CH-MAC: A Cluster-based, Hybrid TDMA MAC Protocol over Wireless Ad-hoc Networks

Hoyong Choi†, Jihwan Bang‡, Namjo Ahn†, Jinhwan Jung†, Jungwook Choi*, Soobum Park*, and Yung Yi†
† School of Electrical Engineering, KAIST, South Korea, (email: {hychoi, njahn, jhjung}@lanada.kaist.ac.kr, yiyung@kaist.edu)
‡ Search Solutions Inc., South Korea, (email: jihwan.bang@navercorp.com)
* LIG Nex1, South Korea, (email: {jungwook.choi, sbpark93}@lignex1.com)

***Abstract*—In this paper, we propose a distributed TDMA MAC protocol over wireless ad-hoc networks, called CH-MAC. The key design component of CH-MAC is the notion of clusters, where a cluster head maintains the information about how slots are allocated over the one-hop neighborhood. This cluster-based operation enables CH-MAC to opportunistically operate in a hybrid fashion of being both reactive and proactive. In other words, once a node intends to be allocated some slots, its cluster head provides the information about the interfering slots over near-by nodes in a proactive way, and far-away interference such as the one by hidden nodes is handled in a reactive manner. This hybrid operation based on clusters offers a wider design space of appropriately trading off the slot-allocation response time and the number of control overheads. We validate the efficiency of CH-MAC over a variety of scenarios in terms of the number of nodes and the network topology.**

I. Introduction

We consider a MAC protocol that is designed to guarantee a certain level of QoSes (e.g., reliability and delay) in mobile adhoc networks, which is frequently considered in military applications, e.g, battlefields. CSMA, TDMA, or their hybrids are the major types of MAC protocols, and there exists an extensive array of MAC protocols based on those design directions. We refer the readers to [1] and [2] for a nice survey of representative protocols.

We aim at proposing a dynamic, distributed TDMA protocol over MANETs, where by dynamic we mean that we support the dynamic changes of slot schedules in response to time-varying loads, and by distributed we mean that there exists no centralized coordinator which is responsible for allocating time slots to nodes. Supporting time-varying loads in scheduling time slots is challenging, because it essentially requires to reschedule slots for multiple nodes. Obviously, we are not the only one which develops such a type of TDMA protocol. There exist many criteria under which existing ad-hoc TDMA protocols are classified. In this paper, we broadly classify them into two categories: proactive and reactive. In proactive protocols, nodes advertise their status information (e.g., slot schedule information) to neighboring nodes, where whenever a new request for slots arrives, it is quickly served thus a small delay for new slot scheduling. However, such proactive protocols suffer from significant control message overhead, which is often too problematic to be used in a network with scarce resource, as experienced in many military networks. On the other hand, reactive protocols utilize network resource more economically, whereas they experience a long delay of slot allocation.

In this paper, we take a hybrid approach by operating a dynamic TDMA protocol in a partially proactive and reactive manner, with the hope of obtaining the advantages from those two design choices. The key design component of CH-MAC is the notion of clusters, where a cluster head maintains the information about how slots are allocated over its one-hop neighborhood. This cluster-based operation enables CH-MAC to opportunistically operate in a hybrid fashion of being both reactive and proactive. In other words, once a node intends to request some slots, its cluster head provides the information about the interfering slots over nearby nodes in a proactive way, while far-away interference such as the one by hidden nodes is handled in a reactive manner. This hybrid operation based on clusters offers a wider design space of appropriately trading off the slot-allocation response time and the amount of control overheads.

We evaluate CH-MAC using simulations implemented on NS-3 simulator under various traffic loads and topologies which are random and dumbbell networks designed to have multiple hidden terminal problems. Our results demonstrate that CH-MAC takes advantages from proactive and reactive protocols as a hybrid approach of them. More precisely, CH-MAC has a better delay performance than the reactive protocol which is significant at high traffic loads. Also, CH-MAC has up to 5 times lower control overheads compared to that of the proactive protocol. In terms of a throughput performance, CH-MAC outperforms the proactive and reactive protocols measured by the session service time. Finally, we demonstrate the robustness of CH-MAC under mobility scenarios where the performance of CH-MAC recovers within 2 frames.

II. Related Work

There exist an extensive array of research papers on TDMA in literature, where the key focus has been on (i) theoretically studying the hardness of slot scheduling, (ii) developing algorithmic solutions of scheduling slots in a distributed manner, and (iii) proposing practical MAC protocols over wireless multi-hop networks. We refer the readers to these nice surveys for the exhaustive list of those related works [1] and [2].

We present the related work considering our contribution that we support load-adaptation in a hybrid manner, i.e., partially proactive and partially reactive. TDMA protocols

supporting time-varying loads include [3]–[7]. In the load-adaptive TDMA protocols, data slots are scheduled depending each node's traffic demand. MH-DESYNC [3] is a distributed TDMA scheduling protocol that operates in a purely proactive manner, where each node periodically exchanges slot usage information with all of its interfering nodes. To allocate a required amount of slots requested by a node, say v, v chooses contention-free slots based on the pre-advertised slot usage information. TMRR [4] proposes a flow-based slot scheduling which schedules slots sequentially over the entire path of the newly created flow with the goal of minimizing end-to-end delay. In TMRR, the slot allocation information, also called resource map, is broadcasted in the beacon slots at every frame. E-TDMA [5] proposes a contention-based slot allocation, i.e,. slots are assigned in a reactive manner, which can adapts to topology and traffic demand changes by incrementally updating schedules. In [6], each node shares only local information to find feasible schedules that guarantee the required number of slots, where depending on how the local information becomes available, it can operate in a proactive or reactive way. The authors in [7] propose a distributed TDMA protocol which controls the size of frame depending on demands. Our work differs from the above in the sense that CH-MAC explicitly places proactive and reactive features in the slot scheduling procedure that widens our design space to have flexible trade-off between latency and control overhead in scheduling slots.

Other related work on the protocols that try to assign at least one time-slot to each node/link includes [8]–[12]. The authors in [9] uses a five-phase reservation protocol (FPRP) which reserves slots based on contentions, so that the slots are allocated to some nodes that win the contention. DRAND [10] is also a contention-based slot scheduling protocol which exchanges scheduling information between neighbors and allocates contention-free slots. In [12], the authors propose a sequential scheduling algorithm among nodes, implying that each node is scheduled sequentially a slot which is not used by its neighbors. TDMA has also been an active medium access rule in wireless sensor networks, where traffic load is highly periodic and/or low. Thus, most of protocols are designed towards energy efficiency and simplicity, finding contention-free schedule that guarantees one slot to each node is a major focus [13]–[18].

III. Design of CH-MAC

In this section, we present the design of CH-MAC (Cluster-based Hybrid TDMA MAC), by describing the key design direction, followed by the design details.

A. Key Design Direction

- ***Session-based slot maintenance.*** Our protocol allocates collision-free schedules and releases them when a new session is initiated and terminated, respectively. This means that when a node opens a new session, it explicitly sends a slot allocation request to the network, where the requested number of slots is determined by the opening session to be serviced. We adopt this session-based allocation to consider sessions requiring strict QoS guarantee, which happens in tactical military applications. This session-based slot maintenance enables dynamic change of slot allocation/deallocation, since the lifetime of a session is typically in the order of hundred frames, thus the gain from slot allocation change offsets the control message overhead for those changes.
- ***Cluster-based topology maintenance.*** In CH-MAC, a cluster is formed as the collection of a cluster head and member nodes which are connected to the cluster head by one hop (thus, each cluster head is connected over two hops at maximum). Relying on these clusters, all the information about slot allocation inside a cluster is managed by its cluster head. The current slot schedules are informed to all nodes in its cluster by the cluster head and those nodes request slot allocation to the cluster head based on the received slot information. Thus, whenever a new request for slots is generated at some node v, using the information from v's cluster head, v is immediately able to know which slots are collision-free with the nodes in v's cluster (intra-cluster phase). The final decision for the allocated slots is made for assuring that the candidate slots are collision-free over two hops (inter-cluster phase).
- ***Hybrid operation: Proactive and reactive.*** As presented in the previous paragraph, CH-MAC tries to find a conflict-free schedule in two phases: intra-cluster and inter-cluster, each of which operates proactively and reactively, respectively. This hybrid operation is highly flexible so that the degree of proactiveness and reactiveness can be adjusted by the designer in a situation-dependent manner (i.e., the number of nodes, the amount of usual traffic load, etc). This significantly widens our design space, which opens a way of positively trading off those two rationales, i.e., latency and control overhead in slot allocation.

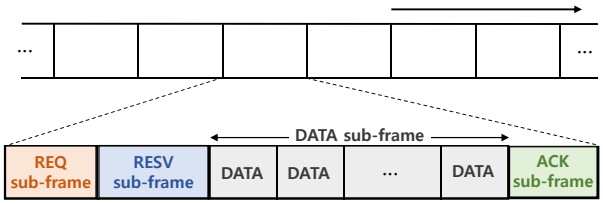


Fig. 1: CH-MAC frame structure with sub-frames.

B. Frame Structure

In CH-MAC, time is divided into multiple frames. Each frame is repeated over time and it is composed of four sub-frames: **REQ, RESV, DATA**, and **ACK**. Each sub-frame has its own pre-specified number of fixed-size time slots (simply slots), where the length of a time slot is chosen such that it covers one MAC layer packet transmission. The lengths (i.e., the number of slots) of an entire frame and of each sub-frame are the parameters of CH-MAC. The role of each sub-frame is elaborated in what follows.

Algorithm 1: When the node v requests m data slots

Input: v's 1-hop neighbors $N(v)$ and its cluster head $\texttt{CH}(v)$

Output: Collision-free m slots

S1. Usage Information Update
until v receives UI-ACK from $N(v)$ **do**
Choose a slot s_{req} randomly from **REQ**
Transmit UI-REQ at s_{req}
Receive UI-ACK from $N(v)$ in **ACK**
if m data slots are able to be scheduled **then** Go to **S2**.
else Wait until m slots become available.

S2. Slot Allocation Request
Choose a slot s_{req} randomly from **REQ**
Transmit SL-REQ requesting m data slots at s_{req}
Receive NOTI in **RESV** from $\texttt{CH}(v)$
if m data slots $[s_{\text{ac}}(i)]_{i=1}^{m}$ are allocated for v **then** Go to **S3.**
else Go to **S1.**

S3. Allocation Double Check
Transmit m TEST messages at $[s_{\text{ac}}(i)]_{i=1}^{m}$
Receive UI-ACK from all of $N(v)$ in **ACK**
if v receives UI-ACK from $N(v)$
and $[s_{\text{ac}}(i)]_{i=1}^{m}$ are marked in all UI-ACKs **then**
Data transmission at $[s_{\text{ac}}(i)]_{i=1}^{m}$
else if $\{v$ receives UI-ACK from some of $N(v)$
and $[s_{\text{ac}}(i)]_{i=1}^{m}$ are not marked in at least one of UI-ACKs$\}$
or v does not receive UI-ACK from any of $N(v)$ **then**
Choose a slot s_{req} randomly from **REQ**
Transmit SL-RLS at s_{req}
Go to **S1.**

- **REQ:** REQ sub-frame is used for two purposes: (i) *Usage Information Request* (UI-REQ) for requesting the current slot usage information from one-hop neighbors and (ii) *Slot Request* (SL-REQ) for requesting slot allocation to the cluster head. This sub-frame is used only when a node intends to allocate new slots in the network or release a node's allocated slots in the network.
- **RESV:** RESV sub-frame is used by the cluster head to proactively broadcast the slot allocation information to all member nodes of its cluster.
- **DATA:** As the name implies, DATA sub-frame is used to perform actual data transmissions, where each data transmission is guaranteed to be collision-free.
- **ACK:** ACK sub-frame is used to reply to the UI-REQ sent by the REQ sub-frame. When a node has been requested by UI-REQ, the node collects slot usage status from its one-hop neighbors in DATA sub-frame, and then sends *Usage Information ACK* (UI-ACK) that includes the collected slot usage status of the one-hop neighbors. Thus, ACK sub-frame is placed right after DATA sub-frame, since each node should collect the slot usage status in DATA sub-frame to send UI-ACK.

For presentational simplicity, we use **REQ**, **RESV**, **DATA**, and **ACK** to mean those four sub-frames throughout this paper.

C. Protocol Description: When New Slots are Requested

In this section, we now describe CH-MAC with focus on how slots are allocated in case new slots are requested as in Algorithm 1, which is the most complex part, and briefly explain CH-MAC for other cases in the next subsection.

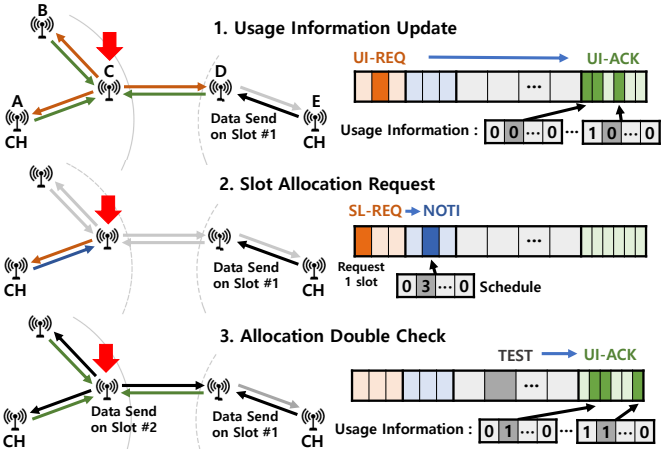


Fig. 2: An example of successful slot allocation procedure. Firstly, C sends UI-REQ and receives UI-ACK with the usage information of neighbors. In the second figure, C requests a slot and is allocated to the slot #2. Lastly, C transmits a TEST message in the slot #2 and receives UI-ACKs; thus C successes to get the collision-free slot.

Overview. Consider that node v opens a new session and requires m slots for data transmission. (i) in the Usage Information Update phase, each node obtains the slot usage information on which slots have already been allocated by the member nodes in v's cluster, (ii) in the Slot Allocation Request phase, using the slot usage information, v requests m slots from its cluster head $\texttt{CH}(v)$, which temporarily allocates those slots if available, and (iii) in the Allocation Double Check phase, those temporarily allocated slots are checked further for the possible collisions by hidden terminals. If the slots passes this check, data transmission finally occurs over the m scheduled slots. We describe the case where the number of available slots is less than m in Section III-D. We next elaborate each step, formally described in Algorithm 1, which we exemplify using the example scenario in Fig. 2.

S1. Usage Information Update. Once v requires m slots, v starts the slot allocation protocol in the Usage Information Update phase. In **S1**, v transmits UI-REQ at a slot randomly chosen from **REQ**. When v's 1-hop neighbors $N(v)$ receive UI-REQ, they reply UI-ACK to UI-REQ. UI-ACK includes the slot usage information of a transmitter of UI-ACK. Since every node always listens to **DATA** to check whether each data slot is used by its 1-hop neighbors or not, every node makes a list of the data slot usage of its 1-hop neighbors, which we call the slot usage information. Thus, whenever v collects UI-ACK from all of its 1-hop neighbors, v knows which data slots are currently scheduled to its 1-hop or 2-hop neighbors. If there exists m empty data slots which are not used by the 1-hop nor 2-hop neighbors, v goes to **S2**, otherwise v waits until m slots become available and the latency requirement of the session can be satisfied. If the latency requirement cannot be fulfilled due to the long waiting, v fails to allocate slots and waits for a new session.

Example: Fig. 2 shows a situation where C has a session to open, requesting a slot. C selects the second slot of **REQ** and transmits UI-REQ in that slot to its 1-hop neighbors A, B, and D. Those neighbors receive UI-REQ and then send UI-ACK which contains their slot usage information. Since the second data slot is not used by the 1-hop neighbors, the second column of the Usage Information from A, B, and D is 0.

S2. Slot Allocation Req. In **S2**, after v obtains the slot usage information, v transmits SL-REQ which includes the obtained slot usage information and the number of required slots to $\texttt{CH}(v)$ in **REQ**. Then, $\texttt{CH}(v)$ temporally allocates the requested slots to v, which is announced by NOTI in **RECV**. $\texttt{CH}(v)$ uses NOTI to announce the slot allocation information to the member nodes in its cluster. If the requested slots are allocated in NOTI for v, it goes to **S3**, otherwise v should return to **S1** to obtain the latest slot usage information again.

Example: C selects the first slot of **REQ** and transmits SL-REQ in that slot to its cluster head A so as to request 1 data slot. When A receives SL-REQ, A decides to schedule the second data slot to C. This is represented by Schedule which includes the ID (i.e., 3) of C. Then, A transmits NOTI which contains the Schedule in the second slot of **RESV**. As a result, C is able to be allocated to the second data slot in **DATA**.

S3. Allocation Double Check and Data TX. Once v moves to **S3**, it denotes that the temporally allocated slots for v are not used by any of its 1-hop or 2-hop neighbors until the last frame. In **S3**, for the double check v transmits data packets with a TEST message to the allocated slots. Similar to UI-REQ in **S1**, $N(v)$ transmits UI-ACK as a response to the TEST message. When v collects all UI-ACKs from $N(v)$ and the allocated slots are correctly marked in all UI-ACKs, which means that all 1-hop neighbors receive the TEST message without collisions, v starts to perform data transmission at the scheduled data slots. However, those allocated slots can be scheduled at the same time to another node from a different clusters (e.g., hidden terminal problem, see the following subsection). In this case, it is possible that at least one node in $N(v)$ cannot receive the TEST message at the allocated slots due to collisions. Thus, if v receives UI-ACK from $N(v)$, where the allocated slots are not marked correctly or does not receive UI-ACK from any of $N(v)$, v should request to release the slots to $\texttt{CH}(v)$ using SL-RLS (Slot Release) and go to **S1** so as to start the scheduling protocol again.

Example: Following the schedule received at NOTI, C sends a TEST message to the second slot in **DATA** to its neighbors. When the neighbors A, B, and D receive the TEST message, they send UI-ACK with the updated usage information where the second slot is marked. As shown in Fig. 2, all second columns in the usage information are updated to 1. Thus, the collision-free slot is successfully allocated to C.

Rationale of Allocation Double Check. Up to **S2** Slot Allocation Request phase, if a node is temporally scheduled to m data slots, it denotes that those m data slots are not used until the last frame. However, there still exist some cases where collisions occur in the m slots; hence we say

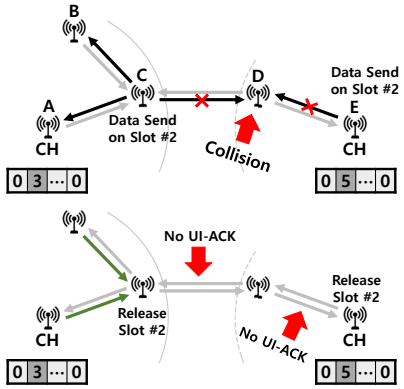


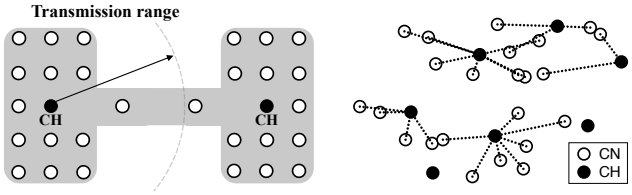
Fig. 3: Example of failure slot allocation. After Slot Allocation Request phase, C and E are allocated to the same slot #2. They transmit TEST messages in the slot #2 and they are collided in that slot. Then, C and E cannot get UI-ACK from D because D did not receive the TEST message due to the collision. Thus, C and E fail to reserve the slot.

the slots are temporally scheduled. This is because, when the node is scheduled, some other nodes in different clusters can be scheduled coincidentally to some of the m slots; thereby those m slots may not collision-free and CH-MAC introduces Allocation Double Check. We show an example that illustrates one of the collision cases in Fig. 3. Let us assume that while C is allocated to the slot #2 as the example in Fig. 2, E is also coincidentally assigned to the same slot. Then, in **S3** Allocation Double Check phase, D cannot receive any TEST message in the slot #2 due to the collision which occurs by concurrent transmissions from C and E. Thus, D will not transmit UI-ACK. Since both C and E cannot receive UI-ACK from D, they can detect the collision at the scheduled slot. By doing so, the collision on the scheduled slot is detected, so that both C and E should release the slot #2 and go to **S1** to request a different slot again.

D. Protocol Description: Other Cases

(a) When v is a cluster head. v as a cluster head can also request m slots to transmit data packets. In general, the protocol of v is almost the same as the protocol explained earlier. More precisely, in **S1**, v performs the same as the above protocol. In **S2**, since v is a cluster head, it allocates m data slots to itself rather than requests it to others and transmits NOTI in **RECV** to notify the schedule to nodes in its cluster. Lastly, the process in **S3** is also the same as the above protocol except when at least one of UI-ACKs does not correctly mark the allocated m data slots. Then, v releases the m data slots by itself and goes to $S1$ to retry.

(b) When v does not request a slot. Whenever v does not open a new session, thus does not request a new slot, it always listens to **DATA** and maintains the slot usage information of v's 1-hop neighbors, which denotes which data slot is used by one of the neighbors. While maintaining the slot usage information, when v receives UI-REQ in **REQ** or a data packet with a TEST message in **DATA**, v transmits UI-ACK that includes the slot usage information in **ACK** as a response to UI-REQ or the data packet with the TEST message.



(a) Dumbbell topology with 30 nodes (b) Random topology with 30 nodes

Fig. 4: Simulation topologies: CH (Cluster Head), CN (Cluster member Nodes)

(c) When the number of available slots is smaller than m. If v requests $\texttt{CH}(v)$ to allocate m slots but only fewer than m slots are available, then the only available slots are allocated first to v. In this case, v goes to **S3** to double check the allocated slots and concurrently goes to **S1** to wait until the remaining slots become available.

IV. Simulation Results

A. Setup

Topology and sessions. For our performance evaluation, we use NS-3 which is an open source network simulator, and implement CH-MAC. We have tested many types of topologies, but due to space limitation, we only show the results for dumbbell and random topologies with 30 nodes, as depicted in Fig. 4(a). Note that dumbbell topologies generate highly challenging scenarios due to a large degree of hidden terminals. We generate dynamic arrivals of link-level sessions to each node through Poisson process, whose characteristic is assumed to be homogeneous across links in terms of arrival intensity and session workload. We assume that each session has 100 packets to transmit. The inter-arrival time of two consecutive sessions follows an exponential distribution, whose average ranges from 0.33 to 5 secs (corresponding to session arrival rates from 0.2 to 3 sessions/sec). Since we use a constant number of packets per session in all plots, we just call session arrival rate as the *traffic load* of the system. We have run each simulation scenario 100 times, each with different random seed. Simulation duration is set to be 100 seconds which is long enough to see the stable performance.

Tested algorithms and other parameters. As mentioned earlier, CH-MAC operates in a hybrid manner of mixing reactive and proactive operations for distributed slot allocation, with the goal of trading off latency and control overhead. Thus, as compared protocols, we implement two baselines: `Proactive` and `Reactive`, purely proactive and reactive ones, just by changing the parameters of CH-MAC. Although we do not compare CH-MAC with other protocols (e.g., MH-DESYNC [3] or DRAND [10]) mentioned in Section II, we believe that these baselines show the performance of two extreme protocol behaviors, which enables us to purely focus on how beneficial a mixture of proactive and reactive control in slot allocation is, for fair comparison. Considering the different message sizes in different sub-frames, we set the slot lengths in the sub-frames of **REQ, RESV, DATA**, and **ACK** as 2ms, 2ms, 4ms, and 0.5ms, respectively. `Proactive` can be made by removing **REQ** and **ACK**, whereas we can emulate `Reative` by removing **RESV**. Other slot-related parameters are summarized in the Table I.

TABLE I: # of slots of sub-frame and frame length

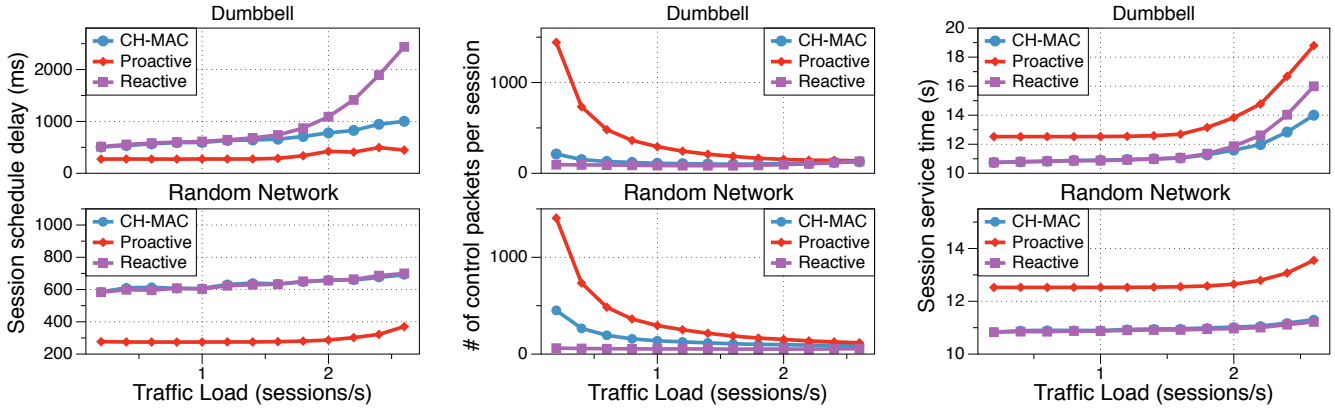
	# of slots in a sub-frame				Frame length
	REQ	RESV	DATA	ACK	Total
CH-MAC	6	6	16	30	103ms
`Proactive`	0	30	16	0	124ms
`Reactive`	12	0	16	30	103ms

B. Results

Delay for slot scheduling. Fig. 5(a) shows the average delay, measured from traffic generation to slot scheduling. Slot schedule delay consists of: (a) time for information gathering and (b) time for slot collision check. Since `Proactive` exchanges the control information among nodes without slot allocation requests, there is no information gathering delay, and we have small delay for slot collision checks. CH-MAC and `Reactive` exchange the control information whenever new sessions arrive, resulting in longer delay of information gathering compared to `Proactive`. Since CH-MAC utilizes a cluster structure to reduce the slot collision inside the cluster, delay for slot collision check in CH-MAC is much shorter than `Reactive`, highlighted when the traffic load is high, as shown in Fig. 5(a). We observe that the delays of CH-MAC and `Reactive` for the random topology does not show much difference. As shown in Fig. 4(b), the average size of a cluster and the density of the random topology are relatively smaller than those of the dumbbell topology, thus collisions resolved by the cluster heads in CH-MAC are much less frequent.

Control overhead and service time. Next, we examine the number of control packets per session, which shows how efficiently the protocol exchanges the information through control messages. As shown in Fig. 5(b), the control overhead of CH-MAC ranges between `Proactive` and `Reactive`, where only cluster heads send the control packets in a proactive manner, while all other nodes do in a reactive manner. For the efficiency metric, we measure the average time duration for which a session stays in the system. This quantifies how fast and efficiently each protocol utilizes data slots in each frame. `Proactive` uses more control slots in each frame to obtain low delay, while the portion of data slots in each frame decreases as a trade-off. `Reactive` efficiently use the limited number of control slots to reduce overheads, while delay gets longer. CH-MAC chooses a nice intermediate point between those two, wherein CH-MAC requires a small number of control slots to reduce the control delay, while incurring small control overheads, as seen in Fig. 5(c).

Robustness to mobility. A MAC protocol for MANETs must work properly when network topology changes. We test the robustness of CH-MAC under a topology change at 60 seconds with a dumbbell topology with 30 nodes. We set two kinds of mobilities: (a) low mobility where at most two of a node's neighbors change and (b) high mobility where at least half of a node's neighbors change. We assume that there exists an external mechanism that can detect the topology changes, notified to CH-MAC within one second of delay. We measured



(a) Slot schedule delay (b) Control packets transmitted per session (c) Session service time

Fig. 5: Simulation results of CH-MAC, proactive and reactive control protocols with 0 to 3 sessions generated per second.

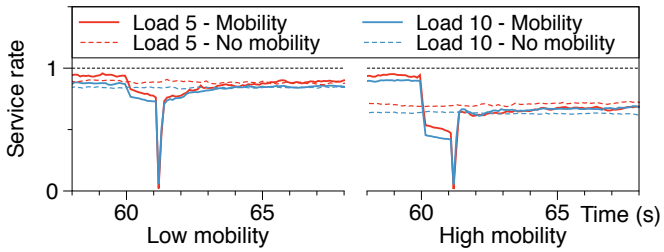


Fig. 6: Session service rates. Scenario with topology change is drawn with solid lines. Scenario which starts from the changed topology is drawn with dashed lines, to compare with original performance without mobility changes.

the service rate of sessions, which is a portion of sessions whose packet is currently being transmitted. As summarized in Fig. 6, the service rate drops at 60 seconds, since some destination sessions already unreachable. The service rate recovers when CH-MAC is notified of the topology change, and then it recovers quickly to the original value (dashed lines in Fig. 6) within just two frames.

V. Conclusion

In this paper, we proposed a new TDMA MAC protocol, CH-MAC, over wireless ad-hoc networks, that schedules slots in a distributed manner. Considering a fundamental tradeoff between scheduling delay and control overhead, CH-MAC proposes a hybrid approach of proactive and reactive operations for finding a conflict-free schedule. To this end, we exploit a cluster-based architecture, so that intra-cluster phase takes the proactive selection of candidate safe slots, and inter-cluster phase is responsible for clearing out the collision from hidden terminals in a reactive manner. Despite the extensive array of researches on TDMA protocols over wireless ad-hoc networks, we believe that this hybrid type of approach is of broad interest to a flexible MAC design depending on diverse traffic conditions and QoS. requirements.

Acknowledgment

This work has been supported by the Small-scale Mobile Ad-hoc Network with Bio-networking Technology project of Agency for Defense Development (UC170004ED).

References

[1] A. Sgora, D. J. Vergados, and D. D. Vergados, "A survey of TDMA scheduling schemes in wireless multihop networks," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, p. 53, 2015.

[2] T. Kaur and D. Kumar, "TDMA-based MAC protocols for wireless sensor networks: A survey and comparative analysis," in *Proc. of WECON*, Oct 2016.

[3] Y.-J. Kim, H.-H. Choi, and J.-R. Lee, "A bioinspired fair resource-allocation algorithm for TDMA-based distributed sensor networks for IoT," *International Journal of Distributed Sensor Networks*, vol. 12, no. 4, p. 7296359, 2016.

[4] J.-R. Cha, K.-C. Go, J.-H. Kim, and W.-C. Park, "TDMA-based multi-hop resource reservation protocol for real-time applications in tactical mobile adhoc network," in *Proc. of IEEE MILCOM*, 2010.

[5] C. Zhu and M. S. Corson, "An Evolutionary-TDMA Scheduling Protocol (E-TDMA) for Mobile Ad Hoc Networks," in *Proc. of ATIRP*, 2000.

[6] P. Djukic and S. Valaee, "Distributed link scheduling for TDMA mesh networks," in *Proc. of IEEE ICC*, 2007.

[7] A. Sayadi, B. Wehbi, and A. Laouiti, "One shot slot TDMA-based reservation MAC protocol for wireless ad hoc networks," in *Proc. of IEEE VTC*, 2011.

[8] A. Kanzaki, T. Uemukai, T. Hara, and S. Nishio, "Dynamic TDMA slot assignment in ad hoc networks," in *Proc. of IEEE AINA*, 2003.

[9] C. Zhu and M. S. Corson, "A five-phase reservation protocol (FPRP) for mobile ad hoc networks," in *Proc. of IEEE INFOCOM*, 1998.

[10] I. Rhee, A. Warrier, J. Min, and L. Xu, "DRAND: Distributed randomized TDMA scheduling for wireless ad hoc networks," *IEEE Transactions on Mobile Computing*, vol. 8, no. 10, pp. 1384–1396, 2009.

[11] W. Li, J. Wei, and S. Wang, "An evolutionary-dynamic tdma slot assignment protocol for ad hoc networks," in *Proc. of IEEE WCNC*, March 2007.

[12] Y. Wang and I. Henning, "A deterministic distributed TDMA scheduling algorithm for wireless sensor networks," in *Proc. of IEEE WiCom*, 2007.

[13] S. Gandham, M. Dawande, and R. Prakash, "Link scheduling in wireless sensor networks: Distributed edge-coloring revisited," *Journal of Parallel and Distributed Computing*, vol. 68, no. 8, pp. 1122–1134, 2008.

[14] S. S. Kulkarni and M. Arumugam, "SS-TDMA: A self-stabilizing MAC for sensor networks," *Sensor network operations*, vol. 6, p. 186, 2006.

[15] W. L. Lee, A. Datta, and R. Cardell-Oliver, "FlexiTP: a flexible-schedule-based TDMA protocol for fault-tolerant and energy-efficient wireless sensor networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 6, pp. 851–864, 2008.

[16] J. Degesys, I. Rose, A. Patel, and R. Nagpal, "DESYNC: self-organizing desynchronization and TDMA on wireless sensor networks," in *Proc. of ACM IPSN*, 2007.

[17] R. G. Bai, Y. G. Qu, Y. Guo, and B. H. Zhao, "An energy-efficient TDMA MAC for wireless sensor networks," in *Proc. of IEEE APSCC*, 2007.

[18] M. J. Miller and N. H. Vaidya, "On-demand TDMA scheduling for energy conservation in sensor networks," *University of Illinois at Urbana-Champaign, Tech. Rep*, 2004.