Distributed TDMA for Privacy Sensitive Anonymous Networks

Debasmit Banerjee, Mahmoud Taghizadeh and Subir Biswas
Electrical and Computer Engineering, Michigan State University

Abstract – This paper proposes a distributed TDMA slot allocation protocol that relies on absolutely no information exchange among the participating nodes. This novel property allows the protocol to work in restricted anonymous environments such as in privacy-sensitive body area networks and various military networks in which nodes may need to cooperate in distributed TDMA but are not allowed to explicitly exchange any information such as node-IDs in order to preserve their anonymity. This paper introduces an innovative approach of time-coded packet transmissions for implicitly exchanging slot timing information. It is shown that using such implicit information, together with a notion of interrupt control packets, the nodes are able to self-allocate collision-free TDMA slots in an anonymous manner. The protocol is evaluated and its performance has been shown using extensive simulation models.

Keywords – Privacy; Secure wireless network; Distributed TDMA; Medium Access Control; Anonymous Networking

I. INTRODUCTION

A. Overview

Wireless ad-hoc and sensor networks are used in emerging applications such as body area sensing, intrusion and event monitoring, and for a hoard of military applications in privacy-sensitive environments. Also, recent advances in wireless and mobile communication technologies coupled with the recent proliferation of portable computer devices have led the development efforts for future wireless networks towards wireless and mobile ad hoc networks.

A critical component of wireless networks is the medium access control (MAC) protocol that is responsible for administering wireless channel access. Two broad categories of wireless MAC protocols are TDMA based [1–3] and CSMA contention based [4, 5]. The main advantage of TDMA over CSMA is its contention-free nature and guaranteed fairness, which are especially attractive in highly dense networks with real-time constraints on message latency. TDMA protocols ensure collision-free access without the need for contention of slots between nodes.

Traditionally, TDMA slot allocation is managed by a centralized entity or access point, which is responsible for assigning slots to nodes in the network. Such centralized approach is not favorable for dynamic and distributed networks like ad-hoc and sensor networks due to the lack of infrastructure in such networks. As a result, majority of the TDMA protocols for sensor and ad hoc networks in the literature have focused on distributed slot allocation solutions which rely on in-band [3], [6] or out-of-band [7–9] control mechanisms for slot-assignment. Such control mechanisms require network nodes to exchange their occupied slot information so that they can individually adjust their slot occupancy based on a distributed allocation algorithm.

One more aspect of the shared wireless medium in mobile ad-hoc networks (MANETs) is that it introduces opportunities for passive eavesdropping on data communications. Adversaries can easily overhear messages “flying in the air” without physically compromising a node, and infer various information from these messages. While end-to-end application-level security mechanisms can provide some level of security for the data, valuable information, such as location and relationships of the communicating entities may easily be determined from traffic and data analysis, network-level address visibility, etc. This can pose a serious issue in networks with high anonymity and privacy requirements.

B. Motivation

Anonymity and resistance to traffic analysis is an interesting and difficult problem. Anonymity requirements prevent nodes from revealing their identities and distributed TDMA slot allocation mechanisms proposed in the literature may not be applicable in such networks where identity exposure of the nodes is not a choice. Wireless networks in such environments may require packets to be fully encrypted in all protocol layers (i.e. from physical layer to all the way to the application layer [10]) of the network stack. From a MAC perspective, this implies that no header information, including MAC source and destination addresses, can be exposed. Radosavljevic et. Al. showed in [11] that a periodic traffic flow like in TDMA networks can prevent traffic analysis. The assumption made in the aforementioned work is that an eavesdropper can detect transmissions but cannot understand the encrypted messages being sent.

In this paper we propose a protocol for distributed TDMA slot allocation in multi-hop wireless networks with strict anonymity constraints. We term strict anonymity constraint as zero-exposure and propose a lightweight zero-exposure slot-scheduling MAC protocol suitable for networks requiring fast convergence characteristics. This protocol has significantly lower complexity compared to our previous work [12] that relies on an elaborate shadow packet mechanism for sharing slot occupancy information. We assume that anonymity is achieved using a layer-wise encryption technique. The network might be divided into different trust-domains and nodes which trust each other belong to the same trust-domain. We imagine a network model where multiple heterogeneous nodes belonging to different trust-domains coexist in a wireless neighborhood and cannot share any control-information among them as it might reveal their identity and localization information. Encryption mechanisms used to achieve anonymity and the trust-relationships between nodes is an application issue and is beyond the scope of this paper. The key concept here is to use an innovative approach of time-coded packet transmissions for implicitly exchanging slot timing
information without having to rely on any explicit information exchange (e.g. MAC or other IDs) among the network nodes. Using only such implicit information, the nodes are able to self-allocate collision-free TDMA slots in an anonymous and privacy-preserving manner.

C. Key Challenges

Slot allocation in TDMA protocols for multi-hop wireless networks need to satisfy the constraint that at steady state, no two one-hop or two-hop neighbors can have partially or completely overlapping transmission slots. Overlaps between slots of one-hop neighbors cause direct collisions and overlaps between slots of two-hop neighbors can cause hidden collisions. In an anonymous environment, exchanging slot occupancy information among one-hop or two-hop neighbors is not feasible since two nodes may belong to two different trust-domains which may prevent them from exchanging any explicit information including their MAC layer IDs. Therefore, the primary notable challenge for an anonymous distributed slot allocation protocol is the unavailability of any explicit means to disseminate slot occupancy information within up to two-hop wireless neighborhoods. The second challenge stems from the absence of network-wide time synchronization without which the task of cross-node slot alignment is non-trivial [13].

D. Contributions

The contributions of the paper are as follows. First, a time-coded packet transmission mechanism for implicitly exchanging slot occupancy information is developed. Next, the protocol is implemented using NS2 network simulator and its functionality is validated using extensive simulations. Finally, the protocol performance is evaluated in varying topological arrangements and network dynamics. The rest of the paper is arranged in the following manner. Section II demonstrates the protocol and the slot allocation process in an anonymous environment. Section III shows the evaluation and results of the protocol. Section IV discusses the related work and finally Section V concludes the paper with summary and future work in this area.

II. Zero-Exposure TDMA

A. Time-Coded Packet Transmission

Once a TDMA slot (of duration $\tau$) is self-selected, a node periodically sends packets in that slot once in every TDMA frame (of duration $T$). The packet a node periodically sends during its own TDMA slot is known as a regular packet. A node’s slot timing is implicitly detected by all its one-hop neighbors by simply noting the node’s regular packet transmission time. Using such implicit information, each node in the network can learn the slot timing of all its one-hop neighbors without actually knowing the identification of the neighbor nodes. In the absence of time synchronization, each node maintains its own TDMA frame, starting at the node’s own slot time. Fig. 1(a) shows how the nodes in a three-node network maintain information about the slot occupancy of the other nodes with respect to their own individual TDMA frames. We introduce the concept of a phase ($\phi$) which represents the location (i.e. timing) of a node’s slot from the perspective of a hypothetical global observer that may maintain a global frame. For example, to a global observer, the phase of node A would appear to be $\phi_A$ as shown in Fig. 1(b). Without inter-node time synchronization, the absolute phase does not have any meaning from an individual node’s (i.e. that of a local observer) standpoint. However, a node can detect its phase difference with another node. Phase difference between two nodes is the time difference between their respective slots. For example, the quantity $\delta_{AB}$ in Fig. 1(a) represents the phase difference between nodes B and A as perceived by node B.

![Fig. 1: (a) Local slot occupancy view to individual nodes (b) Slot occupancy view to a global observer](image)

B. Slot Self-Allocation

Upon joining the network, a node chooses an arbitrary time as its frame starting point and gathers the slot occupancy information about its one-hop neighbors by simply listening to their regular packet transmissions. After one frame-duration, a node receives regular packet transmissions from all nodes within its one-hop neighborhood and creates a one-hop slot occupancy list from the time of arrival of the regular packets. Then the node self-allocates the next available slot and defines it to be the new starting point of its own frame. At this point, the node starts sending regular packets in that slot for every consecutive frame.

It is necessary for a node to send a regular packet in its slot every frame to maintain the ownership of its slot. In case the traffic pattern is not periodic and a node does not have any packets to send in a frame, it can send a ‘dummy’ regular packet. This is required because the only way a node’s neighbors become aware of a node’s existence is when they receive a regular packet and they implicitly know that the particular slot is used by another node in their one-hop neighborhood. If, due to energy savings reasons, a node cannot afford to send ‘dummy’ regular packets, the node can choose to not send anything during its slot. This may essentially result in a node giving up its slot and any other node with no assigned slot is free to claim the slot. When the node has data to send again, it first checks for transmission in its previous slot and if it is free, it starts reusing that slot. However, if the slot is currently being used by another node, it randomly selects a different slot which is available.

C. Interrupt packets and Carrier-sensing

The slot self-allocation process makes sure that nodes are aware of their one-hop neighbors’ slot locations before they self-allocate a slot. However, since nodes are not aware of
their two-hop neighbors’ slots, this does not guarantee that nodes select slots non-overlapping with their two-hop neighbors, which might result in hidden collisions. To take care of such situations, we implement a novel interrupt packet mechanism which is explained in the following paragraphs.

Every node senses the channel before sending a regular packet and if the channel is found to be busy, the node defers its transmission and transmits the regular packet only after the channel becomes free. When a node selects a slot overlapping with one of its two-hop neighbors, a common neighbor of these two nodes detects this illegal slot assignment by detecting a collision between their regular packets. Collision detection is done by checking the duration of the received signal. Since the size of the regular packets is fixed and other delay components including queuing and processing delays are assumed to be constant, a received signal that lasts longer than regular packet duration can be inferred as an overlapping transmission or collision at the receiver. In this mechanism, a collision can only be detected if two nodes’ slots have slots that are partially overlapping. This is ensured by a random jitter added by the nodes before self-allocation or before selecting a slot after its original slot gets deferred. The added random jitter results in increasing the effective slot size \( \tau \) from 4ms to 4.146ms. All nodes also add a random jitter to their slot after a fixed number of frames to randomize the exact location of their slots. This functionality prevents two nodes from owning exactly overlapping slots. Fig.2 shows the random-jitter and the effective slot size.

![Fig. 2. Random jitter and effective slot-size](image)

When a node senses regular packet collisions between two of its neighbors, it resolves the collision by sending an interrupt packet. We explain the two-hop collision resolution mechanism with interrupt packets using the example in Fig. 3. In the example, nodes 1 and 3, which are two-hops away from each other, share overlapping slots. Node 2 detects a regular-regular collision from the length of the signal and marks the collided slot start time, as pointed out in Fig. 3. To resolve the collision, node 2 sends an interrupt packet immediately after the start of node 1’s slot. The interrupt packet keeps the channel busy for node 3 which causes it to defer its slot until the end of the slot occupied by node 1 and hence resolves the collision between nodes 1 and 3. Collisions among more than two nodes are resolved by multiple interrupt packets. For interrupt packets to successfully resolve collisions, two criterions must be fulfilled: (i) the slots occupied by the colliding nodes have to be partially overlapping, and (ii) the interrupt packet must keep the channel busy until the end of the first node’s slot (node 1 in the Fig. 3). The first criterion is fulfilled by the random jitter added by the nodes and the second criterion is taken care of by making the interrupt packet size same as the regular packet size.

![Fig. 3: Collision detection and resolution](image)
slots are allocated without any explicit information exchange and thus satisfy the original objective of zero-exposure. Since the algorithm does not use any information from the packets, it is ideal for use in environments where encryption is implemented in all protocol layers. In a network with multiple trust-domains, nodes belonging to different trust-domains may not be able to decipher information from each other’s packets, yet they will be able to use the algorithm for a TDMA slot allocation for channel access. The slot-allocation algorithm at node $i$ is summarized in Algorithm 1.

### Algorithm 1: Operation at node $i$

- **Listen to regular packet transmissions from one-hop neighbors**
- **Create one-hop slot occupancy list**

**Slot selection: Node $i$**

- **Self-allocates the next available slot**
- **my_slot_time = CURRENT_TIME**

**Start: Node $i$**

while(alive)]

if(CURRENT_TIME = my_slot_time) {
   while(channel = BUSY) {
      //do nothing
   }
   send(regular packet)
   my_slot_time = CURRENT_TIME - $\tau$
}

if(collision detected for x frames) {
   $i$=schedule interrupt packet at the beginning of the collided slot in next frame*/
   schedule(interrupt packet)
}

### III. EVALUATION

The proposed Distributed TDMA slot allocation algorithm has been implemented within the ns2 MAC simulation module. The baseline simulation parameters are shown in Table 1. The protocol has been evaluated for linear, fully-connected and grid topologies with network size ranging from 5 to 100 nodes. Network dynamics have been tested by joining two converged sub-networks. The protocol has also been tested for two different node deployment policies – one where all network nodes are introduced at the same time (static), and the other in which nodes are introduced incrementally once the remaining network has converged (incremental).

| TABLE I: Baseline system parameters in simulation |
|---------------------------------|-----------------|
| Propagation model              | Two-ray ground  |
| Channel bandwidth              | 2Mbps           |
| Transmission range             | 240m            |
| Network size                   | 5-60 nodes      |
| Regular & Interrupt packet size| 1024 Bytes      |
| Slot duration                  | 4.146 ms        |

### A. Functionality Validation

The functionality of the protocol has been tested by running simulations for static as well as dynamic networks. Fig. 4 and Fig. 5 shows functionality results for static and dynamic linear topologies. The y-axis represents the phase of the nodes with respect to a hypothetical global observer (see Sec. II:A), and the x-axis represents the time in terms of the number of rounds or frames. For all experiments, the frame size is set to 6 slots (25 ms) which is one slot more than the maximum number of up to two-hop neighbors for the corresponding topology. An inter-node phase difference of less than 4.146 ms, which is the slot size $\tau$, indicates that two nodes have overlapping slots. Specific events of the protocol functionality have been pointed out using alphabetical markings and the corresponding state transition diagrams have been shown alongside the results.

#### 1) Static Network

Functionality tests for static networks have been performed by introducing all nodes in the network at the same time. Fig. 4(a) shows the phase of nodes in a linear network with 7 nodes and demonstrates how hidden collisions are resolved by a common neighbor. Initially, node 4 and node 6 have overlapping slots as seen from the figure. This is detected by node 5 which sends an interrupt packet to resolve the collision. This effect is marked by $E1$ in the figure when node 6 defers its slot due to the interrupt packet sent by node 5, and selects a slot non-overlapping with node 4. $E2$ in the figure depicts how overlapping slots between nodes 5 and 7 are resolved by interrupt packet sent by node 6. The state transition diagram of nodes 4, 5, 6 and 7 are shown in Fig. 4(b). Transient states are shown with dashed lines and the round number ($r$) represents the time. The phase of a node $i$ : $\phi(i)$ represents the state of the node, and overlapping slots are shown in the figure by intersecting states. It can be seen in
the diagram that event $E1$ and $E2$ causes node 6 and node 5 to change their states, respectively.

![Fig. 5](image)

**Fig. 5:** Functionality test for: (a) Dynamic network — two linear subnets joined through node 4 (b) State transition diagrams for topology in part (a)

2) **Dynamic Network**

Fig. 5(a) demonstrates the functionality of the protocol when two isolated converged sub-networks with linear topology are joined due to the insertion of a new node. After the two subnets (each with three nodes connected in a linear fashion) have converged, node 4 enters the network at round = 2003, and joins these two subnets to form a network of seven nodes connected linearly. In the figure, $E1$ marks the slot selection and start of *regular packet* transmission for node 4. At $E2$, node 5 detects overlap between slots of node 4 and node 6 and sends an *interrupt packet* to resolve this overlap. As a result, node 6 delays its slot and selects a non-overlapping slot with node 4. Node 3 also detects an overlap between node 2 and node 4 at $E2$ and sends an *interrupt packet* to resolve this. The *interrupt packet* shifts node 2’s slot as can be seen in the figure. The new slot selected by node 6 overlaps with node 7’s slot, and, as a result, at $E3$, node 7 defers its slot and selects a new slot non-overlapping with node 6. It can be seen from the state transition diagram in Fig. 5(b) that event $E2$ causes node 6 and node 2 to change their states. Here, event $E2$ represents the transmission of *interrupt packets* from both node 3 and node 5.

The above scenarios demonstrate functional characteristics of the protocol using two specific examples. We have done extensive validation testing with linear, fully connected and grid network topologies using the two node-deployment policies – static and incremental and the protocol functionality have been validated through such testing.

**B. Convergence**

Convergence time is defined as the time it takes for all nodes in the network to get a steady slot assigned to it. *F-ratio* is defined as the ratio of the chosen frame size (i.e. number of slots) and the minimum frame size needed for a network to reach convergence. The minimum frame size or the minimum number of required slots is determined by the maximum number of nodes within up to two-hop neighborhood. For example, F-ratio = 1 corresponds to the smallest possible frame size and demonstrates the situation in which the frame size is exactly equal to the number of required slots. Note that due to slot sharing among nodes farther than two-hops, the minimum frame size is usually much lesser than the actual number of nodes in the network. Fig. 6 and Fig. 7 show the convergence results for the two different node deployment policies and different F-ratio values.

1) **Static Node Deployment**

This subsection presents the time it requires for a network to converge when all nodes are introduced in the network at the same time (i.e. static deployment). Fig. 6(a) shows the convergence time of a static linear network with network size varying from 5 to 55 nodes. The x-axis represents the network size and the y-axis shows the convergence time in terms of the number of frames. Results have been shown for different F-ratio (FR) values between 1 and 2. It is seen that when the F-ratio is 1, the convergence time is maximum. Also, with an increase in the network size, the network takes more time to converge. This is because when the F-ratio is 1, nodes are very tightly spaced in a frame and change in a single node’s slot location may cause other nodes to defer their slots. Also, with larger number of nodes in the network, this effect is propagated throughout the network as can be seen from the graph. With increase in F-ratio, the convergence time decreases even as the number of nodes in the network increases. This is because there is enough space in the frame to accommodate all nodes in the network. Due to the larger frame size, a shift in one node’s slot location does not necessary cause other nodes to defer their slots. For F-ratio ≥ 1.45, the disparity in convergence time for different network sizes is much lesser when compared to lower F-
ratios. This is due to the fact that for higher F-ratios, nodes have sufficient space in the frame to converge and hence the reduction in convergence time when F-ratio is further increased is not significant.

Fig. 6(b) shows the convergence time of a grid network with static node deployment. The observations that can be made for this network are similar to linear network results. It can be seen, that with an increase in F-ratio, the convergence time decreases as nodes quickly self-allocate slots from a larger frame space. Although the results have a pattern similar to the linear network results, the convergence times are almost double in the case for grid topology. This is due to the topological difference between the two networks. A grid topology is more complex in nature with the presence of loops which result in circular dependencies among the nodes which results in the delays in convergence. However, for high F-ratios (e.g. F-ratio = 1.9), the convergence time is almost same as linear networks due to the ample space in the frame which reduces the effect of slot changes among nodes in the network.

For fully-connected networks shown in Fig. 6(c), the pattern of results is similar to linear topology. However, the time taken to converge is higher than linear networks and lower than grid network. This is because of the following two reasons. Firstly, a change in one node’s slot affects all nodes in this topology, which is unlike linear topologies, where the effect is limited up to two hop neighbors. However, this effect is diminished for higher F-ratios as can be seen from the graph. Secondly, the circular allocation dependency of grid network is not encountered in a fully-connected topology hence keeping the convergence time lower.

### 2) Incremental Node Deployment

This subsection presents the time it requires for a network to converge when nodes are added one at a time. A new node is added only after the network is converged after the previous node addition. Although experiments have been done for linear, fully-connected and grid network topologies, only results for grid network is shown as the other two networks mostly have instant convergence after a node is added.

Fig. 7 and Fig. 8 show the convergence time for incremental node deployment for grid networks with varying sizes. The x-axis represents the node number being added and the y-axis shows the time required (in terms of number of frames) for the entire network to converge. This node addition results in incremental formation of the topology as the nodes are added individually. This means that as nodes are added to form a 10x10 grid network, the network topology goes from a 4x4 grid to 6x6 grid and so on, finally forming a 10x10 grid network. This is reflected in the results where we see that the convergence time for nodes in a larger network (e.g. 10x10) have similar initial pattern as that of smaller networks (e.g. 6x6, 8x8).

![Fig.7: Incremental node deployment convergence characteristics for: (a) 4x4 grid (b) 6x6 grid](image)

![Fig.8: Incremental node deployment convergence characteristics for: (a) 8x8 grid (b) 10x10](image)

Another observation from the results is the difference in convergence time for different nodes in the network. Few nodes have instant (zero) convergence, while others have non-zero convergence. This was influenced by the initial neighborhood of a node when it joined the network. For our deployment policy in a grid topology, a node can be initially joined to one neighbor or two neighbors. In situations where a node joins two neighbors, it has more information about its neighborhood and hence selects the next available slot which is collision free. However, in cases where a node joins one neighbor, it selects a slot based on information from only one node and ends up selecting a slot overlapping with a two-hop neighbor, which is then resolved, and hence results in non-zero convergence. Another observation that can be made from the results is that for higher F-ratios, the network tends to converge faster than instances with lower F-ratios. This is due to the fact that the probability of a new node finding a free slot is higher when the frame size is larger.

### IV. RELATED WORK

Several TDMA protocols have been proposed in the literature. Classical centralized TDMA mechanisms face the roadblock of non-scalability and inefficiency for large ad hoc and infrastructure-less sensor networks. To address the
scalability issue, Funneling-MAC [14] uses TDMA in high-traffic parts of a network, and CSMA in the low-traffic parts. However, this protocol relies on a central sink to coordinate the TDMA slot allocation. Any centralized protocol suffers from single point of failure weakness. TreeMAC [3] and TDMA-W [7] address this issue by using a non-centralized approach.

However they are dependent on time-synchronization which is a non-trivial task for distributed ad-hoc networks. Z-MAC [15] works in the presence of loose time-synchronization, but it still needs global time-synchronization at startup. The third category of TDMA mechanisms are fully distributed and work in the absence of time synchronization. ISOMAC [6] and DRAND [8] are examples of such distributed TDMA protocols which work independent of network-wide time synchronization and are ideal for completely distributed ad hoc networks.

All the above approaches rely on control packets to forward two-hop neighbors’ slot allocation information in the network. Control packets usually identify the nodes sending the packet and can also contain slot allocation information of neighborhood nodes along with their identity information, thus violating anonymity requirements. Therefore they are not suitable for highly privacy-sensitive.

There has also been some prior work investigating the problem of coexistence between wireless network nodes sharing the same channel, but belonging to different networks. [16] has focused on semantic interference caused by mutually intelligible frames between heterogeneous network nodes belonging to independently co-located wireless networks. This paper discussed different possibilities of isolating co-located networks, like physical layer isolation, MAC layer isolation or cryptographic isolation. In another work [17], researchers discuss the coexistence of a body area network (BAN) and an external Low Data Rate BAN and analyze the performance improvement of the reference BAN by introducing an optimized time-hopping code assignment. [18] studies the effects of beacon collisions, due to co-located IEEE 802.15.4 networks, on beacon-based synchronization, causing performance degradation. Similarly, [19] analyzes the coexistence issues of 802.15.4 networks. Although there is a necessity of a TDMA-like scalable MAC protocols for BANs [20], [21], none of these works try to approach the solution from the perspective of the development a distributed TDMA protocol which can support network coexistence.

V. SUMMARY AND ONGOING WORK

We have presented a novel distributed TDMA slot-scheduling protocol for anonymous networks in which slot-occupancy information is disseminated using time-coded packet transmissions, and that is without using any explicit information in the packet. The protocol does not rely on time-synchronization and it works in wireless ad-hoc networks where nodes belong to different privacy domains where the packets are fully encrypted. The functionality and evaluation of the protocol has been demonstrated using NS2 simulations.

Ongoing work on this topic includes development of a prototype to implement this protocol on a hardware platform.

VI. REFERENCES