ATR-MAC: Deadline Driven Data Collection from Smart Meters in Absence of Aggregation

Ehsan Nourbakhsh, Ravi Prakash
Department of Computer Science
The University of Texas at Dallas
{ehsaan,ravip}@utdallas.edu
http://dslab.utdallas.edu

ABSTRACT

Deployment of Smart Grid networks throughout cities introduces a realistic scenario for a large scale sensor network. Smart Meters create a wireless mesh network for receiving commands and sending metering reports back to provider company. The challenges of reliable and timely communication call for a fresh review of the well studied topic of Wireless Sensor Networks (WSN). We provide a centralized MAC solution for data collection, *ATR-MAC*, that relaxes requirements in general purpose WSNs, but emphasizes on difficulties a Smart Metering deployment will face. *ATR-MAC* considers variable reuse distances for minimizing propagation delay. It also increases efficiency of collection deadline instead of energy efficiency, a common concern of general-purpose WSN MAC protocols. We provide time and buffer size analysis of our algorithm, prove its correctness and show experimental results using implementation on a testbed.

Categories and Subject Descriptors

J.7 [**Computer Applications**]: Computers in other systems—*Smart Grid*

Keywords

smart grid; network infrastructure; evaluation

1. INTRODUCTION

It is widely assumed that smart meters will communicate their information to a data collection point through wireless connections. As the wireless communication range of the meters is limited, and there could be a variety of obstructions to signal propagation, all meters will not be able to directly communicate with the data collection point. So, we consider a model proposed in literature wherein the smart meters form a wireless mesh to relay their signal to the data collection point. Any increase in the frequency of updates from the smart meters will increase the load on the wireless mesh network.

At first glance, one may be tempted to model this problem as that of wireless sensor nodes communicating their data to a sink. However, there are some interesting differences: some favorable, and some unfavorable. Unlike several sensing scenarios, data from individual meters cannot be aggregated, nor is data loss desirable. Each meter's data must be delivered in a reliable and timely fashion to the data collection point. On the other hand, unlike sensor networks there is a certain predictability about the volume of traffic offered by each meter. So, rather than employing contention-based wireless media access, we propose that each smart meter should transmit messages as per a pre-defined schedule. The predictability of traffic pattern, coupled with relatively stable network topology, enables such transmission scheduling. Also, appropriately scheduled communication can preclude collision of wireless transmissions, increase channel throughput, and enable large scale smart meter deployment even in dense urban settings.

As Lichtensteiger *et al.* [5] discuss, it is common practice in smart grid networks to add a relaying router with higher radio range to connect a disconnected community of houses to our mesh. In a traditional tree structure for data collection such as S-MAC[18], it is implicitly assumed that the tree is more-or-less balanced. However, the relaying router in the smart metering context will have a significantly higher number of neighboring nodes, resulting in a skewed tree. This would also be the case for suburban environments in which houses are located along tertiary streets with smart meters along the street forming a long chain of wireless links. These scenarios are all similar to worst case scenarios that algorithms such as TreeMAC[15] and TRAMA[12] are designed to handle.

We propose a centralized contention-free MAC approach to determine the order in which nodes transmit their messages. As described later in Section 3, we model the problem of transmission scheduling as multiple iterations of graph coloring and graph pruning. Colors associated with edges map to time slots during which meter readings are relayed along those edges. For a smart meter with multiple neighboring meters there may be a sequence of time slots during which the meter receives data more often than it forwards data, resulting in an increase in buffer occupancy. However, such sequences are followed by periods during which the buffer is steadily emptied. Following an analysis of correctness of *ATR-MAC* in Section 4.1, we quantify maximum buffer occupancy in Section 4.2 as it will help determine the

MSWiM'13, November 3–8, 2013, Barcelona, Spain.

http://dx.doi.org/10.1145/2507924.2507963.

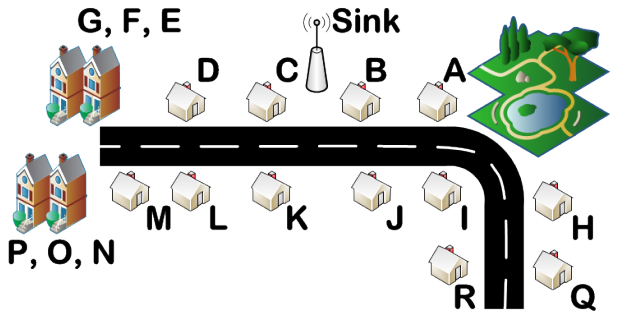

Figure 1: Map of houses in a residential area

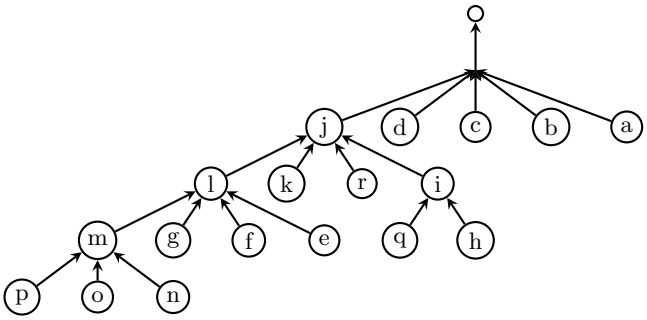

Figure 2: One possible data acquisition tree by *ATR-MAC* corresponding to map of Figure 1

maximum storage requirement of each smart meter. We also provide some discussion about time it would take for the sink to collect all reports using *ATR-MAC* in Section 4.3.

2. SYSTEM MODEL

Each smart meter i is able to communicate with some neighboring nodes, U_i. It may also sense transmissions of some other neighbors, C_i, without decoding them. If $j \in C_i$, we conclude that i is in "interference range" of j. The union of C_i and U_i a set of all its neighbors, Nbr_i. There is a network-wide requirement of sending messages with a frequency of f. So, the time corresponding to $1/f$ is the duration of one *cycle*. Nodes report their readings back to a sink node s once every cycle. Total number of nodes (smart meters) in the network is N. We denote the wireless mesh network of smart meters as a graph $G_{mesh} = (V,\ E_{mesh})$. V, the set of vertices, denotes the set of smart meters. For a pair of smart meters x and y such that $x \in Nbr_y$ and $y \in Nbr_x$, the edge $(x, y) \in E_{mesh}$. A schedule of transmissions for collecting data from all nodes to the sink will form a subgraph $G = (V,\ E)$ where $E \subseteq E_{mesh}$. E represents the set of edges that form a tree rooted at the sink. Furthermore, in the tree we consider each edge to be directed from a node to its parent, indicating the direction in which smart meter reports flow towards the sink. So, $\vec{xy}$ denotes that node x is a child of node y in the tree. The number of nodes in the subtree rooted at node i is represented by $count(i)$. So, $count(sink) = N$. The tree edge directed from node i to its parent is initially assigned an integer equal to $count(i)$. This value signifies the number of messages to be sent along this edge per cycle. Value of an edge originating from x is equal to $1 + \sum values\ of\ edges\ into\ x$.

In the context of a Smart Grid, we follow the model presented by Lichtensteiger *et al.* [5], based on an actual deployment of smart meters. Each smart meter is a sensor node and the sink acts as one of the regional data *collectors*. The collector can use other forms of communication, such as 3G wireless or wired network, to send the messages to the head-end system (HES).

It is required that in time less than or equal to one cycle duration, the sink must receive one consumption report generated by each smart meter. Unit of time scheduling is a *frame*, and frame length can vary over time. Multiple frames will be needed to send all reports to the sink node. This number of frames is called a *cycle*. Times at which node i transmits are $T(i)$, such that

$$T(i) = \{(t, d) | t : \text{time sent}, d : \text{immediate destination node}\}$$

$$T_d(s) = \{t : (t, d) \in T(s)\}$$

Time at which report sent by node i at round r is received by the sink is $R(i, r)$. Both of these times are relative to start of current round.

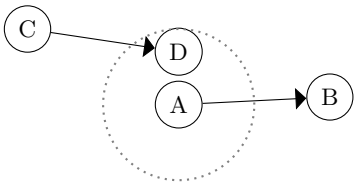

Figure 3: Two interfering edges belonging to the same network.

Consider two edges as shown in Figure 3. Simultaneous communication along the edges should not be permitted if it will cause interference at either b or d, i.e., if c is within interference range of b or a is within interference range of d. So, if both $\vec{ab}$ and $\vec{cd} \in E$ then the set of slot(s) during which node a transmits to b should be disjoint from the set of slot(s) during which c transmits to d, in other words:

$$\vec{ab} \in E \wedge \vec{cd} \in E\ \wedge\ (d \in C_a \vee b \in C_c) \Rightarrow T_b(a) \cap T_d(c) = \phi$$

Considering the above definitions in Section 2, the problem to solve is as follows. We are responsible for devising a Time Division Multiple Access (TDMA) Media Access Control (MAC) solution for Advanced Metering Infrastructure (AMI) aspect of all smart meters, routers and the sink. In this solution sink node is responsible for scheduling transmission times of the nodes.

The assumption is transmissions we schedule here are only from nodes towards the sink and have a periodic nature. The network managed by each sink can be few hundred nodes, as described by Lichtensteiger *et al.* [5]. Due to privacy concerns, reading data by each node may be encrypted and only readable by the sink node. An example of such key management, an ongoing theoretical and practical challenge, is described by Metke *et al.* [8]. This further highlights the requirement of freedom from data aggregation throughout our solution.

Each node may experience wireless interference. Nodes may be subject to interference from other devices, however our solution should avoid causing interference between nodes in the same network. It should also be able to intelligently

cope with interference from neighboring Smart Grid networks whose activity has a periodic nature. In the latter case, an intelligent method to detect and include schedule of the other device is preferable.

Main objective of this solution is to minimize cycle time of network while avoiding loss of reports from nodes in its network. As mentioned earlier, our target network is a group of Smart Meters which are connected to power sources, mitigating the energy consumption concerns. These two design goals make a third goal, scalability, achievable. In other words if $R(i,r)$ represents the time at which report sent by node i at round r is received by the sink, then:

$$
\begin{array}{ll}
min(max(R(i,j))) & \forall i \in V,\ \forall j \\
\text{subject to} & \\
count(R(i,j)) = N & \\
and & \\
x,y \in E \wedge (x \in I(y) \vee y \in I(x)) \Rightarrow T(x) \cap T(y) = \phi &
\end{array}
$$

3. SOLUTION DESCRIPTION

In a network as described in Section 2, each smart meter is transmitting its data towards the sink. But as we get closer to the sink, each node is also responsible for forwarding data it has received from its descendants in the tree. As a result as we go higher in the tree, amount of data that each node forwards increases. In each cycle a node i should be allowed to transmit in as many slots as the cardinality of the subtree rooted at it, *i.e*, $count(i)$. This is because the cardinality of the subtree rooted at node i is the number of messages it generates or relays on behalf of its descendants, per cycle. To ensure efficiency and scalability the schedule should be as short as possible and maximize concurrency.

To solve this problem, we consider the following three collision avoidance principles. 1) Each node has to be silent if any of its neighbors are receiving. 2) Node x cannot transmit to node y if y is experiencing interference. 3) Also given our assumption of half-duplex communications, x cannot transmit if y itself is transmitting. So each node i requires to *own* at least $|Nbr_i|$ silent time slots to be able to receive data from its neighbors. It also requires one dedicated time slot to transmit to next hop towards sink in that frame. So if frame length is set to $1 + \max_{\forall i} |Nbr_i|$, each node in the network is guaranteed to have at least one time slot to transmit towards the sink. This yields **correctness**. But it is not an efficient solution, because many allocated time slots will be wasted when their owner node has no more data to transmit.

Now, to address **efficiency** we try to optimize cycle length by allowing reuse of time slots. If each node i knows and broadcasts amount of traffic it has to forward, adjacent nodes will know when it has no more packets to transmit. This allows them to reduce frame length and as a result reuse dedicated time slots to node i. As time passes, more nodes are silent and in the end only the node with highest ratio of descendants to frame length is still transmitting. We form a tree structure and use it to count descendants of each node and gather information about its neighbors.

Figure 1 shows an example, where smart meters a to r are located along a street. Smart meters e, f, g are on one building and n, o, p are on another, while the rest are meters belonging to houses in the area. A hierarchy similar to Figure 2 might result. In this case, nodes j, l, m relay data to higher layers but the rest are leaves.

Time slot assignment is done in a centralized manner because the schedule for one cycle is calculated once and repeated after that unless network changes occur. So we construct a modified graph $G^{'} = (V^{'}, E^{'})$ such that: $V^{'} = \{(a,b) : a,b \in V \wedge \vec{ab} \in E\}$ and $E^{'} = \{(a,b)(c,d) : \vec{ab} \in E \wedge \vec{cd} \in E \wedge (c \in Nbr(b) \vee a \in Nbr(d))\}$. Vertices that are neighbors in $G^{'}$ represent edges in the actual tree that should not be scheduled for communication during the same time slot because concurrent communication along these tree edges will result in interference.

Algorithm 1: Algorithm to assign time slots

```
slot_number ← 0;
G'' ← G';
forall the (x,y) ∈ V'' do
    begin
        find (x',y') ∈ V' corresponding to (x,y)
        (x,y).integer ← (x',y').integer
begin
    changed ← TRUE;
    while G'' ≠ NULL do
        if changed == TRUE then
            perform vertex coloring of G'' ;
            n =number of colors in vertex color of G'' ;
        ∀i : 1 ≤ i ≤ n: for all vertices (x,y) of G''
        assigned color c_i schedule transmission by meter
        x to meter y during time slot slot_number + i;
        slot_number ← slot_number + n ;
        forall the (x,y) ∈ V'' do
            (x,y).integer − −
        changed ← FALSE;
        forall the (x,y) ∈ V'' do
            if (x,y).integer == 0 then
                delete (x,y) and all edges incident on it;
                changed ← TRUE;
```

Prior to initiating the execution of the scheduling solution, as described in Algorithm 1, each vertex (a,b) in $G^{'}$ is assigned the same integer value as the value of the corresponding tree edge $\vec{ab}$ in G. Then the following steps are executed:

1. A vertex coloring heuristic is run on the graph $G^{'}$. The colors assigned to each vertex correspond to a time slot during which one unit of information can be transmitted in a conflict-free manner from a smart meter to its parent in the tree.

2. Schedule one transmission by each smart meter to its parent in the tree, based on the color assigned to the edge between the node and its parent. The corresponding sequence of slots constitutes a frame.

3. After a run of the steps described above, one unit of data has moved from each smart meter to its parent in the tree. So, we prune graph $G^{'}$ in the following manner: reduce the value associated with each vertex by 1

(to denote that it has one less data value to transfer to its parent until completion of the cycle), and eliminate all vertices whose value is reduced to zero as a result, and all the edges incident on the removed vertices.

4. If the graph is pruned completely, exit.
5. If the graph topology is unchanged, go back to Step 2.
6. If the graph topology is modified, go back to Step 1.

Each vertex in $G^{'}$ gets scheduled for as many time slots at the integer value associated with it. This means that for each edge $\vec{ij}$ in the original tree, there are as many time slots as the number of nodes in the subtree rooted at i. There are as many frames in a cycle as the number of times Steps 2 and 3 of the algorithm are executed. Different frames may have different lengths, depending on the number of nodes in the graph $G^{''}$.

Like any TDMA schedule, interferences from outside of the network may result in some transmissions to fail. As a result, nodes with outstanding messages will transmit their messages to their parents based on a Carrier sense Multiple Access (CSMA) protocol after the TDMA schedule is finished. *ATR-MAC* records interference occurrences, and if interference observed by a node repeats periodically or is constantly present the schedule is updated to avoid allocating transmissions in time slots affected by it (Algorithm 2 described in Section 3).

3.1 Neighbor Discovery and Tree Formation

Prior to executing the scheduling algorithm we need to form a tree of the smart meters rooted at the sink, as described below. First, each smart meter discovers its neighbors by listening for heart beat messages broadcasted periodically. Sink will trigger formation of tree by broadcasting a JOIN message to all its neighbors. Each smart meter will absorb the JOIN message if it has not already joined the tree as the child of another smart meter or sink. On joining the tree a smart meter will also broadcast the JOIN message to its neighbors. Otherwise it will send a negative acknowledgment to the sender of the JOIN message. In this manner the JOIN message diffuses through the network of smart meters and forms a tree rooted at the sink. Starting from the leaves of the tree and moving up, meter i communicates the number of meters in the subtree rooted at it (N_i), all its children, and the set of all its neighbors (Nbr_i) to its parent. Thus, the sink ultimately gets all the information it needs to reconstruct the tree and execute Algorithm 1.

3.2 Refinement and Maintenance

Neighbor discovery and tree formation is required when the network is being set up, or a significant change to number of nodes or environment has occurred. However, an urban area deployment of nodes will be exposed to a variable radio propagation environment. To add to this factor, it is possible that installation of new meters is done in rounds. This means a sizable number of sensors (meters) are installed about the same time. But after they have initiated and formed their network, new nodes are added gradually. Proposed solution should handle these additions gracefully and without disruption. The rest of this section describes this solution.

To maintain the neighbor and conflict relationships more accurately, the sink periodically runs a scanning routine.Once the node identities are known, the sink distributes a schedule of the following format

$$S = \{(i,t) | i \in V, t = \textit{unique time slot from a reference time}\}$$

Each node is present exactly once in this schedule, and the beginning and end of the scanning routine are announced to all nodes based on the synchronized time between the nodes. The sink also announces a time length, τ.

After the reference time has passed, each node transmits a message containing its identity and pads it so the resulting message lasts τ units of time. If a node is not transmitting, it is sensing the environment. Each node in sensing mode records when it receives a message or it detects a transmission without decoding, and timestamps each observation. Each of these observations are recorded as (t, y, id) where t is the time of observation, y is type of observation, i.e., a meaningful reception of the message or simply sensing an ongoing transmission. Since transmissions by all nodes have the same length, τ, a node can distinguish between background noise. Once the scanning routine is finished, all nodes send the set of their observations to the sink.

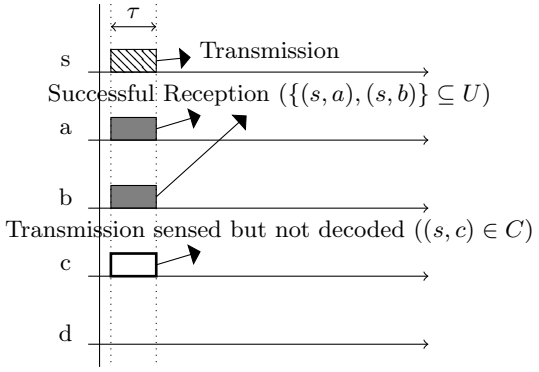

Figure 4: Overview of scanning routine based on schedule provided by sink

The sink is able to use the above data to infer which nodes are able to communicate with each other or are affected by transmission of any other node. This results in two sets of tuples, conflict $C = (i, j)$ and communication sets $U = (i, j)$. If (i, j) is present in C, it means transmissions by node i are sensed at node j and node j will experience interference if it is concurrently receiving a message from another node. However, node j is not capable of decoding the message sent by i. But if (i, j) is present in U, it means node j can successfully receive messages sent from node i. Based on this definition, $C \bigcap U = \emptyset$. Please note that the sets are not necessarily symmetrical. This means presence or absence of (i, j) in one of the sets does not necessarily impact presence or absence of (j, i) in the same set or the other. Thus the Nbr_i set defined earlier is $\{j : (i, j) \in U \vee (j, i) \in C\}$. An example of the transmission schedules and some of the resulting sets are shown in Figure 4.

We also need to maintain observed quality of time slots. If a time slot is marked as Interference more than *threshold* number of times, the node will increase number of its neighbors and asks the sink to re-assign time slots. Details are shown in Algorithm 2.

Algorithm 2: Maintaining scheduled time slots over time

```
change ← FALSE;
if RSS quality of current time slot is undesirable then
    update the history array with negative observation;
    use above array to count how many times this time
    slot has had low RSS quality;
    if count(low quality observations) ≥ threshold
    then
        mark this time slot as a conflict with itself;
        changed ← TRUE;
    if changed then
        recalculate maximum neighbor count;
        if neighbor set has changed then
            report new neighbor set to sink through
            parent;
            sink runs Algorithm 1 on new set;
        if sink responded with new time slots then
            replace current schedule;
else
    update the history array with positive observation;
```

3.3 External Algorithms: Coloring and Time Synchronizing

As any other TDMA MAC protocol, we require time synchronization between the nodes to prevent collisions. We use work by Maróti *et al.* [7] for time synchronization. We further enhance time synchronization between the nodes by piggybacking time information in the periodic heart beat messages they transmit.

For coloring, we utilize the heuristic algorithm by Welsh *et al.* [16]. It is worth mentioning that the coloring is used only when scheduling the network, and is only required when the network topology is changed. Furthermore, the G'' graph from Algorithm 1 is only re-colored when pruning has changed it.

4. ANALYSIS

4.1 Proof of Correctness

DEFINITION 4.1. *A scheduling algorithm A provides **weak interference avoidance** if no two nodes within reception range of each other use same TDMA time slot for transmission.*

DEFINITION 4.2. *A scheduling algorithm A provides **strong interference avoidance** if no two nodes a and b within interference range of each other use the same TDMA time slot for transmission. This requirement holds even if a and b are not able to communicate.*

LEMMA 4.1. ATR-MAC *can provide weak interference avoidance after the Time Assignment phase.*

Proof. **Case 1**: No link failure has occurred.

We provide proof by contradiction. Assume nodes a and b are assigned the same time slot t to transmit to their respective parents. Each of these nodes will use a time slot only if no other node has indicated they are using them. Since a and b are in reception range of each other, they will both report identity of the other in their neighbor set. Sink will assign them time slots such that each is silent when the other is active according to Algorithm 1.

Case 2: Link failure has occurred.

If the wireless link has broken between h and its parent p, they both realize this approximately at the same time since they both use same failure mechanism. Until then t is still reserved for h, and after failure detection h will disassociate itself and inform its children. So even in this case t will not be used by more than one node. □

LEMMA 4.2. ATR-MAC *can provide strong interference avoidance after limited number of runs of Maintenance procedure (Section 3.2).*

Proof. Assume the interference is happening for nodes a and b. If both nodes belong to the same network, weak interference avoidance can also indicate the strong case. If the two nodes are able to receive messages from each other, during the Time Assignment phase they marked each other's time slots as Interference. But it is possible that they are unable to communicate, for example when they are in interference range of each other but not in communication range.

For this case assume Maintenance procedure (Section 3.2) has run more than *threshold* number of times. So the counter is larger than the threshold, and that time slot will be marked as Interference. Then it will obtain an updated schedule that avoids using that time slot. It is very likely that the frame and cycle length of these networks are not the same and one, for example a, has smaller frame length. Value of *threshold* decides if this irregular interruption is sufficient to mark the time slot as Interference or it is ignored. □

4.2 Buffer size

In the first frame each node i sends its own message to its parent, and receives messages from its immediate children. For all subsequent frames it relays one message per frame from messages that are in its buffer. Number of messages buffered at i increases as long as at least two of its children are forwarding packets to it. As a result buffer length starts decreasing from its maximum length when there are no more incoming messages, i.e., its child with largest subtree has forwarded its last packet. So at frame *count(largest subtree of i)* node i has received all $count(i)$ messages in its own subtree and its queue starts shrinking. Since i was sending one message per frame to its parent, total size of buffer is upper bound by $count(i) - count$(largest subtree of i).

As stated earlier, our algorithm is directly targeting situations similar to Figure 2 where the tree is highly skewed. As a result, the above value is going to be very small. The worst case scenario occurs when the largest subtree of node i has size equal to 1, i.e., is a leaf node. Hence, all other "subtree"s are also leaf nodes. In this unlikely case, the size of buffer for node a is equal to $n-1$. However, the schedule created for this node is going to be perfectly packed: first round includes all of its children sending it their messages (and a sending its own to the sink), and rounds two to $n+1$ will be node a forwarding the buffered messages to the sink. Imposing a limit on the buffer size to avoid such extreme scenarios may lengthen the active time of the network but eliminate this problem.

4.3 Cycle Length

Let us assume a minimum of k edges are removed from G'' in each run of the *while* loop in Algorithm 1. The value of k depends on how many leaves were eliminated from the tree in each frame. Since each run of this while loop corresponds to one frame of the schedule, upper bound for total number of frames will be

$$\frac{\text{total edges}}{\text{estimated number of eliminated nodes in each round}} = \lceil \frac{N}{k} \rceil$$

Number of time slots in each frame is variable and equals to number of colors used in vertex coloring of G'' in that frame. The coloring algorithm we use [16] provides an upper bound for number of colors used. This upper bound is equal to maximum degree of all nodes plus one. Since graph G'' is shrinking as *ATR-MAC* assigns time slots in Algorithm 1, it is safe to assume the maximum number of colors is also shrinking. In the worst case scenario this value will be the first number of colors used which we call c.

In short, the *while* loop runs $\lceil \frac{N}{k} \rceil$ times and each time a maximum of c time slots are scheduled. So the cycle length will be equal to $\lceil \frac{N}{k} \rceil \times c$.

One might argue that in the worst case scenario, $k = 1$ and $c = N$. But it would mean that every node is causing interference to all other nodes, and a chain is formed so each frame only eliminates one node from G''. A more realistic assumption would lead to values for k and c to be comparatively small constant values proportional to maximum direct neighbors among all nodes.This will result in a cycle size of $\lceil \frac{c}{k} \times N \rceil = O(N)$.

5. RELATED WORK

The focus of earlier sensor network MAC protocols was to address energy conservation using low-duty cycling. B-MAC [10] is one of the most well known examples which tries to reduce idle listening with adaptive sampling. S-MAC [18] is also targeting energy consumption by customized sleep cycles. Given that our sensors in this current scenario are smart meters, the higher level of focus on energy saving can be relaxed. Furthermore, the attachment of said meters to houses eliminates possibility of constant movement and makes interference and variable noise a higher concern. In other words, we are able to create more optimized solution for the specific scenario of Smart Grid networks rather than providing a general purpose solution.

Data gathering MAC (D-MAC) [6] can be viewed as an extension to S-MAC with a time adapting scheme. Active time of each node is determined based on its depth in the formed tree. To avoid interference with higher layers, a 3 time slot gap is used. However, length of this gap is based on the assumption that no higher level of internal interference beyond two hops is present. D-MAC does not use TDMA and relies on CSMA for contention avoidance in local neighborhoods.

TreeMAC [15] utilizes a tree architecture to create a localized TDMA MAC protocol that ensures conflict-free sending, receiving and snooping. Frame length in TreeMAC is three and cycle size is calculated based on size of the network. Similar to D-MAC, TreeMAC allocates transmission time within one frame based on the depth of each node in formed tree. *ATR-MAC* has a more generalized approach as we use number of children, and not only depth in tree, to allocate time slots. We also change frame length to address declining number of senders in each frame and as a result cycle time is minimized.

TRAMA [12] is also a TDMA based approach which uses flow information in assigning time slots. Nodes are required to periodically report their future traffic trend. They can also allow re-use of time slots if the actual traffic is less than the amount predicted. The obvious downside of this approach is overhead of information sharing. FLAMA [11] addresses this issue by only requesting flow information when necessary. However, accuracy of resulting schedules depends on how closely changes can be predicted. By focusing on requirements and assumptions specific to smart metering, we are able to relax these issues.

DRAND [14] is a distributed scheduling algorithm that uses graph coloring for time slot reuse. Z-MAC [13] uses DRAND for distributed TDMA time slot arrangement that extends to the two-hop neighbors. Z-MAC distinguishes two modes of operation, Low Contention Level (LCL) and High Contention Level (HCL). In LCL, any node can compete to transmit in any slot, but in HCL, only the owners of the current slot and their one-hop neighbors are allowed to compete. Underlying assumption of DRAND is all nodes are similar in internal interference and main interference is contained within two hop neighborhood. Z-MAC assign frame sizes of fixed length, given that no assumption about the application is made in its design.

Iyer *et al.* [3] provide an analysis of performance of two routing protocols on a wireless mesh network used for Advanced Metering Infrastructure (AMI). This marks an interesting path for developing customized routing protocols or adjusting existing ones for such applications.

6. EVALUATION

We used 24 MICA2 868-916MHz motes [1] for evaluating our algorithm. These experiments were conducted using our testbed, ASSERT. Complete details of *ASSERT* are provided in [9]. In our setup, each node has up to 8 neighbors which is consistent with connectivity requirements shown by Xue *et al.* [17]. Given this configuration, we expect our results to be scalable for larger networks.

Sensed values as well as quality of the links between motes were controlled to emulate changes in environment. For comparison, we ran Collection Tree Protocol (CTP) designed by Gnawali *et al.* [2] on B-MAC [10] as well as Z-MAC [13]. Duration of each experiment is 30 minutes. Sensors report their virtual reading every 2 minutes to one sink.

Parameter	value
Transmission Power	-10dBm
Slot size	20 ms
Heartbeat interval	30s

Table 1: *ATR-MAC* experiment parameters

For the initial step of constructing the neighbor and conflict graph, we set $\tau = 32.5ms$. To reduce the chance of missing detections, each node repeats transmissions 11 times. Receiving nodes report a neighbor/conflict only if they detect more than 6 such detections. There is a constant silence time following each transmission, and a larger one before the next node started its sequence. This total time for each node would be $5,500ms$. Thus, the total time for this step is $k \times 5.5s$, where k is total number of nodes controlled by

same root node. This constant conflict detection time is not included in reported setup times.

We used attenuation traces obtained by Lee *et al.* [4] from real world measurements to recreate those topologies. For each experiment size, we chose three *connected* subsets of nodes and repeated our experiment on each of these three topologies. The provided results are an average over these three different topologies, unless otherwise stated. Table 1 shows some of the main parameters used in our experiments.

6.1 Results and Comparison

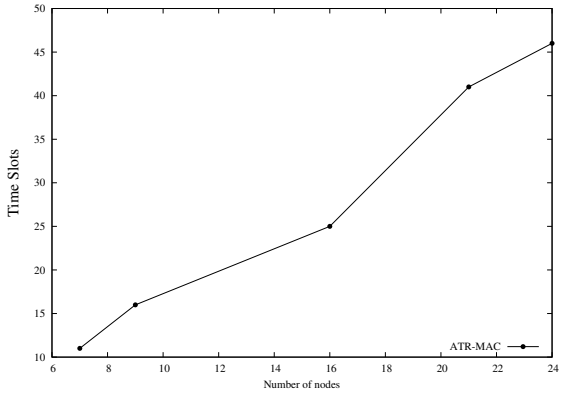

Figure 5: Average calculated cycle duration for three runs of *ATR-MAC* on multiple topologies

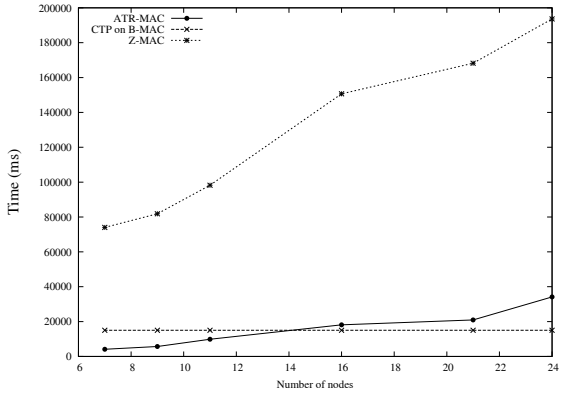

Figure 6: Average setup time versus number of sites

As shown in Figure 6, amount of time taken by *ATR-MAC* to form the trees is relatively small. Setup time for CTP is not dynamic and is adjustable. In our experiments we set this setup phase to be 15 seconds. Setup time of Z-MAC is mainly dedicated to running DRAND. We would like to acknowledge that DRAND would have a better performance in comparison with centralized algorithms when network size is large. However, in a smart metering application each sink is only responsible for a smaller number of nodes. Reaction of CTP to network changes is also only visible when new queries are submitted, and thus not comparable with our proactive approach. Cycle duration calculated by sink using *ATR-MAC* is shown in Figure 5.

To measure the reaction time of nodes, we create a random link failure. The time network takes to react to this interruption is shown in Figure 7. The small amount of time can be attributed to the fact that most of the time it is a local disruption without major message transmissions with other neighbors. However, a link failure between a node with higher number of children and its parent can result in higher reaction time. Cycle changes are sent to nodes after end of current cycle and before start of the next one. Similar reaction in Z-MAC would require running DRAND locally on affected nodes. But a common approach is to schedule DRAND to run periodically to address network changes. In our experiments DRAND was scheduled to run every 10 minutes.

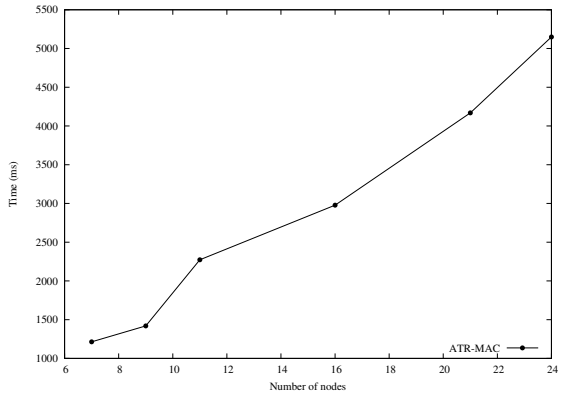

Figure 7: Average reaction time to network changes

Time at which sink receives last pending message for current cycle is considered propagation time. This propagation time is shown in Figure 9. As it can be observed, performance of *ATR-MAC* is close to Z-MAC, and a much better improvement over performance of CTP. The values reported are average of maximum delays observed by all the nodes.

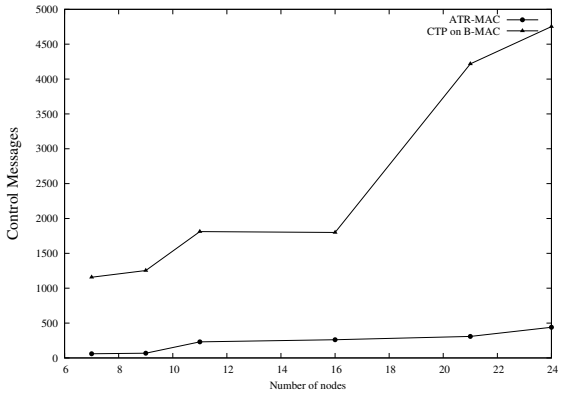

Figure 8: Average control messages overhead

In comparison to CTP, *ATR-MAC* has far less control messages (Figure 8). While B-MAC does not require control messages, CTP will require beacons and route establishment messages which are included in the figure.

7. CONCLUSION AND FUTURE WORK

We proposed *ATR-MAC*, a centralized MAC specifically designed taking into account specific requirements of smart

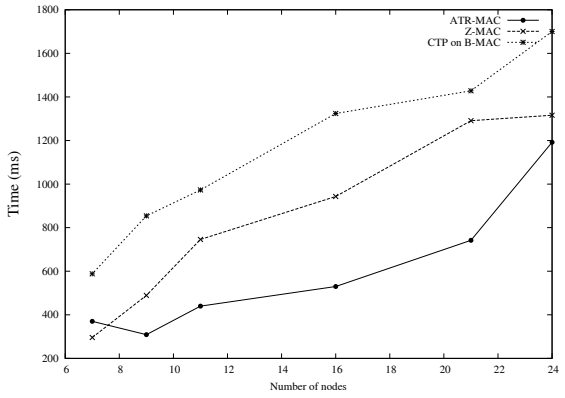

Figure 9: Average of Maximum Propagation time

metering wireless networks. We aim to enhance our work by designing a distributed version of this MAC protocol.

As demonstrated by Lemma 3.2, we can handle interferences caused by other generic devices or networks. *ATR-MAC* also provisions an intelligent way to detect and coexist with periodic interference such as other AMI networks. An extension of this work would study precise impact of presence of multiple networks in vicinity of each other. By focusing on timeliness and contention reduction, *ATR-MAC* is able to provide a scalable solution for data gathering without aggregation in Smart Grid networks. *ATR-MAC* performs efficiently where most MAC protocols perform suboptimally. This is made possible by relaxing the energy efficiency requirement, as is permitted in Smart Grid applications.

8. REFERENCES

[1] MICA2 Platform by MEMSIC, Inc.

[2] O. Gnawali, R. Fonseca, K. Jamieson, D. Moss, and P. Levis. Collection Tree Protocol. In *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, SenSys '09, pages 1–14, 2009.

[3] G. Iyer, P. Agrawal, E. Monnerie, and R.S. Cardozo. Performance Analysis of Wireless Mesh Routing Protocols for Smart Utility Networks. In *IEEE SmartGridComm*, pages 114 –119, 2011.

[4] H.J. Lee, A. Cerpa, and P. Levis. Improving Wireless Simulation Through Noise Modeling. In *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, page 30, 2007.

[5] Bill Lichtensteiger, Branko Bjelajec, Christian Muller, and Christian Wietfeld. RF MeshSystems for Smart Metering: System Architecture and Performance. In *IEEE SmartGridComm*, pages 379–384, 2010.

[6] Gang Lu, Bhaskar Krishnamachari, and Cauligi S. Raghavendra. An Adaptive Energy-Efficient and Low-Latency MAC for Data Gathering in Wireless Sensor Networks. In *IEEE IPDPS*, 2004.

[7] Miklós Maróti, Branislav Kusy, Gyula Simon, and Ákos Lédeczi. The Flooding Time Synchronization Protocol. In *Proceedings of the 2nd international conference on Embedded networked sensor systems*, SenSys '04, page 39–49, 2004.

[8] A.R. Metke and R.L. Ekl. Security technology for smart grid networks. *IEEE SmartGridComm*, 1(1):99–107, 2010.

[9] Ehsan Nourbakhsh, Jeff Dix, Paul Johnson, Ryan Burchfield, S. Venkatesan, Neeraj Mittal, and Ravi Prakash. ASSERT: A Wireless Networking Testbed. In *Proceedings of TridentCom '10: LNICST 46*, pages 209–219, 2010.

[10] J. Polastre, J. Hill, and D. Culler. Versatile Low Power Media Access for Wireless Sensor Networks. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 03–05, 2004.

[11] V. Rajendran, J.J. Garcia-Luna-Aveces, and K. Obraczka. Energy-efficient, application-aware medium access for sensor networks. In *IEEE International Conference on Mobile Adhoc and Sensor Systems Conference*, pages 8 pp. –630, 2005.

[12] Venkatesh Rajendran, Katia Obraczka, and J. J. Garcia-Luna-Aceves. Energy-efficient, Collision-free Medium Access Control for Wireless Sensor Networks. *Wireless Networks*, 12:63–78, 2006.

[13] I. Rhee, A. Warrier, M. Aia, J. Min, and M.L. Sichitiu. Z-MAC: A Hybrid MAC for Wireless Sensor Networks. *IEEE/ACM Transactions on Networking (TON)*, 16(3):511–524, 2008.

[14] I. Rhee, A. Warrier, J. Min, and L. Xu. DRAND: Distributed Randomized TDMA Scheduling for Wireless Ad Hoc Networks. *IEEE Transactions on Mobile Computing*, pages 1384–1396, 2009.

[15] Wen-Zhan Song, Renjie Huang, Behrooz Shirazi, and Richard LaHusen. TreeMAC: Localized TDMA MAC Protocol for Real-time High-data-rate Sensor Networks. *Pervasive and Mobile Computing*, 5(6):750–765, 2009.

[16] D.J.A. Welsh and MB Powell. An Upper Bound for The Chromatic Number of a Graph and Its Application to Timetabling Problems. *The Computer Journal*, 10(1):85, 1967.

[17] Feng Xue and P.R. Kumar. The Number of Neighbors Needed for Connectivity of Wireless Networks. *Wireless networks*, 10(2):169–181, 2004.

[18] W. Ye, J. Heidemann, and D. Estrin. An Energy-efficient MAC Protocol for Wireless Sensor Networks. In *IEEE Proceedings Twenty-First INFOCOM 2002.*, volume 3, pages 1567–1576, 2002.