Distributed synchronization and medium access in wireless mesh networks

Sriram Venkateswaran*, Sumit Singh*, Upamanyu Madhow*, Raghu Mudumbai†
* University of California, Santa Barbara, CA 93106, USA
†The University of Iowa, Iowa City, IA 52242, USA

Abstract—**Implicit local coordination of nodes in a wireless network using mechanisms such as Carrier Sense Multiple Access (CSMA) is conceptually attractive and relatively easy to implement, but often leads to performance that is far inferior to what is possible using explicit global coordination strategies such as Time Division Multiplexing (TDM). In this paper, we give two examples showing that appropriately designed implicit coordination strategies that employ learning and memory can provide performance competitive with that obtained using explicit strategies, while requiring minimal overhead. The first example is an algorithm for distributed timing synchronization maintenance using the timing information already present in ongoing communication in the network. The second example is a distributed medium access control protocol that achieves performance close to time division multiplexing (TDM) without requiring explicit resource allocation: nodes lock into communication patterns that have been found to work, with enough randomization to prevent locking into poor schedules. While the general philosophy of exploiting learning and memory in the design of network protocols is of broad applicability, our numerical results emphasize 60 GHz networks with highly directional links: effective coordination is particularly important for such networks, in view of the "deafness" caused by directionality.**

I. Introduction

Cellular networks and wireless local area networks (WLANs) offer two radically different approaches to wireless networking. Tight centralized control, with significant use of feedback signaling (e.g., for power control and channel state information), is used in cellular networks in order to ensure efficient usage of precious resources such as power and bandwidth. On the other hand, WLANs today use decentralized mechanisms such as Carrier Sense Multiple Access (CSMA), enabling implicit coordination between nodes. While such techniques are easy to implement and quite efficient for small networks, they lead to highly inefficient resource usage in larger networks (e.g., when a mesh network is constructed out of WiFi nodes). In this paper, we provide two examples to make the case that implicit coordination need not lead to inefficiency, showing that decentralized mechanisms that use learning and memory can be as effective as centralized and explicit coordination mechanisms, while minimizing the overhead required for coordination.

Our first example is that of maintaining network-wide timing synchronization for the purpose of time slotted multiple access. We show that it is possible to exploit the timing information implicit in ongoing communication to jointly adapt the clock phases and rates at each node. Each time a node receives a packet, it measures the difference between the *expected* and *actual* time of reception in order to estimate the difference between its clock and that of the transmitter. A node adjusts its clock phase each time it receives a packet, but adapts its rate on a slower timescale by exploiting memory. We show the efficacy of this decentralized scheme for typical traffic patterns, and also investigate the minimum amount of communication required for it to work. The latter also sheds light on the minimal overhead required for maintenance of synchrony for networks using explicit synchronization beacons (which might be necessary for networks with very sparse communication patterns, such as sensor networks with severe energy constraints).

The proposed algorithm has some similarity to prior consensus [1] style algorithms [2]–[4], but the latter require explicit signaling. Firefly-inspired synchronization [5] can be adapted for implicit synchronization, but it is only designed for phase synchrony, and does not handle either propagation delays or oscillator skew.

Our second example is a distributed MAC protocol, first introduced in [6], that employs memory to achieve implicit transmit-receive coordination and efficient spatial reuse, assuming that network-wide time slotting is available (e.g., using the synchronization maintenance algorithm in our first example). While this approach is generally applicable, we focus on the specific example of a 60 GHz network with highly directional links, where implicit coordination is challenging because of the deafness resulting from directionality. We show that a node's transmit and receive history with each neighbor provides feedback for implicit coordination, and persistent use of given slots for communication with a given neighbor leads to approximate TDM schedules with high medium utilization, without incurring the overhead of explicit network-wide coordination.

Prior work on using memory for implicit coordination for MAC includes [7], which focused on omnidirectional networks carrying periodic traffic. Networking for mesh networks with directional links has been studied before (e.g., see [8]–[13]), but typically two kinds of links are used: omnidirectional links for coordination and then directional links for enhanced performance. In contrast, the proposed MAC provides a single mechanism for implicit coordination which is broadly applicable. While we focus on illustrating how well this works for highly directional links (where prior protocols would fail), we believe that further investigation would show that similar ideas work for enhancing the performance networks with omnidirectional links as well ([7] provides a glimpse of this in a somewhat constrained setting of voice over mesh networks).

Putting these two examples together yields a framework for efficient resource utilization in large-scale mesh networks, using time slotted multiple access, but a detailed simulation of such a system is beyond the scope of the present paper.

II. Distributed Implicit Synchronization

Synchronizing nodes to the accuracy required for communication is typically done in two stages – first, by *establishing* and then, by *maintaining* synchrony. At startup, the nodes are assumed to be completely asynchronous; therefore, an explicit synchronization mechanism is used to synchronize the network coarsely. For example, a gateway node broadcasts its time, enabling its one-hop neighbors to set their clocks. These nodes then broadcast their times, enabling nodes further away from the gateway to synchronize their clocks. By this process, the network can be coarsely synchronized in a time interval proportional to the diameter of the network. Since network setup times on the order of a few seconds – much larger than the timescales of communication – are acceptable, and establishing synchrony is a one-time effort, this process is not expected to be a bottleneck. The main challenge in *maintaining* synchrony is the variation in clock rates across nodes because of manufacturing imperfections and temperature changes. They cause the clock phases at different nodes to drift apart, tending to destroy the established synchrony. We propose an algorithm to restore synchrony by leveraging the timing information present in the ongoing communication, thereby minimizing the need for explicit synchronization beacons.

Synchronization based on implicit timestamps: Suppose that node $\mathcal{N}_i$ transmits to its neighboring node $\mathcal{N}_j$ in slot s. Since this is a TDM-based network, the transmission begins when $\mathcal{N}_i$'s clock equals sT_{slot}. Suppose that the time at which $\mathcal{N}_j$ starts receiving this packet is φ_j (after subtracting out the propagation and processing delays that can be estimated in the startup phase). $\mathcal{N}_j$ can now conclude that its clock is ahead of $\mathcal{N}_i$'s clock by $\varphi_j - sT_{slot}$. Thus, $\mathcal{N}_j$ obtains an estimate of its clock error with respect to $\mathcal{N}_i$ *without any explicit timing related signaling*. The receiving node $\mathcal{N}_j$ adjusts its clock phase and frequency based on such implicit timing error measurements in order to drive the network towards synchrony. Note that such adjustments are coupled through the network transmission schedule – the changes made by $\mathcal{N}_j$ based on the timing of its received messages impact the times at which it transmits, and hence the adjustments made by nodes who receive these transmissions. We now state the phase and frequency adjustment rules used by the nodes and then provide the intuition behind them (also see Figure 1).

- **Phase Adjustments:** Suppose that $\mathcal{N}_i$ transmits to $\mathcal{N}_j$ in slot s. Let the times on $\mathcal{N}_i$'s and $\mathcal{N}_j$'s clocks when $\mathcal{N}_i$ begins this transmission be $\varphi_i[s]$ and $\varphi_j[s]$ respectively. $\mathcal{N}_j$ implicitly estimates that its clock is ahead of $\mathcal{N}_i$'s clock by $\varphi_j[s] - \varphi_i[s]$ and makes an instantaneous phase jump in order to reduce the phase error with respect to $\mathcal{N}_i$ "quickly". Denoting the clock phase at $\mathcal{N}_j$ before and after the phase jump by $\varphi_j^-[s]$ and

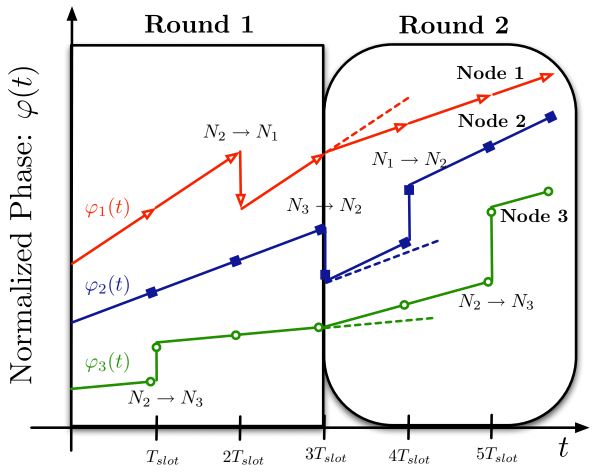


Fig. 1. Nodes make *phase jumps* each time they receive a packet. However, they change their frequencies (slope of the lines) only at the end of a *round* consisting of many slots.

$\varphi_j^+[s]$ respectively, we have,

$$\varphi_j^+[s] = \varphi_j^-[s] + \beta(\varphi_i[s] - \varphi_j^-[s]) \tag{1}$$

where $0 < \beta < 1$ is a tunable parameter.

While this phase adjustment reduces the phase error between $\mathcal{N}_j$ and $\mathcal{N}_i$, it might worsen the phase error between $\mathcal{N}_j$ and a different neighbor $\mathcal{N}_k$. Nevertheless, with "sufficient" bidirectional communication and all nodes making such adjustments, these phase jumps tend to keep the synchronization error between neighboring nodes under control.

- **Frequency Adjustments:** In addition to the phase adjustments, nodes also adjust their frequencies based on the implicit timing error measurements so to attain the long-term benefits of network-wide synchrony in both *phase* and *frequency*. However, the frequency adjustments occur on a slower timescale when compared to the phase adjustments – nodes adjust their frequencies only once per *round*, consisting of a large number of slots. The frequency adjustment rules are intuitive and can be summarized as follows: Each node looks for a trend in the phase errors it observes with its neighbors over a round of slots. If, despite all the phase adjustments it makes, node $\mathcal{N}_i$ consistently finds that its clock is "well ahead" of its neighbors' clocks over a round, $\mathcal{N}_i$ concludes that its clock frequency is larger than the average frequency in the network. Node $\mathcal{N}_i$ then reduces its clock frequency by μ at the end of the round. Similarly, nodes which find their clock phases "well behind" those of their neighbors', increase their frequency by μ at the end of a round. This process is repeated over multiple rounds to achieve frequency synchrony.

Let node $\mathcal{N}_i$ receive implicit timestamps from its neighboring node $\mathcal{N}_j$ in n_{ij} slots within a round that consists of S_R slots. We label these slots $s_{j\to i}(1), \ldots, s_{j\to i}(t), \ldots, s_{j\to i}(n_{ij})$ respectively. Node $\mathcal{N}_i$ adjusts its frequency at the end of the round of S_R slots

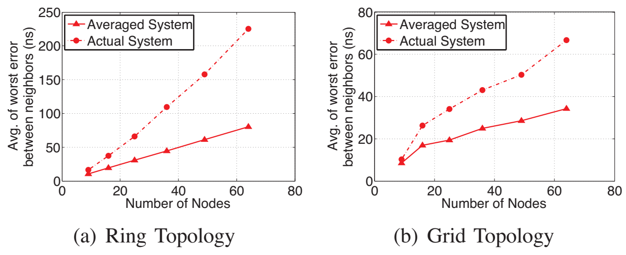


Fig. 2. Worst error between neighbors for the actual system and the averaged system with only phase adjustments in a *directional network*.

based on an estimate of its *excess frequency* – the difference between its clock frequency F_i and the network wide average $\overline{F}$. $\mathcal{N}_i$ estimates its excess frequency, denoted by $\hat{\delta}_i$, by the sum of the observed phase errors *with all its neighbors over the entire round*,

$$\hat{\delta}_i = \frac{\beta}{S_R} \sum_{j \in Nbr(i)} \sum_{t=1}^{n_{ij}} \varphi_i[s_{j \to i}(t)] - \varphi_j[s_{j \to i}(t)] \tag{2}$$

If the excess frequency estimate $\hat{\delta}_i$ falls within a "dead zone" of width ϵ (i.e. $|\hat{\delta}_i| \leq \epsilon$), node $\mathcal{N}_i$ is ambivalent about whether its frequency exceeds/falls below the average frequency in the network. In this scenario, node $\mathcal{N}_i$ does not change its frequency. However, when node $\mathcal{N}_i$ is confident that its phases are "well ahead" of its neighbors' ($\hat{\delta}_i > \epsilon$), it decreases its frequency by μ (and vice versa for $\hat{\delta}_i < -\epsilon$).

Convergence: An exact analysis of this algorithm is complicated by the fact that the neighbor from whom a node receives packets varies across slots. However, we can provide fundamental theoretical insight by analyzing the evolution of the system "on the average". In such a fictitious *averaged system*, each node adjusts its clock phase in *every* slot based on a weighted average of the phases of *all* its neighbors. The weights are chosen based on the transmission schedule in the actual system, with clock phases of neighbors who communicate more frequently being given a larger weight. We show that the rules described above for phase and frequency adjustments lead to synchrony in the averaged system and also obtain rules of thumb to choose the parameters ϵ and μ. We do this in two stages : first, by choosing a round to be "sufficiently long", we show that each node can estimate its excess frequency – based on (2) – to an arbitrary accuracy χ. We then choose the width of the dead-zone ϵ to be larger than the sum of the frequency adjustment step-size μ and frequency estimation error χ, so that we can accommodate wrong frequency steps and frequency estimation errors. From this, we can show that the frequencies converge to a window of width smaller than $2(\epsilon + \mu + \chi)$. Since μ and χ can be chosen freely – and therefore, made arbitrarily small – we can achieve frequency synchrony to any desired accuracy.

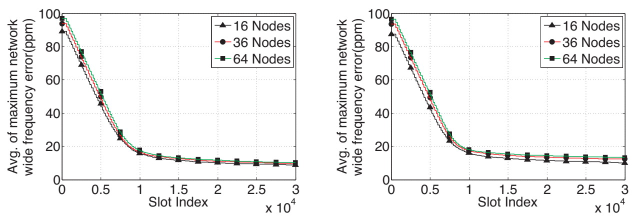
Overhead due to propagation delay: We now quantify the fundamental overhead due to propagation delay that serves as a benchmark against which we can characterize the performance of our algorithm. Consider a network consisting of three nodes, labeled A, B and C, equipped with synchronized clocks and operating in a Time Division Multiplexed (TDM) fashion. We assume that neighboring nodes are located within a distance d_{max} of one another. Suppose that the packets transmitted in each slot are of duration $W \leq T_{slot}$ and the distance between A and B is d_{AB}. Consider a scenario where B receives a packet from A in slot s and then, transmits to C in slot $s+1$. Therefore, A begins its transmission at $t = sT_{slot}$, and B *completes* reception of this packet after an interval of $W + \frac{d_{AB}}{c}$ (corresponding to the sum of the propagation delay and the packet duration) at $t_{end} = sT_{slot} + W + \frac{d_{AB}}{c}$, where c denotes the speed of light. Because of the half-duplex constraint, B must complete reception from node A, in order to be "ready" for the transmission in the subsequent slot to C, beginning at $t_{start} = (s+1)T_{slot}$. Therefore, we need $t_{end} \leq t_{start}$, or,

$$T_{slot} \geq W + \frac{d_{AB}}{c} \tag{3}$$

Setting the distance between A and B to the maximum value d_{max}, we see that a slot must include a "silence period", *at least* of duration d_{max}/c, in addition to the data transmission time W. Indeed, we can show that a silence period of d_{max}/c is sufficient to maintain TDM-based communication in a general network with half duplex nodes. Therefore, the fundamental overhead due to propagation delay, called the *guard time*, is $\tau_{guard} = d_{max}/c$. For the envisioned mesh network with a maximum link range $d_{max} = 100$ m, the guard time $\tau_{guard} = 1/3\ \mu$s ≈ 333 ns.

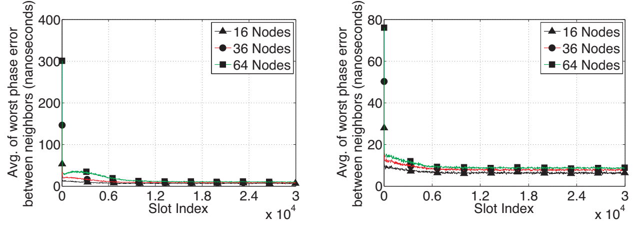
The guard time derived above is sufficient to sustain communication with ideal clocks – clocks that are synchronized in phase across nodes and run at the same frequency. However, some additional guard time is necessary to tolerate synchronization errors and sustain communication until the network reaches synchrony (via the implicit synchronization scheme). We use this additional guard time as a metric to quantify the performance of our algorithm by comparing it against the overhead due to propagation delay.

Scalability of phase-only updates: We first investigate a scheme where nodes only adjust their phases and not their frequencies (a special case of phase-frequency adjustments with $\mu = 0$). This scheme is particularly attractive from an implementation standpoint because of its simplicity – when the clocks are implemented in software, recurrent phase adjustments simply correspond to altering the contents of a register occasionally. However, the downside to such phase-only adjustments is that the clock phases can never converge completely and always leave behind an irreducible phase error. The magnitude of this residual phase error between neighboring nodes is precisely the additional guard time required for successful communication. It depends on the distribution of frequencies across nodes in the network, in addition to the network size and topology. For a given network topology, we estimate the guard time conservatively by choosing a set of "bad" frequencies that *maximize* the phase error between neighbors. The bad frequencies are chosen using a linear program that optimizes the phase error between neighbors



(a) Ring topology (b) Grid topology

Fig. 3. Network wide frequency error with ring and grid topologies.



(a) Ring Topology (b) Grid Topology

Fig. 4. Worst phase error between neighbors in ring and grid topologies.

in the averaged system. This provides a lower bound to the worst-case error in the actual system, averaged over multiple realizations of the random transmission schedule.

We present the results for networks of varying sizes, set in ring and grid topologies, with parameters typical of 60 GHz networks; for example, we choose the slot duration $T_{slot} = 10\mu$s and the skews at different nodes range from -50 ppm to 50 ppm. From Figure 2, we make two immediate observations – (1) the averaged system provides a lower bound to the phase errors in the actual system and (2) the worst-case phase error *between neighbors* grows with the size of the network. The worst-case phase error grows with the number of nodes N as $N^{1.33}$ and $N^{0.875}$ for networks set in ring and grid topologies respectively.

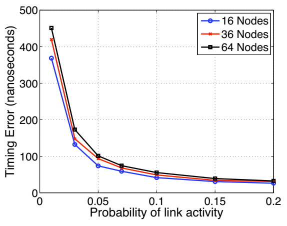
From Figure 2, we see that the error between neighbors in a 64 node network set in a ring topology can be as large as 220 ns. Thus, the additional guard time needed to accommodate synchronization errors in such networks is $2/3^{rds}$ of the fundamental overhead due to propagation delay (333 ns). On the other hand, for small directional networks in a ring topology (say, 16 nodes) or reasonably large networks in a grid topology (36 nodes), we see that the largest phase error between neighbors is only 40 ns. In this case, the increase in the overhead is only 12 %, which is perfectly tolerable. To summarize, phase-only adjustments with implicit timestamps may suffice in small mesh networks with linear topologies or moderately sized meshes in grid topologies, but frequency adjustments are necessary for large networks, especially in linear topologies.

Phase-frequency adjustments in saturated networks: To quantify the performance of the proposed phase-frequency adjustment scheme, we consider a saturated network – one with a lot of ongoing communication. Such networks are well-matched to the proposed implicit timing synchronization scheme – since a large number of packets flow through the network, nodes are able to make implicit timing error measurements frequently. Therefore, nodes adjust their clock phases and frequencies often, thereby increasing the rate of convergence towards synchrony.

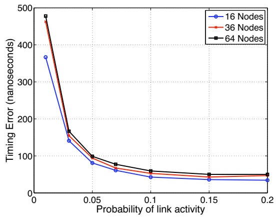
We simulate mesh networks set in ring and grid topologies with the following parameters – noise chosen uniformly from [-5 ns, 5 ns] is added to each implicit timestamp measurement and nodes adjust their frequencies once in 200 slots. The results are shown in Figures 3 and 4 and we make the following observations : (1) The maximum difference in frequencies between any two nodes in the network settles to about 10 ppm (Figures 3(a) and 3(b)) and this value is virtually independent of the size and topology of the network. Since the network-wide frequency error is on the order of 100 ppm initially, the phase-frequency adjustment algorithm decreases the frequency error by a factor of 10. (2) After the phase-frequency adjustments, the maximum phase error between any pair of neighbors in the network is only between 6.25 ns and 10 ns (Figures 4(a) and 4(b)). Crucially, the largest phase error between neighbors does not scale up with the size of the network. Thus, the additional guard time required to tolerate synchronization errors after the phase-frequency adjustments is only 3 % of the fundamental overhead due to propagation delay. Therefore, the proposed algorithm synchronizes saturated networks without any explicit timing related signaling.

Communication required for implicit synchronization: When there is little ongoing communication in the network, the time interval between successive occasions on which node $\mathcal{N}_i$ receives a data packet from its neighboring node $\mathcal{N}_j$ can be quite large. In this interval, uncompensated frequency differences between $\mathcal{N}_i$ and $\mathcal{N}_j$ can cause $\mathcal{N}_i$'s phase to drift away from that of $\mathcal{N}_j$, leading to substantial phase error. Unless the guard interval is large enough to tolerate such large errors, communication between $\mathcal{N}_i$ and $\mathcal{N}_j$ will be unsuccessful. Repeated instances of unsuccessful communication causes the system to spiral into complete breakdown – nodes will no longer receive implicit timestamps (since this relies on successful communication), causing the synchronization accuracy to worsen further. Therefore, choosing an adequate guard time is of paramount importance and we now characterize the necessary guard time as a function of the amount of communication in the network.

We consider mesh networks of 16, 36 and 64 nodes set in ring and grid topologies. We designate some of these nodes to be gateways so that no node is more than 3 hops away from a gateway. We vary the probability of a node having a flow to/from its gateway to control the amount of data flowing through the network. First, we allow the nodes to only adjust their phases and calculate the largest phase error between any pair of neighbors (this is exactly the additional guard time needed to sustain communication). From Figure 5, we see that the phase error decreases with increasing probability of a link being active. Let p denote the probability that a link is active. When $p = 0.01$, we see that the phase error can be as large as 460 ns, which is nearly 1.5 times

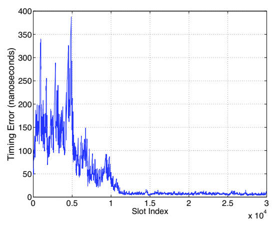


(a) Ring topology

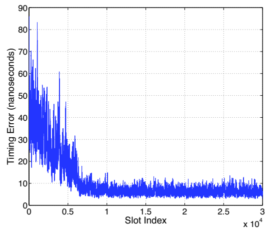


(b) Grid topology

Fig. 5. Phase error between neighboring nodes as a function of the communication in the network.



(a) p = 0.01



(b) p = 0.05

Fig. 6. Evolution of the phase error over many slots with phase-frequency adjustments.

the fundamental overhead due to propagation delay. On the other hand, when $p = 0.1$, the phase error is 50 ns, which is only 15 % of the overhead due to propagation delay. Thus, each link needs to be active roughly 10% of the time for the additional guard time to be tolerable – the activity on the link could be due to data packets flowing through the network or explicit synchronization beacons. Designing such explicit synchronization beacons to augment the ongoing communication is an open issue.

Next, we consider phase-frequency adjustments in a 36-node network set in a grid topology. We plot the largest phase error between any two neighbors in a single realization in Figures 6(a) and 6(b) for $p = 0.01$ and $p = 0.05$ respectively. We see that the phase errors converge to 10 ns (approximately) – only 3 % of the overhead due to propagation delay – in both cases, illustrating the benefits of frequency adjustments.

We now present a MAC protocol that uses the slots constructed by the synchronization scheme and simple learning rules to converge to TDM-like schedules.

III. Memory-guided Directional MAC (MDMAC)

Because of deafness arising from the high directionality of 60 GHz millimeter (mm) wave links, carrier sensing becomes ineffective for monitoring transmission activity in a node's neighborhood. Consequently, the only information readily available to a 60 GHz mesh node is the outcome of its own transmit and receive attempts to its neighbors. The novelty of MDMAC lies in employing simple learning rules at each node based solely on the memory of its transmit/receive outcomes to converge to time division multiplexing (TDM) like schedules. Given the low spatial interference, nodes do not account for the effect of their transmissions on their neighbors. However, mechanisms for dynamically adapting TDM schedules are built in. This allows MDMAC to promptly react to changes in traffic patterns and interference loss (in the few cases when it does occur), and helps avoid locking into grossly unfair schedules.

We now outline the functioning of MDMAC: say node A wishes to initiate a transmission to node B. Node A randomly picks one of the free slots in a frame to attempt a packet transmission. If node B successfully receives the packet, it responds with an ACK to A, and both nodes mark the slot to be used for communication from A to B over future frames. This leads to an implicit slot reservation. In case A's initial transmit attempt to B fails, A flags the slot as "blacklisted" for future transmit attempts to B. The A$\rightarrow$B reservation persists until A has no more packets, or there is repeated packet loss, or if either A or B explicitly terminates the reservation. Such a simple approach, when followed by all nodes, can lead to an approximate TDM schedule, but it suffers from a number of unfairness issues. For example, a node that starts late can get locked out in case of a saturated network. To avoid locking into grossly unfair schedules, we introduce a probabilistic state reset mechanism, where each node resets a slot state (transmit/ receive/ blacklisted for a given neighbor) with a non-zero probability. This randomization of state life-times results in enough churn that allows the TDM schedules to be rearranged.

Approximate Protocol Model: We now present a Markov chain fixed-point analysis to study the effect of different protocol parameters on the performance and compute the expected medium utilization for MDMAC.

We focus on a "typical" node (e.g., in the interior of a large network) with pseudowired links to all its neighbors. Each node maintains state information on each of its outgoing and incoming links for each slot. We assume that the state for each time-slot in a single frame evolves independently, and develop a Markov model for a given slot over multiple frames. Considering a saturated network where each node always has packets for all its neighbors, we assume that the schedule activated over each slot is chosen randomly and independently from other slots.

Outgoing links can be in "Transmit" (T), "Idle" (I) or "Blocked" (B) state, where a node can contend for a new reservation only if the outgoing link is in the "Idle" state. For incoming links, the state space comprises only two states: "Receive" and "Idle". We introduce the "Unavailable" (U) state, which means that the outgoing link cannot contend for that slot because some other link belonging to that node is in the "Transmit" or "Receive" state. The "Unavailable" state allows us to capture and decouple the dependence of state of one link on the states of all the other links within a single node. Having augmented the link state space with the "Unavailable" state, we approximate the states of different links for a given node (in a given time-slot) as independent. We also approximate the state of the links of a node as independent of the state of the links for all other nodes. Clearly, this is not strictly true: a "Transmit" state at node A sending to node B automatically implies that node B is in "Receive" state for

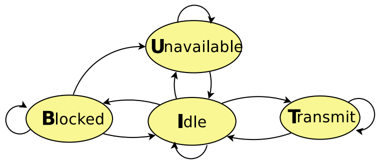


Fig. 7. MDMAC model: state diagram for an outgoing link.

that slot. The preceding decoupling approximations allow us to focus on state transitions for a typical outgoing link to obtain performance insights, which are close to the simulation results presented later in this section.

Fig. 7 presents the state diagram for an outgoing link. Let P_s be the steady state probabilities of state $s \in \{T, I, B, U\}$. We denote the transition probability from s_1 to s_2 as $P_{s_1 s_2}$. Let N denote the number of neighbors for a "typical" node. We now introduce a new tunable protocol parameter: the *listening probability* p_l which is the probability that a node decides not to contend for transmission on any of its outgoing links. Thus, the probability that a node with all links idle chooses to transmit over a given link (each link is chosen with equal probability) is p_{tx}/N, where $p_{tx} = 1 - p_l$. The steady-state probabilities for any state s satisfy:

$$P_s = \sum_{s'} P_{s's} P_{s'} \tag{4}$$

Assuming that the neighbors of the typical nodes are themselves typical, we first consider a simple two-node network.

Modeling a Two-Node Network: For an outgoing link from a node, the Unavailable state means that the incoming link from the neighbor is active, the Idle state means that the incoming link is Blocked or Idle, and the Blocked state means that the incoming link is Blocked or Idle. To model probabilistic state resets, let T_{slot} and T_{block} denote the average transmit/receive slot and blocked slot lifetimes. Therefore,

$$P_{TI} = \frac{1}{T_{slot}},\ P_{UI} = \frac{1}{T_{slot}},\ \text{and } P_{BI} = \frac{1}{T_{block}} \tag{5}$$

We now compute P_{IT}: for this state transition, the node chooses to contend in the slot rather than listen (probability p_{tx}) AND either the receiving node is in Blocked state, OR the receiving node is in Idle state AND chooses to listen (probability p_l). Therefore,

$$P_{IT} = p_{tx}\left(\frac{P_I}{P_I + P_B} p_l + \frac{P_B}{P_I + P_B}\right) \tag{6}$$

where the independence approximation is applied to infer that given the reference link's Idle state, the conditional probabilities of the incoming link being in the Blocked or Idle states are proportional to the respective steady-state probabilities. We can evaluate the other probabilities in a similar fashion.

$$P_{IU} = \frac{p_{tx} p_l P_I}{P_I + P_B}, P_{BU} = \frac{p_{tx} P_I}{P_I + P_B}, P_{IB} = \frac{p_{tx}^2 P_I}{P_I + P_B}. \tag{7}$$

We now present an iterative algorithm to compute the steady-state probabilities in Procedure 1.

Consistency demands that the steady-state probabilities of the Transmit and Unavailable states should be equal for the two-node network because of symmetry, and indeed we find that this is always the case. We now compare the analytical and simulation results for the expected link utilization for successful transmissions for each node. The total medium utilization in this case would be the sum of the transmit link utilization for the two nodes sharing the link capacity. The steady-state probabilities calculated from the Markov chain model are: P_T=P_U= 0.489, P_I= 0.015 and P_B= 0.007. QualNet [14] simulations of the protocol for this setting yields the fraction of the successful transmit and receive state slots as 0.492 each, which demonstrates a close match. The other state fractions are Blocked: 0.013 and Idle: 0.002 - the differences correspond to additional refinements embedded into the actual MDMAC QualNet protocol model.

Algorithm 1 State probabilities computation

```
1: Initialize $P_I = 1,\ P_T = P_U = P_B = 0$.
2: Use the current values of $P_s$ and (5), (6), (7) to compute
   the transition probabilities $P_{ss'}$.
3: Use the values for $P_{ss'}$ obtained in Step 2 and (4) along
   with the normalization condition $\sum_s P_s \equiv 1$ to solve for
   the state probabilities and update the values of $P_s$.
4: Return to Step 2 until convergence.
```

Modeling Arbitrary Networks: The preceding analysis can be extended to compute the state probabilities $\{P_s\}$ of a "typical" link for an arbitrary network, i.e., for nodes with N bi-directional links. Note that unlike the two-node network, a link being in Unavailable state does not automatically mean that the corresponding incoming link is active; any of the other $2N - 1$ links from/to the same node can be active. We first derive an expression for P_{IT}. For this transition, the node must choose to contend for transmission (probability p_{tx}) rather than to listen, and pick the reference link (which is in the Idle state) out of the subset of the N outgoing links that are in Idle rather than Blocked state. The probability of any outgoing link other than the reference link being in the Idle state is $P_I/(P_I + P_B)$. Thus, the probability p_c that the reference link is chosen to contend is given by:

$$p_c = p_{tx} \sum_{k=0}^{N-1} \binom{N-1}{k} \left(\frac{P_B}{P_I + P_B}\right)^m \left(\frac{P_I}{P_I + P_B}\right)^k \frac{1}{k+1},$$

where $m = N-1-k$. For the contention attempt to be successful, the corresponding receiving link on the neighboring node must be in the Idle or Blocked states, and the neighboring node must choose to listen rather than contend to transmit on one of its Idle outgoing links (if there are any Idle links, or else all its outgoing links must be in Blocked state). The corresponding probability (p_{r1}) is

$$p_{r1} = p_l(P_I + P_B(1 - \left(\tfrac{P_B}{P_I+P_B}\right)^{N-1})) + P_B \left(\tfrac{P_B}{P_I+P_B}\right)^{N-1}.$$

Furthermore, the receiving node must choose the reference link out of all of its own other $N-1$ neighboring nodes (also "typical" nodes) that also happen to be Idle and contending for this slot (probability $P_I p_c$). The probability of this event

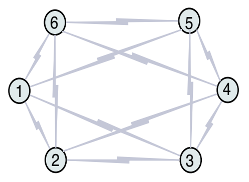


Fig. 8. An example network with 4 neighbors per node.

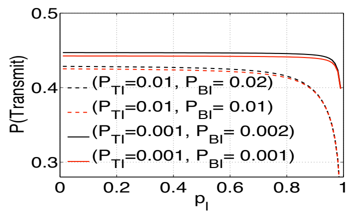


Fig. 9. Medium utilization by node transmissions.

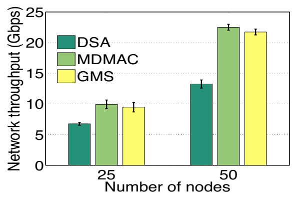


Fig. 10. Aggregate throughput.

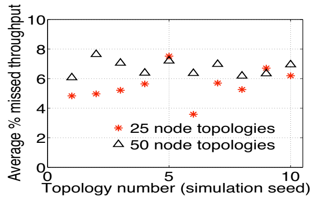


Fig. 11. Missed transmit opportunities.

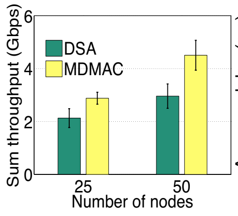


Fig. 12. Throughput.

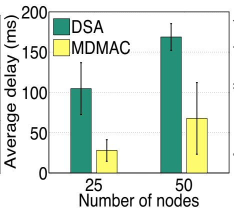


Fig. 13. Total delay.

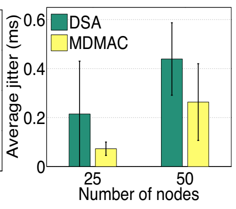


Fig. 14. Delay jitter.

p_{r2} is given by

$$p_{r2} = \sum_{k=0}^{N-1} \binom{N-1}{k} \frac{(P_I p_c)^k}{k+1} (1 - P_I p_c)^m, \tag{8}$$

where $m = N - 1 - k$. We finally have

$$P_{IT} = p_c p_{r1} p_{r2} \tag{9}$$

In case the transmission attempt is not successful, the reference outgoing link is forced into the Blocked state. We obtain P_{IB} as

$$P_{IB} = p_c(1 - p_{r1}p_{r2}) \tag{10}$$

We now define the probability (p_a) that at least one of the neighbor nodes of the reference node attempts to transmit as $p_a = 1 - (1 - P_I p_c)^N$. For P_{IU}, the reference node must either successfully transition to the Transmit state on one of the other $N-1$ outgoing links, or it must successfully receive from one of its contending neighbors. This leads to

$$P_{IU} = (p_{tx} - p_c)p_{r1}p_{r2} + p_l p_a \tag{11}$$

For P_{BU}, the transition to the Unavailable state can happen either if the reference node successfully transitions to the Transmit state on one of its other $N-1$ outgoing links, or if it successfully receives from one of its transmitting neighbors. Therefore,

$$P_{BU} = p_{tx}\left(1 - \left(\frac{P_B}{P_I + P_B}\right)^{N-1}\right) p_{r1}p_{r2} + p_a\left(p_l\left(1 - \left(\frac{P_B}{P_I + P_B}\right)^{N-1}\right) + \left(\frac{P_B}{P_I + P_B}\right)^{N-1}\right) \tag{12}$$

The other transition probabilities (P_{TI}, P_{UI}, P_{BI}) can be obtained from (5). Procedure 1 can again be used to compute the steady-state probabilities $\{P_s\}$. Note that consistency requires that $P_U = (2N-1)P_T$.

We apply our model to an example six-node network in Fig. 8 where every node has four neighbors with link-saturating flows in each direction. The steady-state medium utilization for successful transmissions from each node is obtained as 0.426. Packet-level simulation of the same scenario yields 0.43. We now investigate the effect of the following parameters on the performance of MDMAC (1) P_{TI}, P_{UI}, and P_{BI}; and (2) p_l, under saturated traffic conditions and a given network node density. We consider the probabilities P_{TI} and P_{UI} such that the average transmit/receive state lifetime T_{slot} is in the range of 100 to 1000 frames, i.e., P_{TI}, P_{UI} values of 0.01 to 0.001. In order to understand the effect of tuning the Blocked state lifetimes, for each value of P_{TI}, we consider $P_{BI} = P_{TI}$ or $P_{BI} = P_{TI}/2$. Fig. 9 shows the expected total medium utilization by a node's transmissions to all its neighbors. We find that P_{TI}= 0.001 and P_{TI}= 0.002 yield the highest medium utilization among the parameter choices that we have considered. Also, the medium utilization is relatively insensitive to the value of the listening probability p_l (in the range 0.1-0.7). We observe similar trends for medium utilization over a large range of neighbor densities, and therefore set P_{TI}=P_{UI}= 0.001 and P_{BI}= 0.002 for our evaluation. These relatively large state lifetimes result in less throughput loss due to churn, while offering enough possibility for rearrangement of schedules to ensure fairness. Although our analytical guidance for choice of P_{TI}, P_{UI} and P_{BI} is for saturated traffic, our simulations indicate that these values are effective for unsaturated multihop mesh traffic as well.

Explicit State Reset: In addition to probabilistic state reset and non-persistent contention, we devise an explicit state reset mechanism to enable a quicker response to changing traffic patterns and maintain fair bandwidth allocation to each neighbor. When the fraction of transmit or receive slots at a node exceeds a threshold, the node successively resets the state of a randomly picked slot (i.e., releases the slot to the available slot pool) among the committed slots corresponding to the neighbor that holds the highest share of the committed transmit or receive slots. Basically, a node switches to an "alert" mode when the bandwidth demand starts to approach the total link capacity, and ensures that the bandwidth allocations among competing neighbors are not grossly unfair.

We now present a sampling of performance results for MDMAC obtained via packet-level simulations over the QualNet simulator [14] with the PHY and antenna modules modified to model 60 GHz links.

Fig. 10 compares the aggregate network throughput achieved with MDMAC against directional slotted

Aloha (DSA) protocol and a Greedy Maximal Scheduling scheme [15], considering saturated network traffic for random 25 and 50 node topologies over a 500m x 500m flat terrain. The aggregate network throughput for MDMAC is much higher than DSA, and is comparable to that achieved with centralized Greedy Maximal Scheduling.

Fig. 11 presents estimates of the fraction of total transmit opportunities "missed" by MDMAC over different simulation instances, and shows that on average the TDM schedules generated by MDMAC are quite efficient: they are within 6% and 7% of the corresponding largest cardinality maximal matchings on the network graphs for the 25 and 50 node topologies, respectively.

We now look at MDMAC's performance considering asymmetric multihop mesh traffic with randomly chosen flows to and from the assigned gateway nodes to other mesh nodes. Fig. 12 shows that MDMAC attains a significantly higher aggregate throughput than DSA (35% and 52% for the 25 and 50 node topologies, respectively). A higher fractional gain over the 50 node topologies demonstrates that MDMAC is effective despite the increased contention and interference resulting from the high node density. Figs. 13 and 14 offer insight into the QoS performance in terms of the much lower end-to-end delay and delay jitter for the received packets relative to that with DSA. The end-to-end delay value is low enough to satisfy the typical Internet traffic delay requirements. The TDM-like schedules attained via MDMAC lead to low jitter.

IV. Conclusions

Our two examples show the promise of implicit coordination mechanisms in the design of network protocols: we show that phase/frequency adaptation based on existing communication patterns can lead to network-wide synchronization, and that the decentralized MDMAC protocol can yield efficiency close to that of centralized TDM. Important topics for future research include integrating such concepts into a comprehensive suite of protocols for mesh networks that include network discovery, routing and flow control. While highly directional 60 GHz mesh networks represent perhaps the most exciting applications of these ideas, we believe that the concepts presented here may also lead to performance gains for mesh networks with omnidirectional links at lower carrier frequencies. Of course, much further effort is required to validate this assertion.

Both of the mechanisms presented here are examples of *stigmergy,* where agents coordinate indirectly by modifying their environment. Stigmergy is a powerful mechanism for large-scale self-organization using local interactions, examples of which abound in nature, including flocking/swarming, termite building and food finding. The results here motivate a broader research agenda of systematically designing stigmergic mechanisms for solving problems of scale in communication network design.

Acknowledgements

This work was supported by the Institute for Collaborative Biotechnologies under grant DAAD19-03-D-0004 from the US Army Research Office, and by the National Science Foundation under grant CNS-0832154.

References

[1] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Trans. on Automatic Control*, vol. 49, no. 9, pp. 1520–1533, 2004.

[2] R. Solis, V. Borkar, and P. Kumar, "A new distributed time synchronization protocol for multihop wireless networks," in *Proc. 45th IEEE CDC, San Diego, CA, USA*, 2006.

[3] L. Schenato and G. Gamba, "A distributed consensus protocol for clock synchronization in wireless sensor network," in *Proc. IEEE CDC 2007*.

[4] P. Sommer and R. Wattenhofer, "Gradient Clock Synchronization in Wireless Sensor Networks," in *Proc. ACM/IEEE IPSN '09*, 2009.

[5] R. Mirollo and S. Strogatz, "Synchronization of pulse-coupled biological oscillators," *SIAM J. on Appl. Math.*, pp. 1645–1662, 1990.

[6] S. Singh, R. Mudumbai, and U. Madhow, "Distributed Coordination with Deaf Neighbors: Efficient Medium Access for 60 GHz Mesh Networks," in *Proc. IEEE INFOCOM 2010*, Mar. 2010.

[7] S. Singh, P. Acharya, U. Madhow, and E. Belding, "Sticky CSMA/CA: Implicit synchronization and real-time QoS in mesh networks," *Ad Hoc Netw.*, vol. 5, no. 6, pp. 744–768, 2007.

[8] R. Ramanathan, J. Redi, C. Santivanez, D. Wiggins, and S. Polit, "Ad hoc networking with directional antennas: a complete system solution," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 3, pp. 496–506, March 2005.

[9] Y.-B. Ko, V. Shankarkumar, and N. Vaidya, "Medium access control protocols using directional antennas in ad hoc networks," in *Proc. IEEE INFOCOM '00*, vol. 1, 2000, pp. 13–21.

[10] A. Nasipuri, S. Ye, J. You, and R. Hiromoto, "A MAC protocol for mobile ad hoc networks using directional antennas," *Proc. IEEE WCNC 2000*, vol. 3, pp. 1214–1219 vol.3, 2000.

[11] M. Takai, J. Martin, R. Bagrodia, and A. Ren, "Directional virtual carrier sensing for directional antennas in mobile ad hoc networks," in *Proc. MobiHoc 2002*. New York, NY, USA: ACM, 2002, pp. 183–193.

[12] T. Korakis, G. Jakllari, and L. Tassiulas, "CDR-MAC: A protocol for full exploitation of directional antennas in ad hoc wireless networks," *IEEE Trans. Mob. Comput.*, vol. 7, no. 2, pp. 145–155, Feb. 2008.

[13] R. R. Choudhury, X. Yang, R. Ramanathan, and N. H. Vaidya, "On Designing MAC Protocols for Wireless Networks Using Directional Antennas," *IEEE Trans. Mob. Comput.*, vol. 5, no. 5, pp. 477–491, 2006.

[14] QualNet, v4.1. [Online]. Available: http://www.scalable-networks.com

[15] M. Leconte, J. Ni, and R. Srikant, "Improved bounds on the throughput efficiency of greedy maximal scheduling in wireless networks," in *Proc. ACM MobiHoc '09*. New York, NY, USA: ACM, 2009, pp. 165–174.