TreeDMA: a Hybrid MAC/Routing Solution for Small-Scale Wireless Networks

Clement Kam and Curt Schurgers
Electrical and Computer Engineering Department
University of California, San Diego
La Jolla, CA 92093–0407
Email: ckam@ucsd.edu, curts@ece.ucsd.edu

***Abstract*—Vehicle-based networks need energy-efficient communication to extend the battery lifetime, which subsequently extends the mission time of the node. We observe that these kinds of networks operate under different assumptions than traditional static sensor networks. The main difference is that these are *small-scale* networks. This motivates the need for a protocol that is optimized for such scenarios. To handle the communication, we introduce a combined MAC and routing protocol, TreeDMA, that uses a contention-free MAC in conjunction with beaconing and overhearing. We show through simulation that for these kinds of networks, the energy consumption and latency is comparable to an "ideal" omniscient protocol and superior to traditional layered protocols.**

I. Introduction

There is a lot of interest in utilizing autonomous networks of sensor-equipped mobile vehicles for sampling missions, such as search and rescue [1] and environmental mapping and monitoring [2]. Unlike traditional static sensor network applications, these networked vehicles are dispatched to the area of interest to perform their sensing task. In [3], the authors describe the problem of a hazardous materials leak in a damaged structure. They address the problem of deploying a team of robots equipped with chemical sensors to return real-time data indicating the location and concentration of hazards. Similar problems being studied include using unmanned aerial vehicles (UAVs) in dynamic environments such as wildland fire fighting or studying tornado formation [4], or using autonomous underwater vehicles (AUVs) for underwater environment monitoring [2]. An important area of study is the design of communication protocols that provide data delivery for these kinds of applications.

A lot of networking protocols have been proposed for autonomous networks, each designed with certain attributes in mind. One important attribute is *scalability*, since people envision deploying hundreds of tiny sensor nodes. *Energy efficiency* is a concern since the lifetime of the network is tied to the nodes' on-board batteries. Another desirable attribute is low *latency*, since the data may be time sensitive and a large network can incur serious delays. High *throughput* may be desirable for higher data rate applications, and also as it affects energy consumption and latency. It is often assumed that these attributes are necessary over all types of sensor networks, but in actuality, the importance of each attribute is application-dependant.

The authors acknowledge the generous financial support of the National Security Education Center (NSEC) at Los Alamos National Laboratory (LANL).

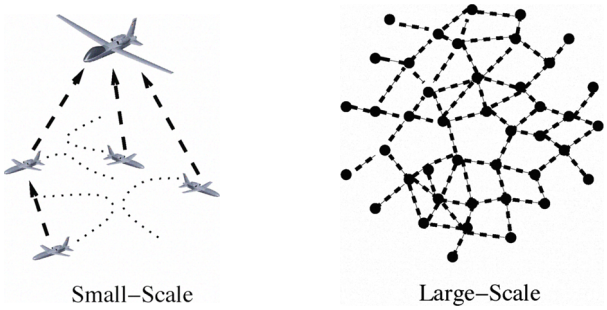


Fig. 1. Small-scale vehicle-based networks vs. large-scale sensor networks.

For applications involving networks of mobile vehicles, there are certain features that distinguish these networks from how people traditionally view sensor networks. First, these vehicles are more expensive to make and deploy than static sensor nodes. Therefore, these networks are characteristically of a *small scale*, with network sizes on the order of 10 to a few tens of nodes. Also, with a smaller network, centralized control is not only feasible but is commonly used in such applications, which limits the use for in-network processing. Therefore, these sensor networks are likely to be *data-gathering* networks, where all the nodes forward their raw data to a single access point, and possibly to a backbone network for further processing. This type of many-to-one traffic is also referred to in the literature as convergecast traffic.

This work focuses on *small-scale*, *data-gathering* mobile networks. Protocol design for vehicle-based sensor networks is different from that of traditional sensor networks (Fig. 1). First, scalability is not an issue for small networks. Also, latency can be kept sufficiently low when there are only a few nodes sharing the channel. For these applications where nodes periodically report sensor data, throughput is not a concern for low resolution data. Since scalability, latency, and throughput are not major concerns, energy efficiency is brought to the forefront.

We thus define our problem as seeking an energy-efficient networking strategy optimized for small-scale, data-gathering

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858

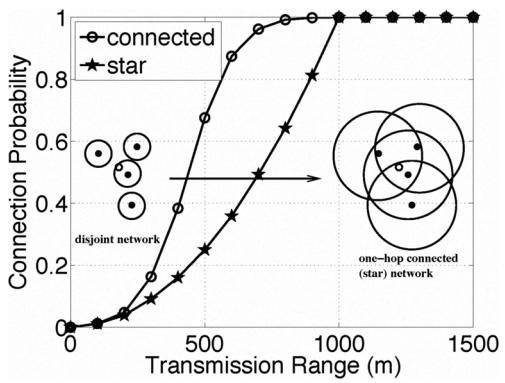


Fig. 2. Connectivity vs. transmission range, 15 node network placed in a disk of 1 kilometer radius.

networks. Before we present our solution, we first conduct a preliminary topology study to motivate our approach.

II. Topology Study

To assist in our networking approach, we conducted a preliminary topology study to gain some insight into the very nature of these types of networks. We considered deploying a small number of nodes to an area of interest, which we define to be a disk of radius R. The nodes need to send real-time data to a sink residing in the center of the disk. Therefore, every node must have at least a multi-hop route to the sink, or what we refer to as a *connected* network. A special case of this is when all nodes can send directly to the sink, which we refer to as a *star* network. When investigating the topologies that arise for various transmission ranges, we discovered an interesting result. For the typical network sizes that we expect (a few tens of nodes), it turns out that there is a very small difference between the transmission range needed for a connected network and the range needed for a star network.

For example, consider designing a system consisting of 15 nodes being deployed in a disk of 1 kilometer radius. To ensure real-time delivery, a radio technology must be chosen with a transmission range sufficiently large to yield a connected network. We plotted the probabilities of connected and star topologies as transmission range varies in Fig. 2. We see that in order for all nodes to have a route to the sink with high probability (> 0.99), we need a transmission range of 800 meters. However, it is more likely that the system will be overdesigned to account for environmental variability, or for the uncertainty and variability of the deployment area. Therefore, it is likely that a longer transmission range will be chosen. [1] Since the curves for connected networks and star networks are so close together, overdesigning the system will most likely lead to having a star network (with probability > 0.99 for a range of 970 meters). So in practical small networks, a vast majority of the nodes will probably be able to communicate directly with the sink, making multi-hop routing unnecessary for them.

[1]Also, transmission range itself is vaguely defined, and only a discrete number of radio technologies are available.

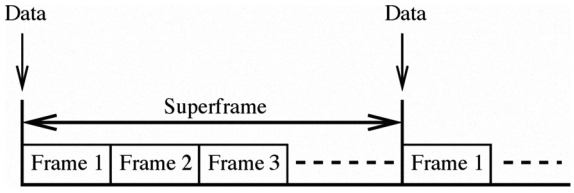


Fig. 3. A superframe.

However, on rare occasion, one of these mobile vehicles may wander outside of direct transmission range of the sink or have its connection with the sink otherwise broken. In the example of the chemical sensing robots, a robot may find a concentration of chemicals that is just outside of the anticipated sensing area. The robot needs to establish a multi-hop route to continue reporting data to the sink. Therefore, it is still important to retain the ability to do multi-hop routing. But given that these kinds of scenarios are the exception rather than the rule, we adopt a design philosophy that leverages the nature of these small, convergecast networks, so that we optimize the protocol for a mostly star topology.

With full-scale routing being unnecessary, it seems that the best approach to the networking problem is to break through the layers. The idea is to have a protocol that opportunistically routes, only when necessary. This is achieved by incorporating the routing functions directly into the MAC layer. In contrast to typical layered protocols, our cross-layered approach eliminates the overhead of special route discovery and establishment/maintenance packets, saving energy and extending the battery lifetime. Note that since these networks are mobile, traditional routing cannot simply rely on cached information, so route discovery would remain an overhead.

III. TreeDMA Protocol Description

We propose TreeDMA, a combined MAC and routing protocol that provides *reliable*, *energy-efficient*, *periodic data delivery* from a *small network of mobile nodes to a sink*. For applications where data is sent periodically, a contention-free MAC protocol, such as TDMA, is preferable because it eliminates collisions. Due to the small network size, it is possible to assign each node a fixed time slot to send in each frame. We define the period at which data is sent to be a fixed number of frames, called a *superframe* (Fig. 3). Also, the protocol is repeated at the start of each superframe to adapt to any topology changes due to mobility. To reduce unnecessary receptions and idle listening, each node will employ a listening schedule to govern when the radio will be awake or asleep.

Since the full TreeDMA involves many subtle details for handling special cases, we will gradually present the protocol operation in three steps. We start with a simple case where we have the most typical topology, a star network. Then we will describe the operation for a slightly more complicated network, in which one node requires a multi-hop route. Finally,

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858

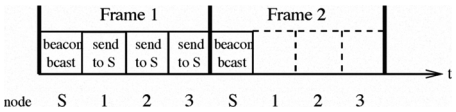


Fig. 4. Star network operation.

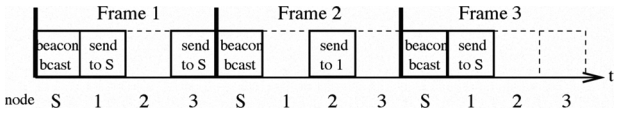


Fig. 5. Operation with one two-hop route.

we will describe the full MAC/Routing protocol, complete with sleeping capability.

A. TreeDMA for Star Networks

We begin with the simple case of a star network, where no multi-hop routes are needed. The operation is shown in Fig. 4. In the first slot of the superframe, the sink S announces its presence to all the nodes by broadcasting a beacon message. Since all of the nodes in this scenario are in range of the sink, they receive the beacon and learn that they can send directly to the sink. In the remainder of the first frame, the sink listens in all the other nodes' sending slots (note that regular nodes do not need to listen in each other's slots). The sink receives data from all nodes, and in doing so, learns that no node is in need of a multi-hop path. At the start of the second frame, the sink sends a beacon indicating that it received all the data, and nodes learn that no one needs to act as a forwarding node. The nodes can therefore go to sleep for the rest of the superframe.

B. TreeDMA with One Two-Hop Route

In this scenario, there is one node that requires a multi-hop route. The operation is shown in Fig. 5. As in case A above, the operation starts with a beacon message. However, now all but one node receive it (e.g., node 2). In the remainder of the frame, the sink listens to all of the nodes' slots and receives from those within range (e.g., nodes 1,3). The two-hop node (e.g., node 2) does not hear the beacon, so it listens in the other nodes' slots to try to overhear a transmission, and it thus learns of a forwarding node (e.g., node 1). At the start of the second frame, the sink sends a beacon saying that the two-hop node needs a route. The nodes listen to the two-hop node's slot, and the two-hop node sends data to the forwarding node (e.g., 2→1) that it overheard in the first frame, thus establishing its route. The way that nodes learn to go to sleep will be described in the full protocol description.

C. Full TreeDMA Protocol

There may be scenarios in which some nodes require a route of 3 or more hops. TreeDMA is equipped to handle the topology of any connected network. Here we describe the operation of the protocol for the general case. The pseudocode for TreeDMA is shown in Figs. 6-7.

1) Network Tree Construction: The previous examples describe the techniques of beaconing and overhearing to get data from the nodes to the sink. The idea behind the full protocol is to use those techniques to efficiently construct a routing tree in an opportunistic way. A routing tree describes all routes to the root of the tree, which is sufficient for convergecast traffic, where only a route to the sink is required. For the listening schedule, each node needs only to keep track of the node to which it sends (its parent) and the nodes from which it receives (its children).

Algorithm 1 TreeDMA Pseudocode

```
Wake up at start of superframe.
Initialize waitNextFrame = listenOneFrame = false.
for all slots do
  if start of a frame then
    Algorithm 2: Set Listening Schedule
  end if
  if my sending slot then
    Algorithm 3: Handle Sending
  end if
  if schedule says to listen AND I receive a packet then
    Algorithm 4: Handle Packet
  end if
end for
```

Algorithm 2 Set Listening Schedule

```
if listenOneFrame then
  listen in slots of nodes without routes
  listenOneFrame=false
end if
if no route then
  listen in non-sink sending slots
  waitNextFrame=true
else
  listen in parent's and children's slots
  if waitNextFrame then
    listenOneFrame=true
  end if
  waitNextFrame=false
end if
```

Fig. 6. TreeDMA Pseudocode (part 1).

The tree constructing is initiated by a beacon message from the sink. The first level of nodes join the tree by responding with data, and other nodes, not receiving the beacon, know they need to listen for data transmissions to later become affiliated with the tree. The idea is that the beacon, which announces the sink location, propagates outward to all nodes in the form of overheard data transmissions, thus performing an implicit type of routing. Although not an efficient solution in generic multi-hop routing, in these small networks, which are dominated by one-hop routes, this solution is a powerful one.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858

Algorithm 3 Handle Sending

```
if I am sink then
  send beacon
else if NOT sent any packets AND no packet to send AND have parent then
  send control packet to parent
else if have packet to send AND have parent then
  send packet to parent
end if
```

Algorithm 4 Handle Packet

```
update REB
if packet is a beacon AND I have no parent then
  add sink as parent
else if packet is addressed to me AND I am not the sink then
  queue packet to be forwarded
  add child if not done so already
else if packet is not for me then
  if no route then
    add parent
    waitNextFrame=true
  end if
else if packet is broadcast control packet from parent then
  broadcast control packet
else if hear all 1's REB from parent AND I have no packet to send then
  broadcast control packet
else if hear all 1's REB from child AND I have no packet to send then
  send control packet to parent
else if hear all 1's REB AND child has no child AND child sent last packet then
  prune child
end if
if I have no children AND I have no packet to send then
  go to sleep
end if
```

Fig. 7. TreeDMA Pseudocode (part 2).

Type	SourceAddr	DestAddr	REB	ACK Bitmap	LastPkt Bit	HasChildren Bit

Fig. 8. TreeDMA header format.

The information necessary to perform the tree-constructing functions is embedded in every transmitted packet using a special TreeDMA header (Fig. 8). One header field, called the Route Exists Bitmap (REB), contains information about whether nodes are currently part of the tree (node's bit=1) or are not (node's bit=0). Each node also maintains a local record of the REB by updating it with any new information in packet header REBs, using a bitwise OR operation. This information is important for nodes already in the tree to listen for nodes that have not yet joined the tree. If the REB is all 1's, then all nodes have joined the tree and have a route.

For reliable delivery, an Acknowledgment Bitmap is included in the packet header. After a child node sends a packet, it then checks the header of its parent's next transmission for the acknowledgment. (The function of the remaining packet header fields will become clear as the protocol is described.)

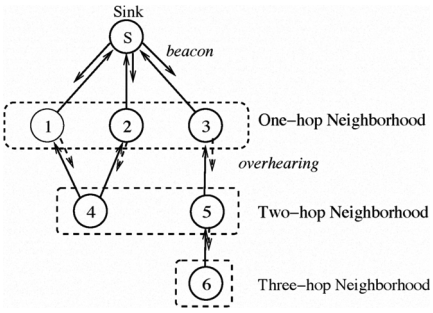


Fig. 9. Tree construction.

The process for constructing a tree is illustrated in Figs. 9 and 10 and is described as follows:

- In Frame 1 of the superframe, the sink node sends a

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858

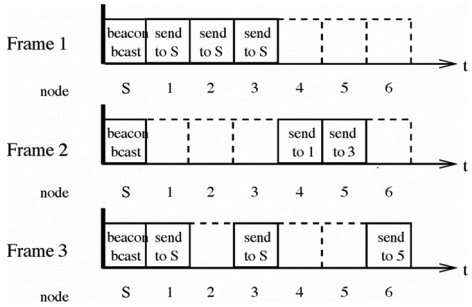


Fig. 10. First frames of tree construction.

beacon announcing its location. Those nodes that hear the beacon (e.g., nodes 1,2,3) learn that they are one hop away from the sink node and can immediately send in their slot. The sink node listens and receives from its neighbors, which it registers as its children in the tree. Those that did not receive the beacon (e.g., nodes 4,5,6) listen in the other nodes' slots to try to overhear a transmission and learn of a route to the sink. If a route is found, the node must wait until the next frame to send. This is because the one-hop neighborhood nodes do not listen in the first frame, since we do not typically expect multi-hopping.

- In Frame 2, the sink node sends a beacon containing the REB telling the receiving nodes (e.g., nodes 1,2,3) which nodes have a route. Nodes in the one-hop neighborhood listen in the slots of nodes that have not yet found a route (e.g., nodes 4,5,6). A node that has overheard the one-hop neighborhood in the first frame sends to the overheard node, which becomes the overhearer's parent (e.g., 4→1, 5→3). If a node hears more than one parent, it picks only one. Now the two-hop neighborhood has joined the network tree in this frame. Also, other nodes can possibly overhear the transmissions and join in the following frame. Each node listens in its parent's slot and its children's slots.
- For each successive frame, the sink continues sending a beacon, and nodes continue to forward data toward the sink. The most recently added nodes listen in the slots of nodes that have not yet found a route (e.g., node 6) so that they may join the tree (e.g., 6→5). This process continues until there are no more nodes to add to the tree.

2) Sleep Check/Tree Pruning: As soon as nodes finish sending data and no longer need to act as forwarding nodes, they should go to sleep to reduce idle listening. The following conditions must be met for a node to sleep:

- The node has no more data to send.
- The node has no active children.
- All nodes have found a route (all 1's REB).

Nodes with active children cannot go to sleep, in case they have to act as forwarding nodes. In order to go to sleep, they must first prune all of their children, declaring them inactive. If a node's child satisfies both of the following conditions, it can be pruned:

- The child itself has no active children (indicated in a Has Children field in the packet header).
- The child has no more data to send (indicated in a Last Packet field in the packet header).

Through the process of tree pruning and sleep checking, the entire tree will eventually be pruned and go to sleep for the remainder of the superframe.

3) Control Packets: In some cases, nodes may not have data to send, and control packets must be sent in their place to ensure proper operation of the protocol. Control packets consist of just the TreeDMA header so that tree information can still be disseminated. There are two cases for which nodes must generate and send control packets:

- A node does not have data to send during a particular superframe, but the network still needs to know that it has a route.
- A node overhears from its parent that all nodes have found a route (all 1's REB) and needs to inform its descendant nodes farther down the tree so that they can be put to sleep and pruned.

For every superframe, the entire protocol is run from scratch since we assume significant node mobility.

IV. Simulation

To evaluate the performance of the protocol, we implemented TreeDMA in the network simulator ns-2. At each node, one packet containing 64 bytes of data is generated every superframe. Since we are dealing with small networks, we assume the time slot assignment is done a priori and does not present an online overhead. Time synchronization is done out of band, using GPS.

We compare TreeDMA with two other protocols. The first is an "ideal" omniscient routing and TDMA MAC protocol. Here, nodes have omniscient knowledge of routes and coordinate sending and listening schedules without any actual communication. The result is a theoretically ideal protocol with no unnecessary sending or listening and with near-minimum latency.[2]

The second is a layered protocol utilizing a lean version of one-phase pull directed diffusion [5] in conjunction with a fixed TDMA. Directed diffusion is a data-centric routing framework that is distributed and scalable. In this version, we trimmed down the protocol to expect only one type of data. The sink initiates the protocol by flooding the network with an interest, which expires at the end of each superframe. Nodes set up gradients according to which neighbor it received from first, and data packets are sent along those gradients. The fixed TDMA assigns sending slots for nodes to send in turn, and non-sending nodes always listen.

[2]This latency is minimum for TDMA with no spatial reuse.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858

A. Radio Modes

The power consumption for a radio varies depending on what state the radio is in. Typically, we identify four different modes. A radio that is not turned on is in *sleep* mode, with the corresponding power consumption P_{sleep}. If the radio is turned on and is transmitting a packet, it is in *transmit* mode (P_{tx}). If the radio is on and is receiving a packet, it is in *receive* mode (P_{rx}). Lastly, if the radio is on but not transmitting or receiving, it is in *idle* mode (P_{idle}). We note that when the radio is sleeping, it consumes virtually no power, so we set $P_{sleep} \approx 0$. We also note that for sensor networks and WLAN technologies, power consumption is very similar for *tx*, *rx*, and *idle*, so we set $P_{tx} \approx P_{rx} \approx P_{idle} = P$.

B. MAC States

We define states at the MAC level to capture how the MAC uses the radio for different cases. Later we will explain how the radio is used, but now we first introduce the states. The *sending* state (*snd*) is when the MAC is sending a packet. In the *receiving* state (*rcv*), the MAC is receiving a packet that is addressed to it, including broadcast. In the *overhearing* state (ovh), the MAC is receiving a packet that is not addressed to it. The *sampling* state (*smp*) is when the MAC is receiving a packet that is undecipherable because it is unable to correctly receive the physical layer (PHY) header (e.g., reception is below the capture threshold). In the *listening* state (*lsn*), the MAC expects to receive a potential packet, but there is no packet to be received. Finally, the *off* state (*off*) is when the MAC does not expect data and can turn off the radio.

We now discuss the ways in which the MAC can use the radio, and how this relates to the power consumption. In the *sending* state, the radio is in *tx* mode for the duration of the packet ($P_{snd} = P_{tx} = P$). For a node that is in the *receiving* state, the radio is in *rx* mode for the duration of the packet ($P_{rcv} = P_{rx} = P$). Nodes that *overhear* packets not addressed to them could similarly receive the entire packet. However, the node really only needs to receive the MAC header, and the radio can actually go to *sleep* for the payload. Therefore, in our evaluation, we set $P_{ovh} \approx \frac{\text{header}}{\text{packet size}} P$. For the *sampling* state, if we do nothing special, the radio would be *idle* the entire time. However, since we have a slotted scheme, a node could listen at the beginning of a slot, realize that a packet is undecipherable, and put the radio to *sleep*. Likewise, for *listening*, a node could realize nothing is there (i.e., no PHY header) early on in the slot. In our evaluation, we assume that for these two states, *sampling* and *listening*, the node will sleep for virtually the entire slot, so we set $P_{smp} = P_{lsn} \approx 0$. Finally, when the MAC is in the *off* state, the radio is put in *sleep* mode: $P_{off} = P_{sleep} \approx 0$.

C. Energy Metric

The metric used to evaluate energy consumption is

$$\text{energy} = (P_{snd}t_{snd} + P_{rcv}t_{rcv} + P_{ovh}t_{ovh} + P_{smp}t_{smp} + P_{lsn}t_{lsn} + P_{off}t_{off})\frac{1}{P_{snd}t_{pkt}} \quad (1)$$

which is the total energy consumed in all MAC states, normalized by the energy needed to send 64 bytes, the payload portion of a data packet. After substituting the values determined in the previous section, the energy expression reduces to

$$\text{energy} = \frac{t_{snd} + t_{rcv} + \left(\frac{\text{header}}{\text{packet size}}\right) t_{ovh}}{t_{pkt}} \quad (2)$$

Even if listening energy was a concern, TreeDMA is designed to handle it by sleeping whenever possible. We will evaluate this statement and include listening data in our simulation results.

D. Simulation Results

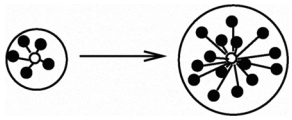
1) Simulation 1, Star Network: In the first simulation, we set every node to have a transmission range of 100 meters. The network density is fixed (0.00204 $\frac{\text{nodes}}{\text{m}^2}$), and we vary the network size by choosing the area of the network around the sink (Fig. 11(a)). Here we vary the radius of the area between 28 meters and 100 meters, which translates to varying the network size between 5 nodes and 64 nodes. Note that for all cases, nodes will be in communication range of the sink. Therefore, the first simulation illustrates the performance for a star network, which is the most typical case for these vehicle applications.

Fig. 11(b) plots the latency, which for all protocols is close to zero because it is simply the time needed to send a packet over one hop. Fig. 11(c) plots the energy required per node to send a data packet. The energy for *TreeDMA* is close to *Ideal* for all network sizes, whereas the energy required for *Layered* suffers from flooding the interest in a dense network. In terms of the energy averaged over all network sizes, *TreeDMA* consumes 10% of the energy of *Layered*, and it consumes only 65% more energy than *Ideal* because nodes receive the beacon (which is needed to provide routing when necessary). For the star network, we see that *TreeDMA* performs favorably.

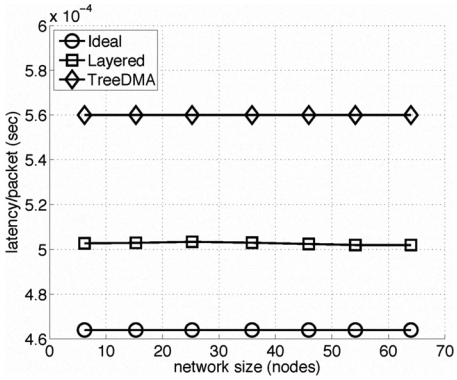
2) Simulation 2, Effect of Multi-Hopping: In the second simultion, we again fix the transmission range to be 100 meters, but we fix a lower network density (0.00112 $\frac{\text{nodes}}{\text{m}^2}$) than in the first simulation. We again vary the area of the network to yield different network sizes (Fig. 12(a)). As the network area gets bigger, nodes are unable to communicate with the sink directly, and we observe the performance when multi-hop routes are present. Fig. 13 plots the average and maximum number of hops to the sink for different network sizes for the previous and current simulations. We observe that for this simulation, the routes are one-hop for network sizes up to 35 nodes, after which some of the routes are multi-hop. This allows us to observe the change in performance as the network transitions from one-hop to multi-hop.

Fig. 12(b) plots the latency, where again we see that when routes are one-hop (network size < 35 nodes), the latency for all protocols is close to zero. For network sizes greater than 35 nodes, multi-hop routes are introduced and latency starts to increase for *Layered* and *TreeDMA* due to the route setup. In

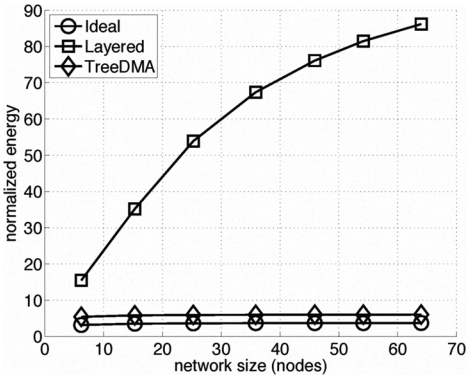
Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858



(a) Star Network.



(b) Latency per packet.

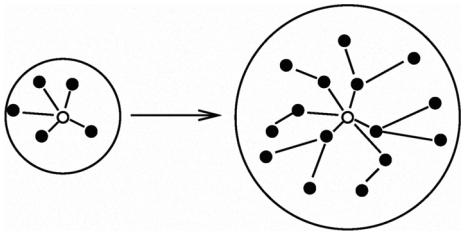


(c) Energy per node per packet.

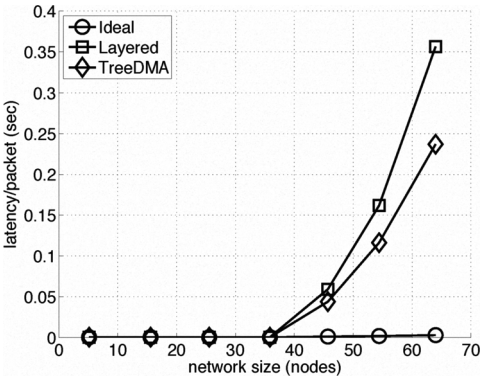
Fig. 11. Simulation 1, Star Network.

Fig. 12(c), we note that the energy required per node to send a data packet for *TreeDMA* stays close to *Ideal* when average hops is 1, but when multi-hop routes are introduced, more energy is required due to overhead from beaconing and control packets. For this particular node density, *TreeDMA* consumes on average 20% of the energy of *Layered* and consumes 132% more energy than *Ideal*. We again observe that *TreeDMA* performs favorably when nodes are one hop from the sink. As multi-hop routes are introduced, more energy is consumed because more overhead is needed to route, but *TreeDMA* is still able to handle such scenarios.

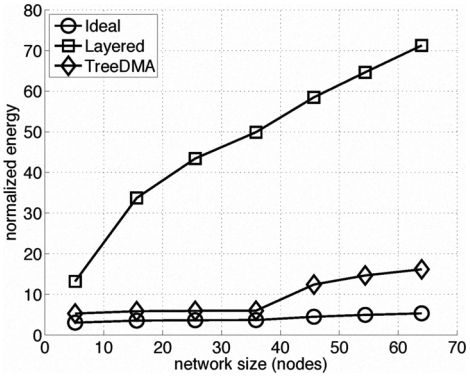
3) Listening Energy: Listening was defined as the state where the MAC expects to receive a potential packet, but there is actually no packet to be received. Although the energy metric defines listening power consumption as $P_{ls} \approx 0$, some radios may not be able to go to sleep simply based on the physical layer information. Therefore, *TreeDMA* is still designed to reduce listening. Fig. 14 plots the listening energy consumption for the first two simulations, for $P_{ls} = P$. When the routes are single-hop, we see that there is no listening energy consumed, but when multi-hop routes are needed, more



(a) Multi-Hop Network.



(b) Latency per packet.



(c) Energy per node per packet.

Fig. 12. Simulation 2, Effect of Multi-Hopping.

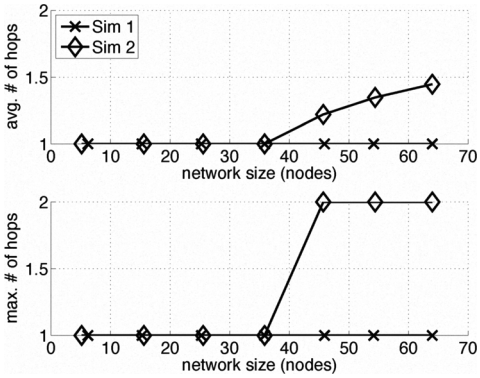


Fig. 13. Average and maximum number of hops.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858

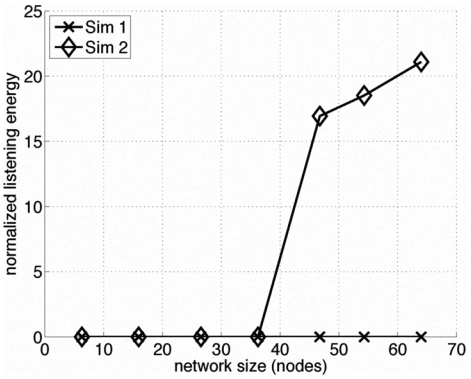


Fig. 14. Listening energy per node per packet.

listening occurs during tree construction.

V. Related Work

Much protocol research in sensor networks has been based on the paradigm that networks consist of large numbers of energy-constrained nodes. Prominent examples are described in the survey article by Demirkol [6].

There has been some research in studying scheduling for convergecast networks to minimize latency ([7],[8]). These approaches utilize spatial reuse, in which nodes can be assigned the same slot if the transmissions can be received successfully. Since nodes will mostly be sending directly to the sink for vehicle sensor networks, there is little gain from spatial reuse. MAC protocols such as DMAC [9] and TDMA-EC [10] propose heuristic solutions to reduce latency by scheduling send times based on a node's position in the tree.

One data-gathering protocol that includes tree construction and scheduling is Dozer [11]. Dozer performs distributed tree construction using periodic beacons from all connected nodes in the tree, with parent-child relationships established by a handshake. The TDMA schedule is also exchanged using the beacon message, and separate parent and child TDMA schedules are kept, eliminating the need for global synchronization. We argue that for the small-scale networks of mobile vehicles, having a distributed architecture is unnecessary and introduces extra overhead.

Overhearing has been utilized in ad hoc networking as a way of reducing extra transmissions. In [12], Kim includes a reservation period in which overhearing of RTS and data packets were used in lieu of CTS and ACK packets. ISO-MAC [13] includes extra fields in the packet header for slot availability information. These are distributed algorithms that do not specifically address convergecast traffic, and their benefit is most enhanced in multi-hopping scenarios.

VI. Conclusion

In our networking approach, we have exploited our knowledge of the nature of small-scale mobile vehicle-based networks, for which multi-hop routes are uncommon. Fig. 2 showed that for a small network size, the transmission range for a connected network is very close to the range for a star network. We argued that in practice, most small networks will usually be star networks. Our cross-layered approach is optimized to handle the one-hop case most efficiently, while still retaining the ability to multi-hop route when necessary. TreeDMA has been shown to perform comparably with an "ideal" protocol for the typical one-hop scenario, yet it is also capable of handling multiple hops. From an energy standpoint, it outperforms a layered protocol, one-phase pull diffusion/TDMA, by as much as a factor of 10 for common small-scale vehicle networks.

References

[1] S. Poduri and G. S. Sukhatme, "Constrained coverage for mobile sensor networks," *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, vol. 1, pp. 165–171, April 2004.

[2] D. O. Popa, A. C. Sanderson, R. J. Komerska, S. S. Mupparapu, D. R. Blidberg, and S. G. Chappel, "Adaptive sampling algorithms for multiple autonomous underwater vehicles," *2004 IEEE/OES Autonomous Underwater Vehicles*, pp. 108–118, 2004.

[3] A. Howard, M. J. Mataric, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," *Proceedings of the 6th International Symposium on Distributed Autonomous Robotics Systems*, June 2002.

[4] J. Elston, E. W. Frew, and B. Argrow, "Networked UAV communication, command, and control," *AIAA Guidance, Navigation, and Control Conference*, August 2006.

[5] F. Silva, J. Heidemann, R. Govindan, and D. Estrin, "Directed diffusion," USC/Information Sciences Institute, Los Angeles, CA, Tech. Rep. ISI-TR-2004-586, January 2004.

[6] I. Demirkol, C. Ersoy, and F. Alagoz, "MAC protocols for wireless sensor networks: A survey," *IEEE Communications Magazine*, pp. 115–121, April 2006.

[7] H.-W. Tsai and T.-S. Chen, "Minimal time and conflict-free schedule for convergecast in wireless networks," *IEEE International Conference on Communications*, May 2008.

[8] V. Annamalai, S. K. S. Gupta, and L. Schwiebert, "On tree-based convergecasting in wireless sensor networks," *IEEE Wireless Communications and Networking*, March 2003.

[9] G. Lu, B. Krishnamachari, and C. S. Raghavendra, "An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks," *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, p. 224, April 2004.

[10] B. Ren, J. Xiao, J. Ma, and S. Cheng, *Mobile Ad-Hoc and Sensor Networks*. Springer Berlin / Heidelberg, 2005, vol. 3794/2005, ch. An Energy-Conserving and Collision-Free MAC Protocol Based on TDMA for Wireless Sensor Networks, pp. 603–612.

[11] N. Burri, P. von Rickenbach, and R. Wattenhofer, "Dozer: Ultra-low power data gathering in sensor networks," *Proceedings of the 6th International Conference on Information Processing in Sensor Networks*, pp. 450 – 459, April 2007.

[12] J. Kim, K. Park, J.-H. Shin, and D. Park, "Look-ahead scheduling for energy-efficiency and low-latency in wireless sensor networks," *Proceedings of the 3rd ACM International Workshop on Performance Evaluation of Wireless Ad Hoc, Sensor and Ubiquitous Networks*, pp. 141 – 144, October 2006.

[13] S. Biswas and F. Yu, "An in-band self-organized MAC protocol for sensor networks," *Proceedings of SPIE*, vol. 6248, April 2006.

Digital Object Identifier: 10.4108/ICST.MOBIQUITOUS2009.6858
http://dx.doi.org/10.4108/ICST.MOBIQUITOUS2009.6858