, and the multi-hop TCP throughput.

flow, the relatively high airtime of `zotacF3` cannot help the throughput of the multi-hop flow. The more consistent airtime and lower jitter for REACT leads to higher TCP throughput over REACT than over standard 802.11. Specifically, 802.11 achieved throughput of 0.86 Mbps, while multi-hop REACT achieved 0.91 Mbps.

5 Summary and Future Work

A new tuning algorithm, SALT, converges more tightly to REACT's airtime allocations, and tighter convergence leads to reductions in unfairness and jitter compared to 802.11 DCF. A new multi-hop reservation algorithm that leverages the airtime allocation and realization capabilities of this combination was proposed. With a reservation in place we have shown that REACT and SALT achieve higher throughput in a multi-hop TCP flow than in one that runs over 802.11 DCF.

There are several avenues of future work. To date, no real-world implementation of REACT handles nodes leaving an auction. This could occur if a node goes offline unexpectedly, or moves out of the range of the auctioneer. In simulation, Lutz et al. [19] used neighbour timeouts to determine when to evict nodes from the auction.

At present, SALT converges slowly. Perhaps combining SALT with the original tuning algorithm to leverage the lower jitter and faster convergence of each could be explored.

The reservation algorithm must be adapted to work with a dynamic routing protocol. This would likely require communication between REACT and the routing software.

All of these directions would contribute to the promising results of REACT with SALT to enabling fair, scalable mesh networks.

Acknowledgements. This work is supported in part by the U.S. National Science Foundation under Grant No. 1421058.

References

1. Bharghavan, V., Demers, A., Shenker, S., Zhang, L.: MACAW: a media access protocol for wireless LANs. In: ACM SIGCOMM (1994)
2. Bianchi, G., Tinnirello, I.: Remarks on IEEE 802.11 DCF performance analysis. IEEE Commun. Lett. **9**(8), 765–767 (2005)
3. Bicket, J., Aguayo, D., Biswas, S., Morris, R.: Architecture and evaluation of an unplanned 802.11b mesh network. In: Proceedings of the 11th Annual ACM MobiCom, pp. 31–42 (2005). https://doi.org/10.1145/1080829.1080833
4. Blefari-Melazzi, N., Detti, A., Habib, I., Ordine, A., Salsano, S.: TCP fairness issues in IEEE 802.11 networks: problem analysis and solutions based on rate control. IEEE Trans. Wirel. Commun. **6**(4), 1346–1355 (2007)
5. Bouckaert, S., Vandenberghe, W., Jooris, B., Moerman, I., Demeester, P.: The w-iLab.t testbed. In: Magedanz, T., Gavras, A., Thanh, N.H., Chase, J.S. (eds.) TridentCom 2010. LNICST, vol. 46, pp. 145–154. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-17851-1_11
6. Cali, F., Conti, M., Gregori, E.: Dynamic tuning of the IEEE 802.11 protocol to achieve a theoretical throughput limit. IEEE/ACM Trans. Netw. **8**(6), 785–799 (2000)
7. Camp, J., Robinson, J., Steger, C., Knightly, E.: Measurement driven deployment of a two-tier urban mesh access network. In: Proceedings of the 4th ACM Mobisys, pp. 96–109 (2006)
8. Carlson, E., Prehofer, C., Bettstetter, C., Karl, H., Wolisz, A.: A distributed end-to-end reservation protocol for IEEE 802.11-based wireless mesh networks. IEEE J. Sel. Areas Commun. **24**(11), 2018–2027 (2006)
9. Carrano, R., Magalhaes, L., Saade, D., Albuquerque, C.: IEEE 802.11s multihop MAC: a tutorial. IEEE Commun. Surv. Tutor. **13**(1), 52–67 (2011)
10. Ergin, M.A., Ramachandran, K., Gruteser, M.: An experimental study of inter-cell interference effects on system performance in unplanned wireless LAN deployments. Comput. Netw. **52**(14), 2728–2744 (2008)

11. Fu, Z., Zerfos, P., Luo, H., Lu, S., Zhang, L., Gerla, M.: The impact of multi-hop wireless channel on TCP throughput and loss. In: Proceedings of IEEE INFOCOM, April 2003
12. Garetto, M., Salonidis, T., Knightly, E.: Modeling per-flow throughput and capturing starvation in CSMA multi-hop wireless networks. IEEE/ACM Trans. Netw. **16**(4), 864–877 (2008)
13. Garlisi, D., Giuliano, F., Lo Valvo, A., Lutz, J., Syrotiuk, V.R., Tinnirello, I.: Making Wi-Fi work in multi-hop topologies: automatic negotiation and allocation of airtime. In: Proceedings of IEEE CNERT, pp. 48–55 (2015)
14. Gupta, A., Wormsbecker, I., Williamson, C.: Experimental evaluation of TCP performance in multi-hop wireless ad hoc networks. In: Proceedings of the 12th Annual IEEE International Symposium on MASCOTS, pp. 3–11 (2004)
15. IEEE standard 802.11: W-LAN medium access control & physical layer specifications, December 1999
16. Imboden, T., Akkaya, K., Moore, Z.: Performance evaluation of wireless mesh networks using IEEE 802.11s and IEEE 802.11n. In: Proceedings of the IEEE ICC, pp. 5675–5679, June 2012
17. Jardosh, A.P., Mittal, K., Ramachandran, K.N., Belding, E.M., Almeroth, K.C.: IQU: practical queue-based user association management for WLANs. In: Proceedings of the 12th ACM MobiCom, pp. 158–169 (2006)
18. Kosek-Szott, K., et al.: What's new for QoS in IEEE 802.11? IEEE Netw. **27**(6), 95–104 (2013)
19. Lutz, J., Colbourn, C.J., Syrotiuk, V.R.: ATLAS: adaptive topology-and load-aware scheduling. IEEE Trans. Mob. Comput. **13**(10), 2255–2268 (2014)
20. Mellott, M.J.: Smoothed airtime linear tuning and optimized REACT with multi-hop extensions. Master's thesis, Arizona State University (2018)
21. Papagiannaki, K., Yarvis, M., Conner, W.: Experimental characterization of home wireless networks and design implications. In: Proceedings of the 25th IEEE INFOCOM, pp. 1–13, April 2006
22. Shen, Q., Fang, X., Li, P., Fang, Y.: Admission control based on available bandwidth estimation for wireless mesh networks. IEEE Trans. Veh. Technol. **58**(5), 2519–2528 (2009)