PERFORMING CLOCK SYNCHRONIZATION FOR POWER MANAGEMENT IN MULTI-HOP AD HOC NETWORKS

Jatinder Singh
Sant Longowal Institute of Engg & Technology, Longowal, Punjab, India
E-mail : garai.jatinder@yahoo.co.in

Abstract

One of the common underlying assumptions in IEEE 802.11 standards' ad hoc mode operations is that all stations in the same IBSS can hear each other directly. In other words, the ad hoc mode of 802.11 supports only single-hop ad hoc networks. When extended to multi-hop environments, IEEE 802.11 protocols show a poor performance, if they are able to work at all. This has been acknowledged by researchers of wireless ad hoc networks [1, 2, 3]. However, because 802.11 already has a world-wide customer base — millions of devices around the world are "802.11- capable," and that number is quickly increasing — much research has been done on IEEE 802.11-based, MANET-oriented protocols. In this paper, I try to alleviate the IEEE 802.11 MANET problem by considering two fundamental issues in wireless networks: time synchronization and power management. I ponder the relationship between them, and compare different power management schemes. Eventually, I establish that clock synchronization plays a very important role in efficient power management algorithms, and present an integrated protocol that merges synchronized sub networks to achieve power-saving in a global synchronized environment. I then show the correctness of the proposed protocol, and finally present simulation results that demonstrate the merits of the proposed protocol.

Keywords: Wireless Sensor Networks, Medium Access Control, Sensor MAC, Timeout MAC, Distributed Coordination Function, Carrier Sense Multiple Access /Collision Avoidance, Carrier Sense Multiple Access /Collision Detection

I. INTRODUCTION

A. The Problem

Power management is an important approach towards energy conservation in MANET, and is adopted by IEEE 802.11. Also, due to a station's periodically turning off its transceiver, paging becomes an essential part of the power management protocol. It is important for the stations to wake up at the same time to receive the paging information. For this reason, time synchronization plays an important role in power management in IEEE 802.11-based ad hoc networks1. However, clock synchronization itself is an issue yet to be solved. For example, TSF in IEEE 802.11 (Section 2.2.1) has two main characteristics:

- There is only one station sending the beacon packet in each beacon interval.
- Clocks are adjusted forward only.

Because of these characteristics, Huang et al. point out the IEEE 802.11 TSF suffers scalability problems [4]. Huang et al. shows that due to inefficient clock adjustment and multiple access collision, the IEEE 802.11 TSF's time synchronizing ability deteriorates quickly when more stations are in the system. To solve this problem, the Adaptive Timing Synchronization Procedure is proposed. In this procedure, instead of all stations participating the beacon transmission in every beacon interval, each station maintains a different frequency for beacon transmissions. This frequency is related to the relative speed of the station's clock. The faster a station's clock, the more frequently it will participate in the beacon transmission at the beginning of beacon intervals. This will reduce the contention of beacon transmission while increasing the convergence speed of clock synchronization. Other synchronization research is underway in multi-hop networks. Since clock synchronization is difficult and unreliable in MANET, some researchers propose power management protocols without using any clock information at all [5, 6].

B. Power Efficiency

A synchronized system gives the most simple and efficient support to power management, and has the best performance on power-saving and the shortest delay, since stations wake up at the same time and only when it is necessary. It also makes the beacon and ATIM window possible, which can separate short but vital beacon and ATIM packets from data packets. When overall traffic is very low, each station only has to stay awake for the beacon and ATIM windows, which usually counts for less than 20% of each beacon interval. However, synchronous power management does rely on the performance of TSF, which not only is difficult to maintain a good scalability to the size of the multi-hop network, but also has to be able to discover and MANET.

The schemes based on no clock synchronization give up on maintaining the synchronization in a dynamic distributed system. When transmission is needed, the station has to try to discover the station it wants to talk to using the schemes devised in the papers. To guarantee such a discovery in the random environment of an asynchronous system, the station has to stay awake for several beacon intervals. According to the lower bound proof in [6], stations need $\sqrt{T}$ fully awake intervals for

one overlap in every T intervals. A larger T value will bring down the overall percentage of time the station is awake, but it also introduces unacceptable delay. For example, at very low traffic situations, to reach the performance of the synchronous system of a 20% awake ratio (the percentage time stations have to stay awake), T has to be at least 25. That means that in the worst case, the receiver will not get paging information until the 25th beacon interval after the sender starts the contact initiation. A reasonable delay with T = 7 used in [6] equals a 43.86% awake ratio. The authors of asynchronous systems are well aware of this problem. In [6], they propose that the station passively tracks the clock and the schedules of the neighboring stations once it gets them, and uses them to guide future packet exchanges. This is very close to a synchronization mechanism, although it does not actively adjust the clock.

C. Broadcast and Multicast

Broadcasting and multicasting is important for fully distributed systems like wireless ad hoc networks. Most maintenance and control packets have to be sent as broadcast messages, and we also can imagine that in some scenarios for the use of wireless ad hoc networks, such battlefields, the aftershocks of disasters or emergencies, and sensor networks, there might be many data packets that are broadcast or multicast traffic. This presents a big challenge for asynchronous systems.

In a synchronous system as described in the IEEE 802.11 standard, because of the synchronized clock in each station which can make all stations stay awake for the same period of time, and the existence of the ATIM window used to notify all stations of incoming broadcast/multicast packets, all the receivers will be ready for the packets when the sender sends them. Even in the case of multicast, the non-participating stations can be made to yield to the multicast traffic since they were also notified during the ATIM window. In the IEEE standard for Wireless LAN, there is no ACK packet for the broadcast data packet.

However, in an asynchronous system, because the offsets among a group of stations could be any amount from zero to the length of the beacon interval, the broadcast/ multicast packets have to be sent in the intersection of all the receiving stations' data windows. The more receiving stations for a broadcast/multicast session, the shorter the valid sending window for the broadcast/multicast packets will become. To overcome this, the authors of [5] propose that both MTIM packets (their version of ATIM packets) for the broadcast/multicast packets and the real broadcast/multicast packets are sent to a group of receivers who have some overlap for their ATIM windows. Will this solve the problem? Let us take a look at the broadcast/multicast procedure in the asynchronous system.

For the MTIM packet, because the ATIM window is only a small part of a beacon interval (a typical ATIM window is about 10% of the length of a beacon interval), it is harder to find overlap for multiple ATIM packets. I calculate the expected number of overlapped ATIM windows over the time of one ATIM window with a uniform distribution of beacon interval starting times.

$$E(overlapATIM) = \sum_{i=1}^{N_r} i \times C_{N_r}^{i} t_w^{i} (1 - t_w)^{N_r - i} \qquad (1)$$

where Nr is the number of receivers of the broadcasting packets and t_w is ratio of the ATIM window to the whole beacon interval. This shows that about one ninth of the broadcast receiving stations can find an intersection within one ATIM window when an ATIM window occupies 10% of a beacon interval. When the ATIM window expands to 20%, this expected overlap stations number increases to one fourth of the receiving stations. There are two more factors we have to consider when estimating the overlap stations number in a real life scenario. The first is the length of overlap. In equation 1, any amount of overlap qualifies, but in the real world, as small as an ATIM packet is, it still needs time to transmit. Also, the protocol has to take the collision of ATIM packets into consideration. From the standard, we can tell that sending a management packet like an ATIM could take from 40 μs to 82 μs, depending on the data rate used. In a synchronous system with a dedicated ATIM window at the beginning of each beacon interval, the sender's broadcasting ATIM has to compete with other stations' ATIM packets, or the protocol can have a special broadcast ATIM which has priority over unicast TIMs, like the Delivery Traffic Indication Message (DTIM) in IEEE 802.11 for AP in the infrastructure mode. However, in an asynchronous system, the ATIM packets have to contend for the medium with the data and control packets from other stations. Since each ATIM packet represents at least one data packet to be transferred in the beacon interval, the number of ATIM packets should always be less than or equal to the number of data packets. Besides this, the longer data packet also makes it difficult for the contending ATIM packet to gain access to the medium. So even though the ATIM packets have a higher priority than the data packets (ATIM packets use SIFS to decide whether the medium is free or not, see Section 1.3), they still have to wait for the current packet to finish transmission.

In [6], the authors did not specifically talk about broadcast/multicast, but in this kind of asynchronous system, broadcast/multicast packets have to be sent more than once to groups of receivers during the overlap of their asynchronous data windows. So the problems I discussed above are still present. The scheme presented in [6] no

longer uses the ATIM packet to pre-announce the incoming data packets. Although this eliminates the extra steps of finding the overlap of the receiving stations' ATIM windows and competing to send the ATIM packets, this brings in new issues. In a system with special ATIM packets, one more round of contention for ATIM packets does allow for some control over the number of sending stations that can participate in the contention in the data window (those who failed the ATIM contention will not contend to send data packets in the following data window). By limiting the length of the ATIM window (so the number of stations passing the ATIM contention is also limited), the protocol can limit the contention in the data window. In [7], the authors study this relationship between the length of the ATIM window and data window contention. Controlling the contention in the data window is specially desirable for broadcast/multicast transmissions because the broadcast/multicast packets are not acknowledged. Hence, eliminating the ATIM might not always bring desirable results.

D. Beacon And Data Contention

In a synchronous system like IEEE 802.11, beacon and ATIM packets are protected from data traffic by being assigned to transfer in a special period of time when other packets are not allowed to transmit. We cannot implement this kind of protection in an asynchronous system. However, giving control and management packets higher priority over data by using different inter-frame spacings will solve this problem. Even when a beacon packet starts in the middle of an on-going data transmission, the delay of waiting for data packet to finish will become less significant as the transfer rate gets faster in the physical layer. Spreading beacon and ATIM packets throughout the whole beacon interval can reduce the contention of those messages with each other. In an asynchronous system, beacons play a less important role as TSF, so this benefit is reduced. However, this can help us in the scalability problem of TSF. Other than the issues I discuss above, physical layer operations might also require a synchronous system, which I will not detail here. Thus, a global clock synchronization scheme in a multi-hop environment would be a great benefit, if not essential, for an efficient power management protocol. How to achieve and maintain such a synchronization is the main challenge, and that becomes the problem I try to solve in the next section.

II. THE PROTOCOL

In a system where a central station exists and all other stations synchronize with it, like cellular networks, it is very easy to achieve such global synchronization, but in a distributed mobile system like MANET, both scalability and mobility bring serious challenges to synchronizing clocks across the whole system. As I mentioned before, the TSF in the IEEE 802.11 standard is based on a single-hop environment, for a system with a dynamic topology in a multi-hop MANET. The dramatic increase in the number of stations that need to be synchronized when going from single-hop to multi-hop necessitates that something be done to the TSF. In [8], the proposed Adaptive Timing Synchronization Procedure has already partly solved the scalability issue for synchronizing a system of a reasonable size. Recently, Lai and Zhou [9] have proposed a clock synchronization protocol for multi-hop networks with a synchronization accuracy of less than 100 microseconds. However, this protocol hinges on two undesired assumptions:

· The MANET is initiated by a single station. Because of the lack of merging groups of unsynchronized stations, the protocol handled one station a time. It results in the perception that the MANET is “grown” from a single point.

· The MANET, as a graph, is always connected. If there exist two disconnected subgraphs, they must be considered as two different MANETs and it is assumed that there is no communication between them.

With all these previous works and unsolved issues, I set up my research on the assumption that a scheme already exists to keep a connected sub-network of a reasonable size2 in sync within a certain limit, within which the software can ignore the clock difference. This assumption allows my protocol to adopt any advancement in clock synchronization research. These synchronous sub-networks are not aware of each other due to either being out of range, or being “grown” from different nodes3. My goal is to present a protocol that can discover these sub-networks and merge them into a bigger system as they come into communication range with each other (due to mobility), so that a synchronous power management protocol can be used to for energy conservation.

A. Proposed Protocol

Achieving global synchronization as described above is done in three steps. First, stations try to discover unsynchronized stations periodically throughout their lifetime. Second, after unsynchronized stations are discovered in the first step, a merge process will synchronize the stations in two groups, i.e. merging two sub-networks. Third, during other times, the regular clock synchronization scheme (based on my assumption) is used for maintaining synchronization. If, for any reason, like moving out of range, or being turned off for a long period, stations loose synchronization, they can always be re-discovered and re-join the system via steps one and two. Before I present my algorithm, I will visit some design issues.

- The discovery process cannot be run too often since it consumes much more energy. Its frequency depends on the trade-off between power efficiency and the delay of discovery. We always want some kind of "hint" to improve the efficiency of discovery process, i.e. running it as few times as possible, but still making the discovery in a timely manner. In my protocol, an adaptive approach based on the change of neighbors is used as the trigger of the discovery process. I think the topology change would be a good time to run the discovery process, and use the change of neighbors as an indicator of possible topology change around the station. However, neighbor changes do not necessarily mean new nodes in the neighborhood, and power management makes them difficult to track, a station makes the decision with a probability based on the changes of its neighbors it has noticed (neighbor change threshold Thn), and backs it up with a fixed interval (discovery threshold Thd), during which the discovery must be run at least once. Both of these thresholds should be system parameters and adjusted to vary the discovery delay.
- During the merging process, I let the sub-network with a slower clock adopt the clock value of a sub-network with a faster one. This complies with the standard.
- During the merging step, stations that have just acquired a new clock value will maintain two different beacon windows so that they can populate the new clock value to their neighbors who still use the old clock, by sending their beacons with the new clock in its original window. This beacon will be specially marked so that stations receiving it can distinguish it from beacons in the regular synchronization process, and can change their clock immediately.
- Since beacon packets are not acknowledged, stations that adopt the new clock value have to decide when all its neighbors have come to the new clock so that it can exit the merging step. To achieve this, these stations stop sending beacons in the original beacon window after N beacon transmissions, and start to listen for the beacon that still carries original clock value. This listening period should last for Δ/2rd_min (see Section 2.2 for definition and details about this time), and the value N is a system parameter which will be fine-tuned in the real world.

Putting these ideas that I have discussed so far together, I get three procedures in my protocol. One is the discovery procedure, which runs repeatedly to search for unsynchronized sub-networks. In this procedure, like in those asynchronous power management schemes, a station stays in awake state for the full beacon interval to listen to the beacons outside its beacon window. If such beacons are received, it means stations with different schedules are within transmission range. Then the clock values carried in those beacons are stored in the clock table, and the station will enter the merging procedure in the next interval. In the merging procedure, the station tracks two schedules at the same time. One is the schedule associated with the fastest clock in the clock table, and the other is its original schedule. During the beacon window of its original schedule, it sends a special beacon packet mentioned above to change its neighbors' schedule. During the new schedule, it follows the regular synchronization algorithm. While a station is not in the discovery or the merging procedures, it runs the regular procedure for handling regular operations, including the

```
Discovery Procedure
  Clear clock table, and leave only the current clock; Reset discovery counter
  The same action for beacon, ATIM and incoming/outgoing data as in Regular
Procedure
  Stay awake for the whole beacon interval even no traffic to listen for beacons
  if Received special beacon
  then
    Enter Merging Procedure at the end of interval
  else
    if Received beacons from stations running under different clock and schedule
    then
      Enter those clock values into clock table
      Enter Merging Procedure at the end of interval
    else
      Enter Regular Procedure at the end of interval
    endif
  endif
```

Fig. 1. Discovery procedure of the new power-saving algorithm

maintenance of clock synchronization and power management. In this regular procedure, it also checks for neighbor changes to decide whether to run the discovery procedure. The protocol proposed here can adopt any regular procedure, but I use the TSF and ATIM operations in IEEE 802.11 as an example. The pseudo-code of these procedures are in Figure 1, Figure 2, and Figure 3.

B. Correctness Of The Protocol

In this section, I discuss the correctness of the above protocol. I start with rephrasing the assumption presented at the beginning of Section 2.

Assumption (Partial Synchronization Assumption). There is a distributed clock synchronization algorithm based on beacon broadcasting, which can limit the clock difference between any two stations no more than Δ in a connected system. From this assumption, I can see the difference between any pair of stations' clock values, $|t_A - t_B|$, is bounded by the maximal time between two consecutive successful beacon exchanges, Δt_A and Δt_B as shown in Figure 4, and the clock drift rate, r_d.

$$Max\{|t_A - t_B|\} = Max\{\Delta t_A\} \times r_{dA} + Max\{\Delta t_B\} \times r_{dB} \quad (2)$$

Because the propagation delay of RF signal can be ignored,

other transmission delay, like IFS and transceiver preparation time, has already been taken into consideration in timestamp calculations, and there is no queuing delay for beacon packets (since they are transmitted in a special window in which only beacon packets can be transferred and each station only transfers one beacon at a time), we can unify Δt_A and Δt_B as Δt. So Equation 2 can be reduced to

$$Max\{|t_A - t_B|\} = Max\{\Delta t\} \times (r_{dA} + r_{dB}) \quad (3)$$

Assume the station's clock drift rate, r_d , is greater than 0 (if $r_d = 0$, the clock is a perfect clock, then synchronization will be achieved without any special protocol. I do not discuss this rare case in this paper), and using Δ in the Partial Synchronization Assumption to replace

Max}$|t_A - t_B|$…, I can get the maximal time between any consecutive successful beacon exchanges

$$\Delta t_{max} = \Delta/(r_{dA} + r_{dB}) \quad (4)$$

When I consider more than just two stations in the synchronized system, the maximal time between consecutive successful beacon exchanges at any station is bounded by

$$\Delta t_{max} \leq \Delta/2r_{d\_min} \quad (5)$$

Thus, I have got the relationship between the maximal time between consecutive successful beacon exchanges, Δt_{max}, the maximal clock difference, Δ, and the minimal clock drift rate, r_{d_min}.

Now, I will show that a station running the protocol described in Section 2.1 will discover stations with different clock values inside its range in finite time.

```
Merging Procedure
 After discovery procedure, there should be more than one clock values in the
clock table. Find the one with the fastest clock to adopt, discard the rest except
the original one
 Adjust clock to adopt the faster clock
 Reset original beacon counter
 if In the beacon window corresponding to the original clock
 then
  if Beacon counter < N
  then
   Send special beacon with new clock value using the same beacon transmission
procedure as in Regular Procedure
   if Transmission is successful
   then
    Beacon counter++
    if Beacon counter = N
    then
     Reset and start listen timer
    endif
   endif
  else
   if Listen timer < Δ/2r_d_min
   then
    Listen for regular beacons
    if Found regular beacon
    then
     Reset and stop listen timer
     Reset beacon counter
    endif
   else
    Enter Regular Procedure
   endif
  endif
 else
 Follow the same actions as in Regular Procedure according to the newly
adopted clock
 endif
 Stay in Merging Procedure for the next interval
```

Fig. 2. Merging procedure of the new power-saving algorithm

```
Regular Procedure
 Running existing protocols for clock synchronization, power management and
other operations
 if Received special beacon
 then
  Enter Merging Procedure at the end of interval
 else
  Check for neighbor change at the end of interval
  Calculate based on neighbor change
  if Detection probability > Th_n OR discovery counter = Th_d
  then
   Enter Discovery Procedure in the next beacon interval
  else
   Discovery counter++
  endif
 endif
```

Fig. 3. Regular procedure of the new power-saving algorithm

A and B denote the two stations with different clock values and inside each other's range. Without loss of generality, let us assume that A is trying to discover B. From the Partial Synchronization Assumption above, station B will be able to successfully send at least one beacon in $\Delta/2r_{d\_min}$. I also know, from the protocol, that A will stay

Fig. 4. Clock drifting between two nodes

awake for at least a full beacon interval in every Th_d intervals. So B's beacon will be heard by A eventually.

Next, I take a look at the correctness of the merging procedure. Assume two subnetworks have discovered each other. Stations in the sub-network with the faster clock, A, do not need to do anything special. Let o denote the station that discovered the faster clock of A, and belongs to the sub-network B. I try to show that o can populate all stations in B with the faster clock value. For any station, i, in the subnetwork B that does not know the new clock, there exists at least one path from o to i due to the connectivity of the sub-networks. o,i_1, i_2,<, i_n, i denotes the stations on this path. The stations next to each other in this list are in each other's transmission range. According to the protocol in Section 2.1, starting from o, stations adopting the faster clock will broadcast the special beacon in the original beacon window. The Partial Synchronization

assumption guarantees that any station in a sub-network will get at least one chance to successfully broadcast its own beacon in every $\Delta/2r_{d\_min}$. Therefore, in a finite amount of time (no more than $\Delta/2r_{d\_min}$) the special beacon from o will be heard by i_1. Upon receiving the special beacon, i_1 will change its clock and broadcast special beacons to its neighbors including i_2. So unless this process stops prematurely, the faster clock value carried in the special beacons will eventually propagate along the path and reach I.

Next, I show the above process will not stop prematurely. Stations broadcasting the special beacons like o will not exit the merging procedure until they hear no more regular beacon in the original beacon window during a continuous period of $\Delta/2r_{d\_min}$. From the Partial Synchronization Assumption, we know that in such a period, if there is a station that still runs on the original clock in the neighborhood of o, o will hear at least one beacon from it. And if this is the case, according to the protocol, o will not exit the merging procedure.

Last, if the synchronization algorithm in the Partial Synchronization Assumption is not a distributed algorithm. For example, it uses a hierarchical structure to select only some stations as leaders to broadcast beacons, then I shall modify the election process so that station o can force itself to become a leader during the merging procedure.

III. SIMULATION AND RESULT

To confirm my analysis of the relation between clock synchronization and power management, I compare the results on power-saving of different power management algorithms in a simulated environment. I implement my protocol as a example of synchronous power management, and also pick the Periodically-Fully-Awake-Interval protocol from [5] as a representative of asynchronous power management protocols. The Periodically-Fully-Awake-Interval protocol requires a station to stay awake for a full beacon interval in every T intervals, where T is a important parameter of the protocol. To further clarify that the trends shown in the results are caused by clock synchronization, I also include the power-saving scheme described in [10], which uses a scheme running between totally asynchronous algorithms and those seeking global synchronization throughout a system like the one that I propose in this paper, and mark it "local synchronization" in the figures. In my simulation, data traffic is presented as single packet with a fixed length. I describe it using a Bernoulli process with access probability $p_{traffic}$. I do not have any particular requirement for the network topology. Nodes connect with each other randomly, with 20% to 30% of total nodes being neighbors. Node movement is described as another Bernoulli process with moving probability p_{move}. However, since I did not implement any routing procedure, all traffic happens between directly connected neighbors, and the node will not move when it has packets to send. Other simulation parameters are shown in Table 1.

Before comparing different protocols with each other, I first check the behaviors of each protocol under different configurations. I show the effects of the main parameters, T and the number of nodes in the system, of the Periodical-Fully-Awake- Interval protocol under different traffic loads and node mobility in Figure 5. Among the graphs in Figure 5, (a) shows the effect of p_{move} and the distance between the Fully-Awake-Interval, T, with 100 nodes in the system; (b) shows the effect of node number, n, on the node's power efficiency with $p_{move} = 0.2$ and $T = 4$; (c) shows the average packet delay of the same configurations. Not surprisingly, we can see that T has a major effect on the awake ratio in (a). The difference among different T values is clear. On the contrary, the effect of node mobility on the PFAI protocol is almost negligible.

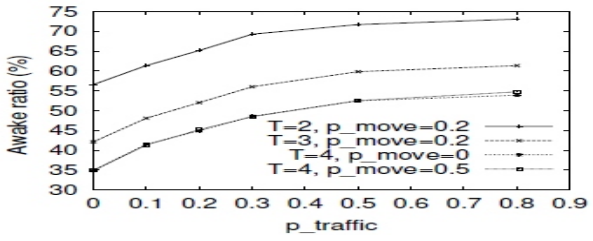
Table 1. Simulation configuration

Parameter	Value
Beacon interval length	2000 slots
Beacon window length	64 slots
ATIM window length	200 slots
Beacon length	2 slots
ATIM packet length	2 slots
Data packet length	40 slots
Min. contention window size, CW_{min}	31 slots
Max. contention window size, CW_{max}	255 slots
Simulation time	1000 beacon intervals

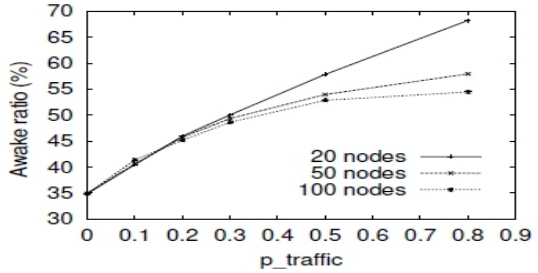
That is because in this protocol, a node's wake-sleep schedule is totally unrelated to other nodes. From (b) and (c), we can see that the PFAI's behavior changes under different traffic loads as the node density of the network changes. When network node density is high (more nodes inside the communication range of a node), the power consumption starts to level off as the traffic load keeps increasing. However, when there are less nodes in the system, the awake ratio (the average percentage of time a node stays awake) increases almost linearly with traffic load. The reason is contention. When node density is high, a heavy traffic load will cause a lot of contentions during ATIM transferring. Those stations that could not send the ATIM successfully will go to sleep until the next beacon interval. That translates to more sleep beacon intervals for a node on average even when a station has a data packet to send. This can be verified from the delay graph in (c). A dense system, n=100, has a much higher

delay than a system that is sparse, n=20. And the delay keeps increasing as traffic load increases. When n=20, the average packet delay does not change much with the traffic load.

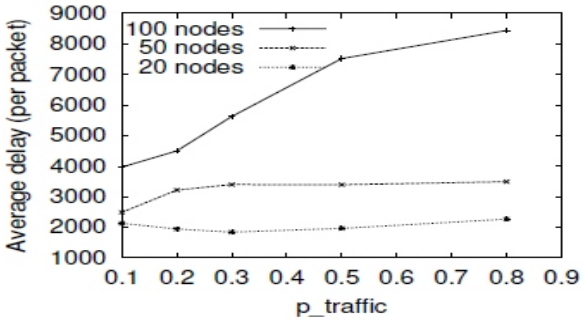
In Figures 6 and 7, we can see the effects of different parameters on the performance of stations with various traffic loads in the power-saving scheme that is based on local clock synchronization. From graph (a) of Figure 6, which shows the effects of mobility, p_{move}, and the percentage of nodes which have multiple schedules, Overlap, on the protocol with n = 100 nodes, s = 20 different schedules in the system, and each node scanning for new neighbors every T = 10 beacon intervals, we can see the algorithm is not sensitive to node mobility. This is because in this protocol, multiple schedules are observed. When some neighbors with schedule A are replaced by neighbors with schedule B, as long as the total number of schedules a node follows is not changing, the node does not have much more work. Overlap has an apparent effect. This is quite straightforward since it controls the percentage of nodes with multiple schedules in the system. These are the nodes that stay awake much longer than those that have only one schedule. Graph (b) shows the effect of the number of different schedules, s. Other parameters are: n = 100, T = 10, p_{move} = 0.5, Overlap = 40%. In it, we can see that the average number of schedules of those multi-schedule nodes following, s, has the most significant effect on the awake ratio. With that number increasing from 2 to 20, when there is no traffic, the average awake ratio of each node climbs from 25% to over 50% although less than half of the nodes are multi-schedule nodes. The proposers of this scheme argue that there would be only a few different schedules in the system. However, since no schedule change is devised,



(a)



(b)

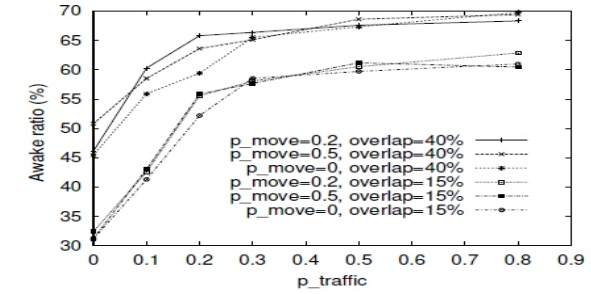


(c)

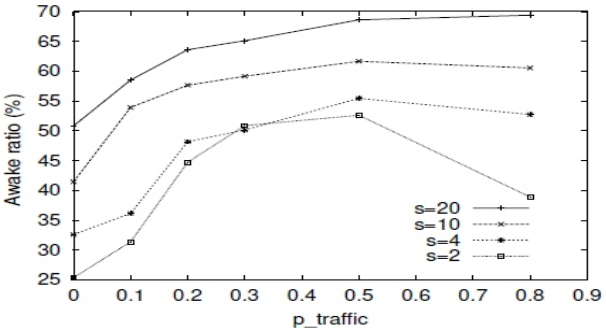
Fig. 5. Simulation results of Periodical-Fully-Awake-Interval protocol

and considering the complex situation of a mobile ad hoc network, I think that making the protocol able to handle a number of different schedules is important. When there are only a few schedules in the system and traffic load is heavy, due to contention, nodes spend less time awake. But the delay per packet increases in these situations. This can be seen in my delay data (it is not shown here, but is similar to the PFAI behavior when the number of nodes in the system increases). The effect of how often a node has to stay awake for the whole beacon interval to scan for different schedules is shown in graph (a) of Figure 7. The graphs in Figure 7: (a) show the effect of the distance between scan intervals, T, with other parameters: n = 100, s = 20 if not specified, p_{move} = 0.5 and Overlap = 15%, and (b) show the effect of the number of nodes in the system, n, with s = 4, T = 20, p_{move} = 0.2 and Overlap = 15%. The difference is clear, but not as significant as the effect of s. And in Figure 7 (b), the number of nodes in the system changes the behavior of the protocol when the traffic is heavy, but it does not have much effect on the awake ratio when the traffic is light. I had the same observations for the PFAI protocol. Both Figure 8 and Figure 9 show my protocol's performance and the effects of the parameters. Graph (a) in Figure 8 shows the effects of mobility, p_{move}, and the percentage of nodes that have multiple schedules, Overlap, on the protocol with n = 100 nodes, s = 20 different schedules in the system, and each node scans for new neighbors every T = 10 beacon intervals. Graph (b) shows the effect of the number of different schedules, s. Other parameters are: n = 100, T = 10, p_{move} = 0.2, Overlap = 15%. In Figure 9, (a) shows the effect of the distance between scan intervals, T, with other parameters: n = 100, s = 20 if not specified, p_{move} = 0.2 and Overlap = 40%; (b) shows the effect of the number of nodes in the system, n, with s = 4, T = 10, p_{move} = 0.2 and and Overlap = 40%. The results in Figure 8 (a) indicate that the protocol is more sensitive to movement than Overlap. This is because clock synchronization and the power-saving schedule merging try to unify all nodes' schedules. Those nodes in the

overlap area (with multiple schedules) eventually reduce to one schedule only.

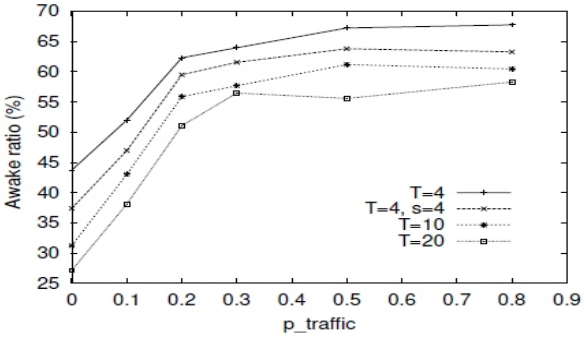


(a)

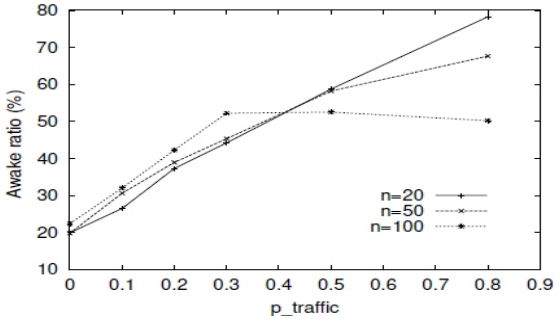


(b)

Fig. 6. Simulation results of power-saving scheme based on local synchronization (part I)



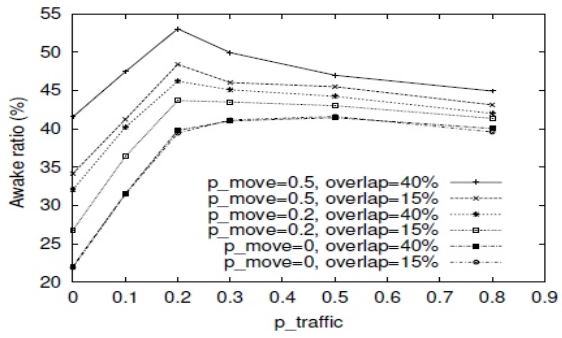
(a)



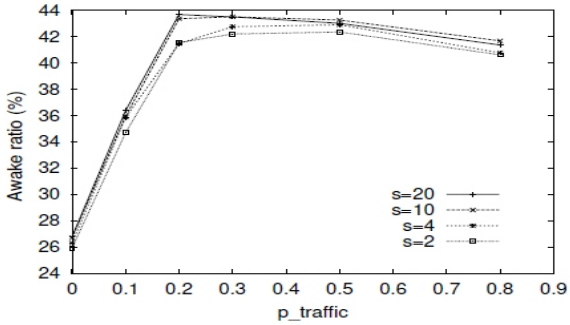
(b)

Fig. 7. Simulation results of power-saving scheme based on local synchronization (part II)

However, when node mobility is high, new schedules are generated due to node movements. It takes more effort to keep them all synchronized. This schedule merging effect can also be seen clearly in graph (b) of Figure 8, which shows that the number of different schedules has a much smaller impact on the results. For Figure 9, in (a), T becomes one of the main parameters that will define the power performance in this protocol. Although the awake ratio changes a lot from T = 4 to T = 10, it stays very close whether the nodes search for new neighbors at least every 10 beacon intervals, or every 20 beacon intervals. This is expected since periodical detection is not the only way my protocol decides when to execute the detection procedure. The protocol also monitors neighbor change and starts the detection procedure based on this change. The simulation results show that after a certain point, the estimation used in the protocol for neighbor discovery is pretty effective. Again in (b), it is shown that the effect of the number of nodes in the system on power-savings is the same as in the other two protocols. When I put all three protocols together for comparison, I unveiled some interesting results. First, I compare them in a highly mobile system with p_{move} = 0.5, i.e. around half of the nodes change their position (and clock due to loss of synchronization), in this group of figures. In Figure 10, we can see a saturated system (100 nodes) in which the awake ratio goes up and then goes back down, and the delay keeps increasing, as

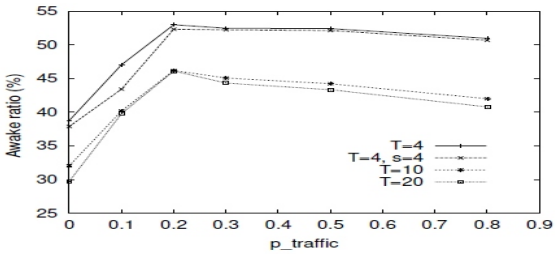


(a)

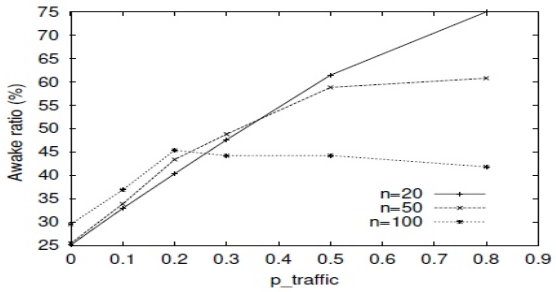


(b)

Fig. 8. Simulation results of my protocol (part I)



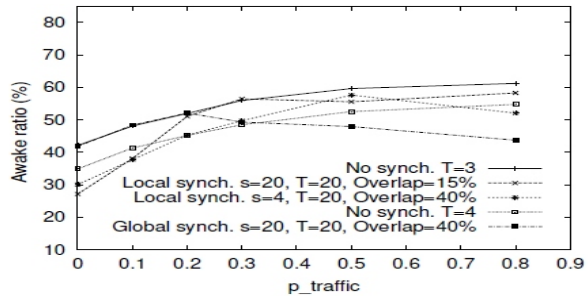
(a)



(b)

Fig. 9. Simulation results of my protocol (part II)

traffic load increases. When there are only 20 nodes in Figure 11, the system has not reached its saturation point. The awake ratio becomes higher and higher to accommodate the increasing traffic, while delay does not change much. Among the three protocols, the Periodically-Fully-Awake-Interval protocol shows some stability. It responds to most of the parameters slowly, except for the distance between the Fully-Awake-Intervals, T. It has the highest awake ratio when the traffic is very light, because of the short period between the Fully-Awake-Intervals. With the increase of traffic load, both awake ratio and packet delay increase, but at a relatively low rate. Although the scheme based on local clock synchronization has a lower awake ratio and delay value in a light traffic situation, $p_{traffic}$ ¨ 0.2 in this case, it surpasses PF AI with T = 4 when traffic picks up. And the delay increment is even more dramatic. This can be contributed to the high contentions coming from issues with multiple schedules that I pointed out in previous sections. My protocol shows the best result overall. Because of the clock synchronization and the merged schedule, its performance is close to the underlying CSMA/CA MAC protocol. After the system reaches its saturation point, the power-saving scheme saves energy by preventing data transmissions from increasing, since it can only introduce more contention rather than getting more packets through. And the packet delay does not make a sudden jump, as it does in the scheme based on local synchronization. In Figures 12, 13, 14, and 15, I show the comparison in a low mobility (p_{move} = 0.2) and stable environment (p_{move} = 0). In a low mobility and stable environment, the protocols perform similarly. With less changes happening in the system, the saturation point is pushed to a heavier traffic load. This leads to a little improvement of the scheme based on local clock synchronization when comparing to the PFAI protocol. One interesting point I want to make here is the packet delay of the PFAI in a dense network, i.e. n = 100. We notice that only at a very light traffic load, $p_{traffic}$ ⩽0.1 or even less depending on than a larger T value. For most of the configurations, the T = 4 delay curve is below, i.e. delay is shorter than, the T = 3 curve. And for the awake ratio, T = 4 is still significantly less than T = 3. After tracing different configurations of traffic loads and T node mobility, a smaller T value shows a lower packet delay values, I found that the longer delay is due to higher contention with a smaller T. A smaller T value means a packet is more likely being transferred. But when the system is saturated,



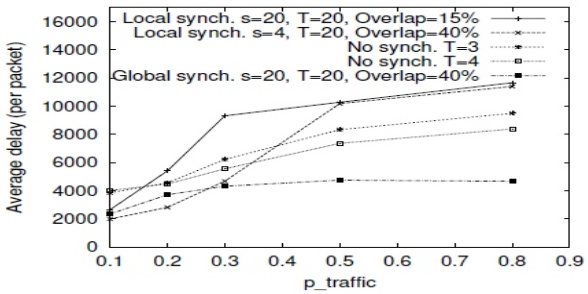
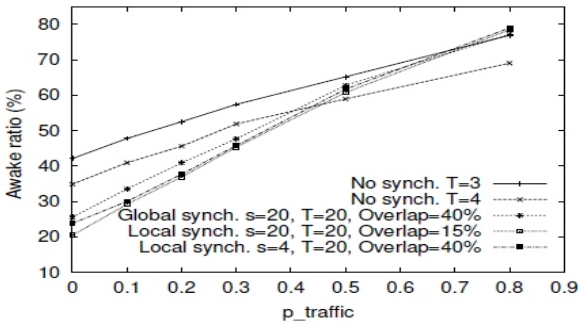


Fig. 10. Comparison of protocols at high mobility with 100 nodes



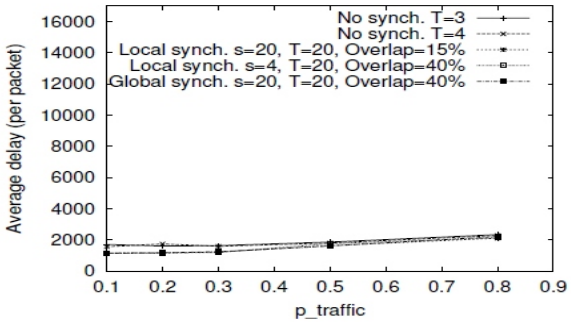


Fig. 11. Comparison of protocols at high mobility with 20 nodes

putting more packets into transmission will not do any good. So from this we can see that the parameter T is very important in PFAI.

IV. CONCLUSION

From studying the relationship between clock synchronization and power-saving schemes, I showed that clock information is very important for the efficiency of power-saving protocols which are based on the IEEE 802.11 power management procedure. Based on this observation, I presented an integrated protocol for both clock synchronization and power-saving for mobile ad hoc networks. In this protocol, I addressed the problems that rise from topology change introduced by node mobility. My protocol can handle the discovery and merging of new sub-networks with different clock values.

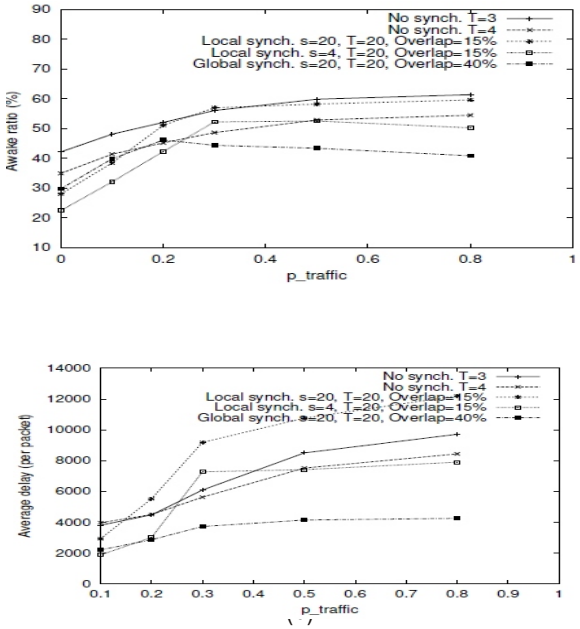


Fig. 12. Comparison of protocols at low mobility with 100 nodes

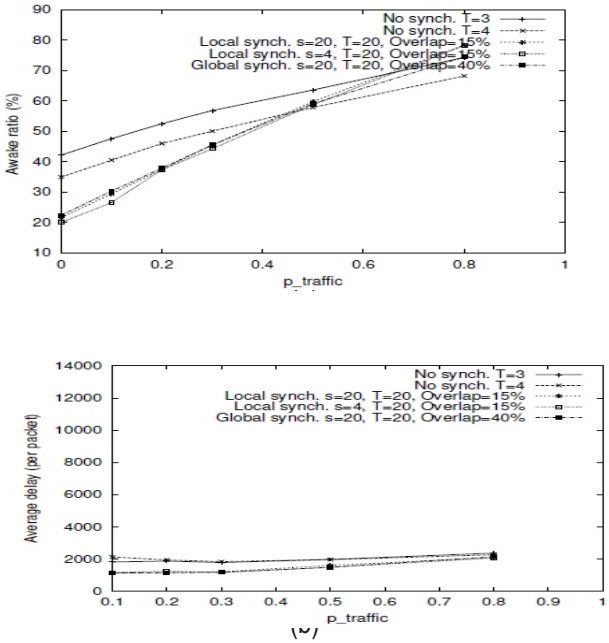


Fig. 13. Comparison of protocols at low mobility with 20 nodes

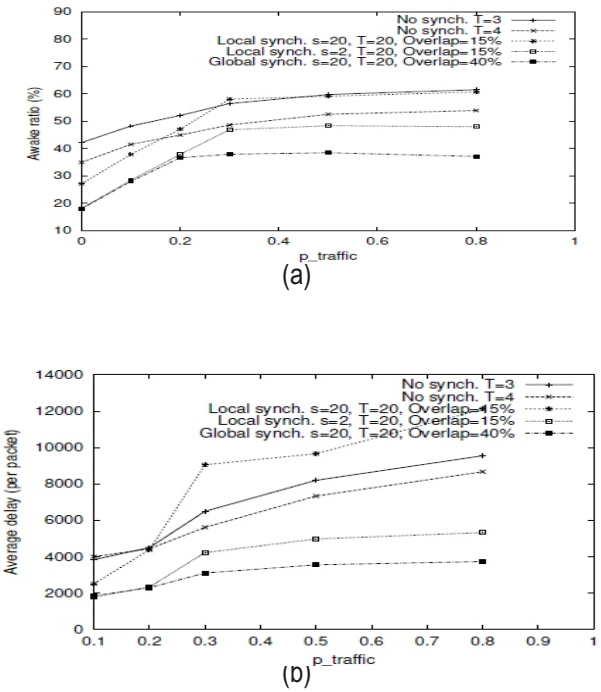


Fig. 14. Comparison of protocols at no mobility with 100 nodes

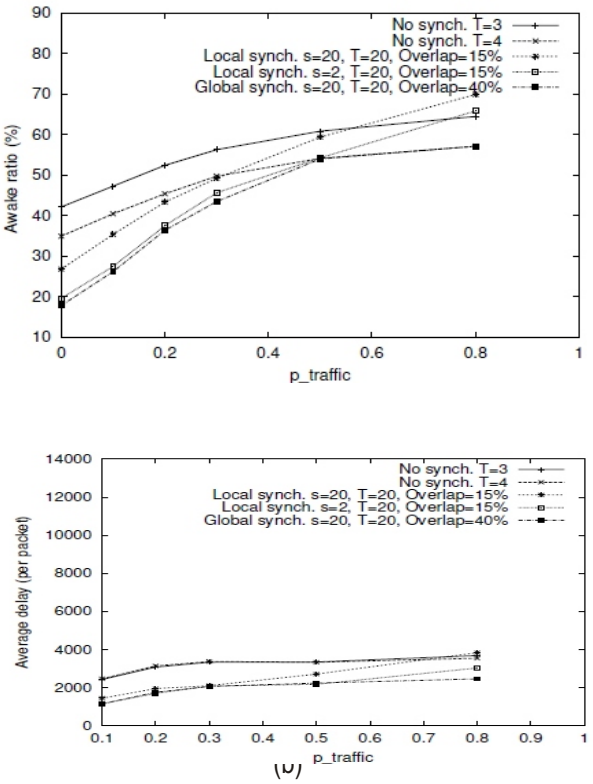


Fig. 15. Comparison of protocols at no mobility with 50 nodes

From the simulation results, we can see the protocol performs more efficient power-saving than other protocols of its type.

REFERENCES

[1] Mahesh K. Marina, George D. Kondylis, and Ulas C. Kozat, 2001, "Rbrp: a robust broadcast reservation protocol for mobile ad hoc networks",In Proceedings of IEEE International Conference on ommunications volume 3, pp. 878–885.

[2] Shugong Xu and Tarek Saadawi, 2001, "Does the ieee 802.11 mac protocol work well in multihop wireless ad hoc networks?", IEEE Communications, 39(6):130–137.

[3] Shugong Xu and Tarek Saadawi, 2001, "Revealing and solving the tcp instability problem in 802.11 based multi-hop mobile ad hoc networks". In Porceedings of IEEE VTS 54th Vehicular Technology Conference, volume 1, pp. 257–261.

[4] Chi-Fu Huang, Yu-Chee Tseng, Shih-Lin Wu, and Jang-Ping Sheu, 2001, "Increasing the throughput of multihop packet radio networks with power adjustment",In Preceedings of the Tenth International Conference on Computer Communications and Networks , pp. 220–225.

[5] Yu-Chee Tseng, Chi-Shun Hsu, and Ten-Yueng Hsieh, 2002, "Power-saving protocols for ieee 802.11-based multi-hop ad hoc network",In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies volume 1, pp. 200–209.

[6] Rong Zheng, Jennifer C. Hou, and Lui Sha, 2003, "Asynchronous wakeup for ad hoc networks", In Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing, pp. 35–45.

[7] Hagen Woesner, Jean-Pierre Ebert, Morten Schl±ager, and Adam Wolisz, 1998, "Power-saving mechanisms in emerging standards for wireless lans: The mac level perspective". IEEE Personal Communications, 5(3):40–48

[8] Chi-Fu Huang, Yu-Chee Tseng, Shih-Lin Wu, and Jang-Ping Sheu, 2001, "Increasing the throughput of multihop packet radio networks with power adjustment",In Preceedings of the Tenth International Conference on Computer Communications and Networks , pp. 220–225.

[9] Ten H. Lai and Dong Zhou, "A scalable timing synchronization protocol via constructing connected dominating sets in ieee 802.11-based multihop ad hoc networks". Submitted for publication.

[10] Wei Ye, John Heideman, and Deborah Estrin, 2002, "An energy-efficient mac protocol for wireless sensor networks", In Proceedings of the 21st Annual Joint Conference of the IEEE Computer and Communications Societies volume 1,pp. 200–209.