Addressing Slot Drift in Decentralized Collision Free Access Schemes for WLANs

Wunan Gong and David Malone

Hamilton Institute, NUI Maynooth, Ireland.

Abstract. Decentralized Collision-Free Access (DCFA) schemes are an appealing family of MAC mechanisms in WLANs due to their good system throughput and decentralized nature. In this paper we consider the problem of *slot drift* for these schemes and provide evidence that DCFA can be vulnerable to such problems. We propose two schemes to enhance DCFA in this regard: Global View Synchronization (GVS) and Smart Collision Free (SCF). GVS aims to provide slot indexing, which helps stations correct their counters after drift. SCF accelerates the convergence process to the collision-free state for WLANs, and so reduces the impact of drift. Simulation results show that both GVS and SCF improve the system performance in the presence of slot drift.

Keywords: WLAN, Decentralization, collision-free MACs, slot drift

1 Introduction

In a WLAN, multiple stations typically share a common physical channel. The Medium Access Control (MAC) plays an important role in arbitrating accesses to the shared medium and thus influences channel utilization and system throughput. CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) and TDMA (Time Division Multiple Access) are two popular MAC mechanisms. In CSMA/CA, a station that has a packet to send has to keep sensing the wireless channel. If the channel is sensed idle, the station is permitted to start the transmission. Otherwise, it needs to defer the transmission, often using random backoff, and tries to transmit at some later time. In TDMA, time is divided into fixed or variable slots and different stations transmit in different slots. These two mechanisms have inherent drawbacks. In the former, collisions are inevitable, and in the latter, there are signaling overheads and tight synchronization requirements.

In recent years, new schemes that overcome the drawbacks and retain the good aspects of both TDMA and CSMA/CA have been proposed. Examples include L-BEB [2], ZC [3] and L-MAC [4]. In these schemes, time is divided into MAC slots and a fixed number of consecutive slots are grouped into a cycle. There are three types of MAC slots: idle slots, busy slots with successful transmissions, and busy slots with collisions. Note that these schemes allow stations to have slots of variable duration determined, say, by carrier sense.

In these schemes each station has a decentralized reservation process to reserve an exclusive slot for transmission in a cycle. This process is used when the

station just joins the network or when it loses its reservation. The reservation process stops and the system converges to a collision free state when all the active stations in the system have reserved an exclusive slot. We name such schemes *Decentralized Collision Free Access* (DCFA) schemes. Obviously, for convergence to a collision-free state, the cycle length should be no smaller than the number of stations in the WLAN system.

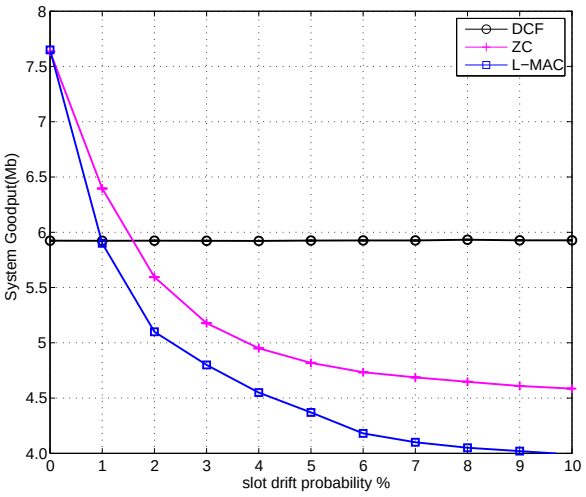


Fig. 1. The impact of Slot Drifts on ZC and L-MAC, compared with DCF.

DCFAs are appealing since collision-free access can outperform 802.11's DCF. However, the collision-free state is vulnerable to perturbations caused by channel noise, new entrants and, as we will show, slot drift. In DCFAs, transmission failures are used as a signal that reservation has failed. Thus, channel errors or new stations introducing collisions may move the system from a collision-free state. However, previous studies have shown low rates of new entrants and channel noise do not significantly degrade DCFA's performance [4].

For DCFA, it is important that each station has an accurate view of where it is in the transmission cycle, and ideally all the stations in the system should follow the slot evolution process synchronously and accurately. If a station miscounts the slots we refer to this as *slot drift*, which may lead to collisions. In practice, we have identified a number of possible reasons for slot drift: 1) station sensing errors, where carrier sense leads to the misidentification of slots; 2) clock errors, where the number of slots during an idle period is miscounted; 3) hardware/software miscounting, caused by implementation details.

As examples of these: 1) the 802.11 standard [1] requires only a 90% CCA detection for 9us slots; 2) for slots involving sleep and wakeup, clock drift/synchronisation problems are well-known in sensor networks; 3) the OpenFWWF firmware for Broadcom WiFi cards highlights the challenges in correctly implementing slot counting in software.

Perhaps surprisingly, the impact of slot drift may be significant. For example, Fig. 1 shows a simulation of DCF, ZC and L-MAC in the presence of slot drift. In these simulations we adopt a simple model of slot drift: we simulate a probability p of slot drift by introducing a probability of $p/2$ that each station leads the idealized slot count by one slot and a $p/2$ that it lags the idealized slot count by one slot. These probabilities are applied once in each idealized slot. We observe that ZC and L-MAC are outperformed by DCF in terms of system goodput in the presence of 2% slot drift. Consequently, we consider how to enhance DCFA schemes to be resilient to slot drift.

We consider two ways to enhance DCFA schemes: preventative and reactive. In preventative schemes, we aim to prevent the system from diverging from the collision-free state. To this end, we propose Global View Synchronization (GVS) which aims to correct slot drift as it happens. In reactive schemes, we try to alleviate the impact deviation from the collision-free state. In this regard, we propose the Smart Collision Free (SCF) scheme, a modification of ZC which speeds reconvergence.

The remainder of this paper is organized as follows: Section 2 outlines the related works; Global View Synchronization and Smart Collision Free are separately elaborated in Section 3 and Section 4; Section 5 presents the simulation results and performance analysis; We conclude the paper in Section 6.

2 Related Work

The MAC used in 802.11 networks [1] is called DCF. A core strategy used by DCF is BEB (Binary Exponential Backoff). With DCF, each station needs to uniformly choose a backoff counter C_b in the range [0, w-1] after a packet transmission. The value w is called the contention window and it has a minimum value CW_{min} and a maximum value CW_{max}. For each MAC slot, the backoff counter decrements by one and stations start packet transmission when it reaches 0. If the last transmission was successful, w is set to CW_{min}, otherwise stations double the contention window w unless it is CW_{max}.

L-BEB[2] is an evolution of BEB. L-BEB differs from BEB in one respect: stations choose a fixed backoff counter after a successful transmission. The value of the fixed backoff counter is shared by all stations, and is the cycle length for the network. If the number of stations is less than the cycle length, L-BEB will settle into a collision-free state after an initial period.

In ZC[3], each station notes all the idle slots in a cycle. In the case of a transmission collision, the station uniformly selects one slot for transmission in its next cycle from a candidate slot set which is composed of all the idle slots and

the colliding slot. In the case of successful transmission in a certain slot, stations continue to use that slot in the subsequent cycles for future transmissions.

L-MAC[4] adapts the idea of a self-managed distributed channel selection algorithm from [5] and keeps updating selection probability for each slot in a cycle. In the case of a successful transmission, the stations persist with the same slot in the following cycle and the selection probability for that slot is set to 1.

While the details of the reservation phase of each of these protocols is different, when the system converges to collision-free state, the behaviors are the same. These protocols can be implemented with a backoff counter, and do not require stations to agree on slot labeling (or indexing). However, we can still think of then as each station trying to reserve an exclusive slot in a cycle.

3 Global View Synchronization

GVS is our preventative scheme to enhance DCFA. We aim to correct slot drift on stations before their next transmissions, so the potential collisions can be avoided and the collision-free state can thus be maintained. The basic idea of GVS is that it provides a slot labeling benchmark to facilitate the correction of slot drift on stations.

As mentioned, stations in the DCFA system do not require consensus on the slot labeling. Once a station has reserved a slot, it will periodically send packets (if it has packet to send) and the period is the cycle length. We refer to the slot labeling as a station's *view*. For example, in Fig. 2 there are three stations with different views of the MAC slots. The cycle length is 4 and the highlighted square shows their reserved slots. Station 1 and station 2 both reserved slot 2, but these two "slot 2"s are different due to their staggered labelings.

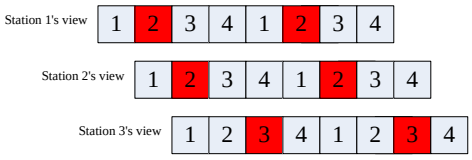


Fig. 2. Slot views of three stations.

Now, we aim to make all the stations have the same view and regard this *global view* as a benchmark. If a station encounters slot drift, their slot labeling will change, and so can be corrected to match the global view. Since slot drift happen from time to time, hence it is not possible to make all the stations always have the same view. Consequently, the global view will be the stations in the view held by some large subset of stations.

Consequently, GVS deals with two issues: 1) maintaining a global view; 2) correcting drift on stations. The station performing GVS must share the common

cycle length, C, with the network and maintain some slot view information which it includes in its transmissions. Stations can then observe if their view is not aligned with other stations. The details of GVS are:

1. Each station announces its view in each transmitted packet. A station's view can be represented by its current slot index, so here the announced view is actually a slot index. The announcement only uses several bits in each transmitted packet.
2. Each station maintains a "view set", S. The elements in the set correspond to views successfully sent by this station and successfully received from other stations. Since slot indices are always evolving, the stored elements indicate the difference between the current slot index of the station and the successfully sent/received slot indices:

$$d_v = (\text{index}_{rx|tx} - \text{index}_{curr} + C) \mod C.$$

3. If a station successfully transmits or receives a view, it checks whether there is already an element in its view set S which corresponds to that view (i.e. if it has seen that view before). If so, is assumes multiple stations share this view, and so it changes its view to match and then replaces S with the empty set. Otherwise it inserts that view into the set. Thus, seeing two stations with the same view results in a synchronization of views.

   if $d_v \in S$ **then**
   $\text{index}_{curr} \leftarrow \text{index}_{rx|tx},\ S \leftarrow \emptyset$
   else
   $S \leftarrow S \cup \{d_v\}$
   end if

4. A station may have noted some important slot indices which may correspond to idle slots, reserved slot, etc. These noted slot indices need to be updated when a station updates its view. However, there are two reasons why a station may be updating its view: 1) it has just entered the network and needs to synchronize its view to the global view; 2) it encounters a slot drift. We handle these cases differently, as follows. A station enters a WLAN system with state set to INITIAL and changes to state STABLE after the first view synchronization operation. Corresponding to these, we have two types of slot index updating operations: index-shifting in the INITIAL state and index-keeping in the STABLE state. From the INITIAL state, index-shifting updates stored indices by the difference in the labeling,

$$\text{index}_{new} = (\text{index}_{old} + d_v) \mod C,$$

   because we believe the stored slots were not labeled using the global view, and so the labels must be updated. Index-keeping does not relabel the slots, because we believe the slots were correctly labeled while synchronized to the global view, and the stations view has only recently drifted.

This scheme results in all stations converging to the same view in two cycles, providing there is no slot drift or collisions in these cycles. To see this, note that

if two stations share a view, then the first two to announce the same view results in the network being synchronized. If no two stations share a view, then the first station to transmit in the second cycle will synchronize the network.

There is a possibility that two stations encounter slot drift and then announce their views consecutively, or that collisions or errors may result in delays in synchronization. We evaluate GVS's practical capability to maintain a global view and improving performance in the presence of slot drift in Section 5.

4 Smart Collision Free

In this section, we look at another way to improving the performance of a collision-free MAC. We aim to speed its reconvergence to a collision-free state after being perturbed. In particular, we propose the Smart Collision Free algorithm, which is an evolution of ZC. Recall that in ZC, the candidate slot set for a station in the reservation process is composed of all the idle slots and the colliding slot in a cycle. In the case that all the stations have the same view, all the colliding stations share the same idle slot sets in a cycle for selection. If we divide the idle slots in a cycle into multiple subsets and let colliding stations with different colliding slots select from different idle slot subsets, the collision probability can be reduced and thus convergence speed can be improved. This is the basic idea of SCF: to partition the idle slot set.

The details of one possible implementation of SCF is shown in Alg.1. In each cycle, each station notes all the idle slot indices and the number of collisions. If its transmission collides, the station notes the colliding slot index and the relative position of the colliding slot in all the colliding slots in the cycle. At the end of a cycle[1], the station selects the next reserved slot by partitioning the idle slots among groups of colliding stations. Note that since stations do the selection at the end of a cycle, when a station just enters a WLAN, it should monitor the channel for one cycle.

Fig. 3 shows an example of SCF. There are 10 slots in a cycle, and in this cycle there are 3 colliding slots and 5 idle slots. Ideally, we aim to evenly split the idle slot sets into 3 subsets. However, as 5 can't be divided by 3, we allocate the first 3 idle slots (the slots are ordered by time) to 3 subsets and leave 2 slots which will be selected in a probabilistic way. Now, consider a station which collides in the second slot (the block labeled "Colliding slot" and numbered with 2). At the end of the cycle, the station needs to select the next reserved slot. Since the colliding slot position is 2, it selects the second idle slot. Meanwhile, with probability of 2/3, the station needs to select one slot from idle slots 4 and 5 and the selection is random. Then the station puts the selected slots and the colliding slot into the candidate slot set. Finally, the station randomly select one slot as the next reserved slot from the candidate slot set.

Note, the SCF scheme assumes that all the stations have the same view, so they regard the idle slots as having the same order. In the situation where

[1] The discussion of ZC [3] notes that decisions can be made either at the end of a cycle or after a collision, and there is no significant difference in convergence speed.

Algorithm 1 The SCF Algorithm.

slot_{curr} : The currently reserved slot for this station.
slot_n : The new reserved slot for this station.
slot_c : The slot this station collided on.
CS : The candidate slot set.
IS : The idle slot set, with IS_i the i^{th} idle slot.
n_c: Number of slots with a collision in the last cycle.
i_c : This station collided in the i_c^{th} slot with a collision in the last cycle.
for each cycle **do**
  if no transmission **then**
    if $\text{slot}_{curr} \in IS$ **then**
      $\text{slot}_n \leftarrow \text{slot}_{curr}$
    else
      $\text{slot}_n \leftarrow$ random element of IS
    end if
  else if successful transmission **then**
    $\text{slot}_n \leftarrow \text{slot}_{curr}$
  else failed transmission
    Find q, r so that $|IS| = qn_c + r$.
    $CS \leftarrow \{\text{slot}_c, IS_{(i_c-1)q+1}, \ldots IS_{(i_c)q}\}$
    with probability r/n_c **do**
      $CS \leftarrow CS \cup \{$ random slot from $IS_{qn_c+1}, \ldots IS_{qn_c+r}\}$
    done
  end if
end for

not all views are synchronized, we will see that SCF may still help to speed reconvergence to a collision free schedule.

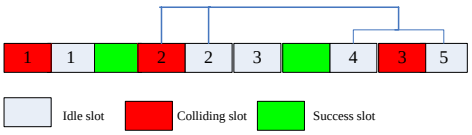


Fig. 3. An illustrative example of SCF.

5 Performance Evaluation

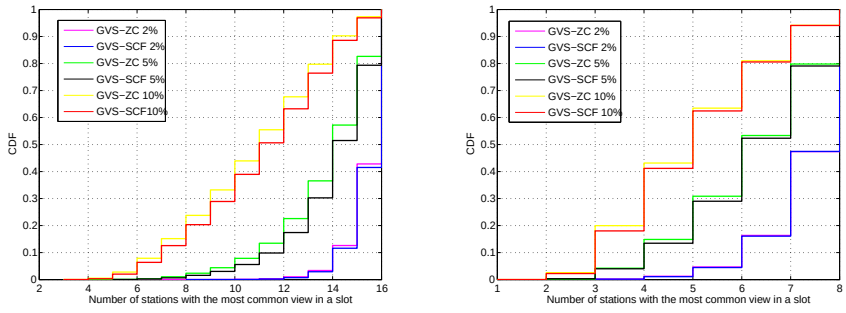
We conducted a simulation study of GVS and SCF. We compare ZC (the scheme with the fastest convergence of the DCFA schemes) to SCF in the presence of slot drift. We also assess GVS's ability to help ZC and SCF maintain a synchronized view, and the resulting impact on performance.

Table 1. Network Parameters in 802.11b.

Parameters	Durations(μs)
Slot time, σ	20
Propagation delay, δ	1
$CW_{min} = 32\sigma$	640
$CW_{max} = 1024\sigma$	20480
DIFS	50
SIFS	10
PLCP Header@1Mbps	192
MAC Header+CRC, 28 Bytes@11Mbps	20
UDP+IP+Payload, 1500 Bytes@11Mbps	747.6
ACK, 14 Bytes@2Mbps	56
ACK_TIMEOUT	12

As we are focused on performance of slot allocation, we work with a slot-level simulator written in C++. For the simulation, unless otherwise noted, all stations are transmitting UDP traffic with payload 1500 bytes and a PHY rate of 11 Mbps. The same channel is shared by all stations and there are no hidden node problems. For simulation of DCF, we adopted the typical 802.11b protocol and the parameters are shown in Table 1. All the results are obtained over repeated simulations.

5.1 Global View Synchronization



(a) C=N=16 (b) C=16, N=8

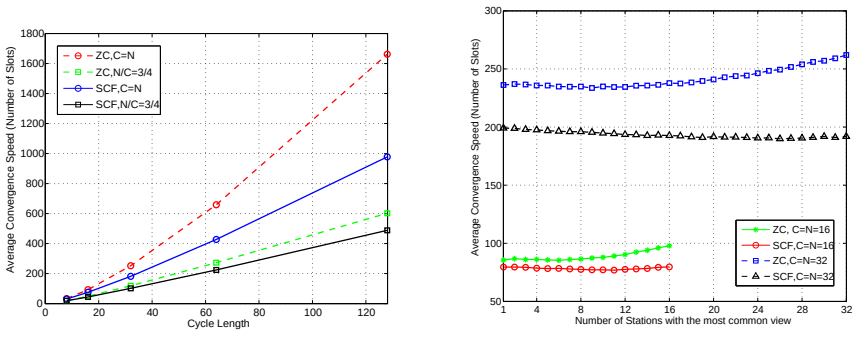
Fig. 4. CDF of number of stations with the majority view. C: Cycle Length, N: Number of stations in the system.

Fig. 4 shows how effective GVS is in helping stations to hold the same view when either ZC or SCF is used. We fix a cycle length of $C = 16$ and consider a system with $N = 16$ or $N = 8$ stations (Fig. 4(a) and (b) respectively). Stations always have a packet to send. We apply slot drift probability of 2%, 5% and 10% to all stations and consider the number of stations holding the majority view when GVS is applied. Specifically, let M_i be the size of the largest group of stations with a consistent slot labeling for slot i, then Fig. 4 shows the CDF of the values of M_i.

As expected, GVS performs better with smaller slot drift probability. For example, considering Fig. 4(a), we see that for 10% of the time the majority group is smaller than 14 stations for 2% slot drift, but smaller than 10 stations for 5% and smaller than just 6 stations if the slot drift is 10%. However, overall GVS does a reasonable job at keeping views consistent. We also see that GVS works marginally better with SCF than ZC. We attribute this to quicker convergence times. Though we have not shown results here, it is also possible to apply GVS to L-MAC and L-BEB.

5.2 Convergence Speed in Ideal conditions

In the presence of slot drift, the collision-free state is volatile. To evaluate convergence speed, we consider a scenario with no slot drift where all stations start the slot reservation process at the same time and begin with no reserved slot. This would be representative of a situation when all stations power up at the same time. The convergence speed is measured by the average number of elapsed slots before convergence to a collision-free state.

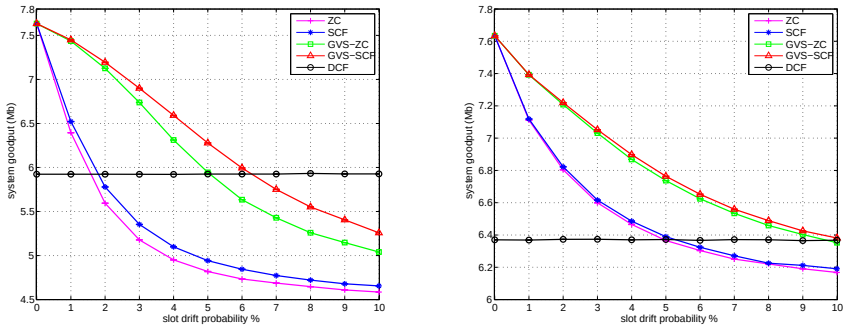


(a) (b)

Fig. 5. Convergence speed comparison between ZC and SCF. C: Cycle Length, N: Number of stations in the system.

In Fig. 5(a) we consider networks with various cycle lengths and a number of stations N so that $N/C = 1$ or $N/C = 3/4$. We assume that stations initially randomly select their view and may not have the same view. We see that SCF generally outperforms ZC, with the difference being most significant when C is large and N is close to C. In Fig. 5(b), the number of stations that hold the "majority view" is varied. We see that under similar conditions, SCF shows marginally shorter convergence times, and that SCF's convergence times become shorter as more stations share the same view. Interestingly, ZC's convergence is actually marginally shorter when larger groups hold the same view. We believe this is because in our implementation of ZC, stations choose new slots at the end of the cycle, and there is a small advantage to stations making a decision at different times.

5.3 Long-term System Goodput



(a) C=N=16 (b) C=16, N=8

Fig. 6. System goodput vs slot drift probability, comparison of 5 schemes.

We now compare the long-term system goodput of five schemes: DCF, SCF, ZC, SCF with GVS and ZC with GVS. We fix the cycle length at $C = 16$ to allow comparison with DCF. The slot drift probability is varied from 0% to 10% and stations always have packets to send. Fig. 6 shows the goodput for $N = 16$ and $N = 8$. We see that GVS can considerably improve the system goodput in the presence of slot drift, and SCF schemes are generally better than ZC schemes. When GVS-SCF is adopted, the system can sustain up to 6% slot drift probability $N = 16$ and up to 10% with $N = 8$ before the system goodput drops below that of DCF.

It is noticeable that there seem to be no changes in system goodput for DCF when slot drift probability increases. We can explain this using Bianchi's model [7]. We know the system goodput depends on the collision probability p and transmission probabilities τ in the network. Both with and without slot drift, the network can be modeled using

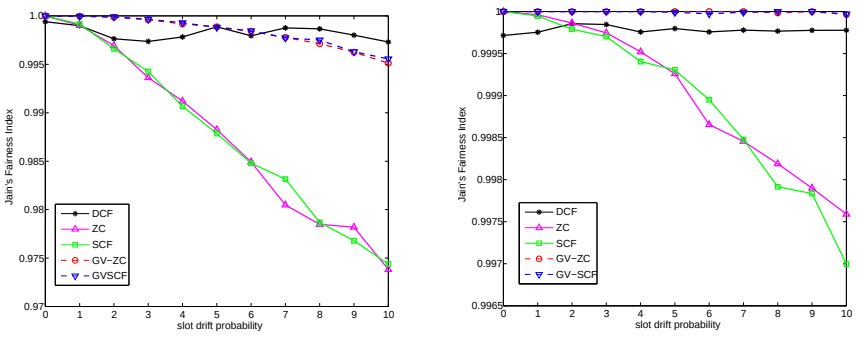
$$p = 1 - (1 - \tau)^{(N-1)} \tag{1}$$

From the work of Kumar et al [6], we have

$$\tau = \frac{1 + p + p^2 + ... + p^K}{b_0 + pb_1 + p^2 b_2 + ... + p^K b_K} \tag{2}$$

where, K is the retry limit and b_k is the average number of backoff slots chosen at the k^{th} attempt. To find p and τ we search for solutions of Eq.1 and Eq.2. We note that slot drift only has a direct impact on the b_k. Since we have modeled slot drift as a mean-zero random walk which is added to the backoff process, we expect that the b_k will be unchanged, and so DCF's performance will be unchanged by this sort of slot drift.

5.4 Fairness



(a) $C = N = 16$ (b) $C = 16, N = 8$

Fig. 7. System fairness vs slot drift probability, comparison of 5 access schemes.

Fairness is another concern in WLANs. When slot drift is applied to all stations in a system, the system remains homogeneous and fairness is not an issue. However, slot drift probabilities on different stations may vary. In this

section we consider similar networks to those in Section 5.3 but here slot drift is applied to only one station. Fig. 7(a) and (b) shows Jain's index[8][9], and reveals that system fairness can be improved with Global View Synchronization. Our results indicate the station with drift can usually correct it before its transmission and thus avoid collisions.

6 Conclusion

In this paper, we investigated the potential problem of slot drift in Decentralized Collision Free Access systems and proposed two schemes, GVS and SCF, to deal with slot drift problems. We have shown how GVS can help the system maintain a common slot indexing and use it as a benchmark for correcting slot drift on stations. SCF speeds up the convergence process to the collision-free state and thus alleviates the impacts due to system's deviation from the collision-free state.

Acknowledgment

The authors were supported by SFI grant 08/SRC/I1403 (FAME) and 07/SK/I1216a.

References

1. *IEEE 802.11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*
2. J. Barcelo, B. Bellalta, et, *Learning-BEB: Avoiding Collisions in WLAN* 14th Eunice Open European Summer School 2008.
3. J. Lee and J. Walrand, *Design and analysis of an asynchronous zero collision MAC protocol.* Tech. Rep. UCB/EECS-2007-63, May 2007.
4. M. Fang, D. Malone, K. R. Duffy, and D. J. Leith *Decentralised learning MACs for collision-free access in WLANs.* Springer Wireless Networks, May 2012.
5. D.J. Leith and P. Clifford *A self-managed distributed channel selection algorithm for WLANs.* In Proceedings of International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks, pages 19, 2006.
6. A. Kumar, E. Altman, D. Miorandi, and M. Goyal. *New insights from a fixed-point analysis of single cell IEEE 802.11 WLANs* IEEE/ACM Transaction on Networking, VOL.15,NO.3, June 2007
7. G. Bianchi. *Performance analysis of the IEEE 802.11 distributed coordination function* IEEE JSAC, 18(3):535547, March 2000
8. R. Jain, D. Chiu, and W. Hawe *A quantitative measure of fairness and discrimination for resource allocation in shared computer systems.* Arxiv preprint cs/9809099, 1998.
9. C.E. Koksal, H. Kassab, and H. Balakrishnan *An analysis of short-term fairness in wireless media access protocols.* Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pages 118119. ACM New York, NY, USA, 2000.