Content based multicast (CBM) in ad hoc networks

Hu Zhou and Suresh Singh
Department of ECE
Oregon State University
Corvallis, OR 97331
Email: singh@ece.orst.edu

Abstract— **This paper presents a radically new multicast model for ad hoc wireless networks. In this model, the *content* of the multicast data determines the receiver set for the data and the receiver set changes dynamically as the content of the multicast changes. Delivering the multicast to intended receivers while minimizing the message overhead and reducing battery consumption is a challenge made harder because of node mobility. In this paper we develop a novel multicast model that distributes this form of threat information in a message efficient manner. We present results from detailed simulations that demonstrate the efficiency of our protocol and discus the scalability of this model to larger networks.**

I. Introduction

Ad Hoc networks are multi-hop wireless networks consisting of 1000's of radio-equipped nodes that may be as simple as autonomous (mobile or stationary) sensors to laptops mounted on vehicles or carried by people. These types of networks are useful in any situation where temporary network connectivity is needed, such as in disaster relief or in the battlefield. In this paper we present a novel *multicast model* relevant to these types of networks. This model is distinct from existing multicast models in that **the receiver set for information changes dynamically based on the content of the information being multicast as well as on the mobility of the receivers themselves.**

In order to explain our multicast model better, let us focus on the two obvious application areas. The first application is in disaster relief networks set up in areas devastated by natural or manmade causes. Imagine the effect of a severe earthquake in Southern California. It is likely that the affected area will contain a variety of threats to disaster relief personnel such as gas leaks, intense fires, toxic leaks, riots, etc. Disaster relief personnel sent into these areas will need to be kept appraised of the location and types of existing threats to ensure their safety. They will also need to be kept informed of the deployment of other relief personnel, equipment and resources (such as, for example, *nodes that distribute public keys*, or *DHCP servers* that enable new nodes to join the network, etc.) in the area who they can call upon in an emergency or as needed. The second application is in battlefield networks where soldiers in the field need to be constantly informed of impending threats and information such as time-to-threat, distance-to-threat and type-of-threat must be presented to them in a timely manner. Similarly, information regarding the location and movement of nearby allies also needs to be presented to the soldiers.

A person in the field is typically most interested in obtaining information that will provide her with a complete picture of her "immediate surroundings". In other words, people in the field need to know about immediate threats (within, say, a ten minute time horizon) rather than about more distant threats (that may be over an hour away). Similarly, they may need to know about *all* threats within a radius of one kilometer (this information is useful, for example, in planning an escape route in the event that a fire gets out of control). Likewise, a person in the field may need to know the location of other disaster relief people within 100 meters or about the availability of specialized equipment within 1km. In this case, the location and velocity of each disaster relief person or resource in the neighborhood constitutes the data of interest.

In both scenarios, threat information can be collected by autonomous sensors dropped into the affected area. Likewise, information regarding the movement, capability and location of other disaster relief personnel and equipment (or allies, in the military model) is also easily available. It is then the task of the network protocols to

deliver this information in a timely manner to personnel in the field *who really need the information* (in addition to planners, such as FEMA). The multicast model we discuss here is concerned with *efficiently distributing this information from the sensors and other sources to the personnel in the field on an as needed basis.*

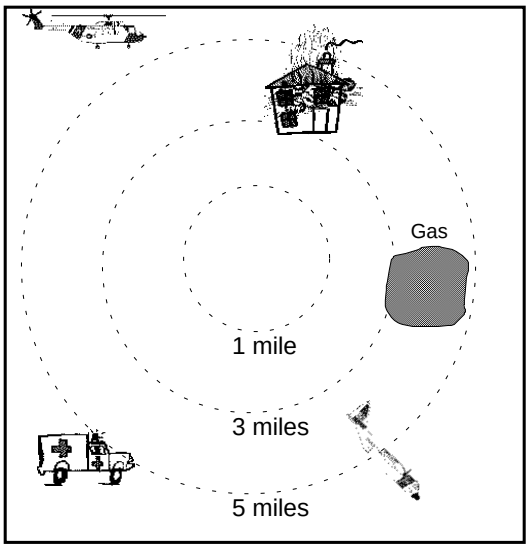
In the *Content-Based Multicast* model (also called CBM in this paper), we assume that nodes are interested in obtaining information about *resources* and *threats* that are:

- t time away from their current location (assuming a non-zero relative velocity), and/or
- that are d distance away.

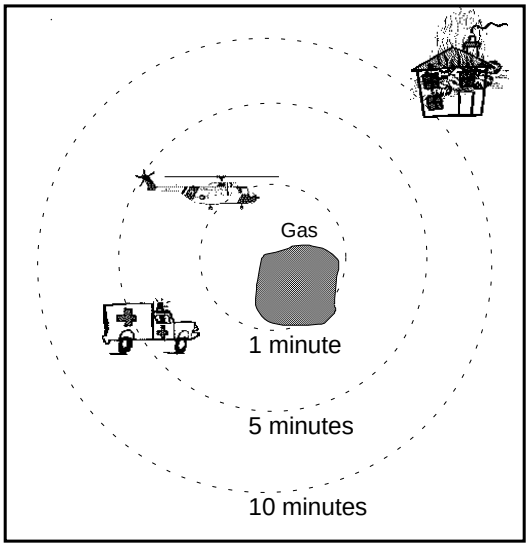
Sensors and other nodes generate information about the movement, intensity and, location of threats and resources. They then need to get this information to those nodes that need it (based on each node's individual t and d specifications). However, this needs to be done in a way that *minimizes message cost (and hence energy consumed)*. The problem is, since nodes, threats and resources move, senders have no way of identifying their receiver set and similarly, receivers have no way of knowing which sender's information is of interest to them! How then can this information be disseminated? In the remainder of this section we discuss the challenges in conducting the multicast in a decentralized way that can incorporate the t/d specifications of all receivers.

A. Example of CBM Multicast

In order to describe the problem of CBM multicast, it is helpful to consider a simple example. Figure 1 illustrates two views of the neighborhood that we would like to present to a disaster relief person. The view on the left indicates the distance to different types of threats (the gas cloud), problems (the burning house) and resources (the ambulance and the helicopter to ferry out endangered personnel) and the view on the right indicates the time to these threats and resources. Thus, the gas cloud is three miles away but, due to the wind direction and speed, it is less than one minute from the person in the field. Likewise, the helicopter is over five miles away but, because of its speed, it will be in the neighborhood in between one and five minutes. On the other hand, there is an aeroplane within three miles, but because of its direction of travel, it will be far away soon (and thus unable to forward messages from the ground personnel to FEMA, say). Thus, we see that information



(a) Distance to threats and resources



(b) Time to threats and resources

Fig. 1. Two views of the neighborhood.

most relevant to a person in the field includes,

- the location of all resources, threats and problems that are within t seconds of the person's current position, and
- the location of all resources, threats and problems that are no more than d meters away.

It is noteworthy that different relief personnel will dynamically change the t and d specifications depending on field conditions and, furthermore, different personnel will typically have different t and d specifications. Thus, delivering information to relief personnel in a way

that satisfies everyones t and d specifications is a non-trivial problem.

B. Effect of Mobility on the Relevance of Information

For the sake of clarity, let us initially assume that the t and d specifications for all personnel is the same. Consider Figure 2 where a sensor detects a gas cloud in the area and determines that the wind is blowing from the east. In this case, the sensor needs to multicast this information in the westerly direction only. If the wind changes direction and begins blowing from the south west, however, the sensor will need to multicast its information to receivers located in a north easterly direction. Thus, the *direction* of movement the threat influences the conduct of the multicast. A second consideration here is *how far* should this information be propagated in the network? In Figure 2, everyone in region A needs to be warned (if we assume $t = 10$ minutes) but region B is not in immediate (i.e., ten minutes) danger so there is no need to extend the multicast to nodes here. If we now relax the assumption that the t and d specifications are identical for all personnel, we see that the question of how far the multicast needs to be extended becomes complicated. This is because there may be nodes in region B with large t values (these may be people on foot) who will need to be included in the multicast and there may be other nodes in region A with small t values (people in automobiles) who need not receive the multicast.

Let us consider another situation, illustrated in Figure 2, where an automobile is travelling towards the gas cloud at speed. Here it is necessary to ensure that the multicast reaches the car because it will be in danger within 10 minutes. Unfortunately, however, the sensor does not know about the existence of the car and cannot include it in its multicast. Thus, there is a need to extend the multicast from the sensors so that *mobile personnel*, such as the car, also receive the information even though they are initially far away.

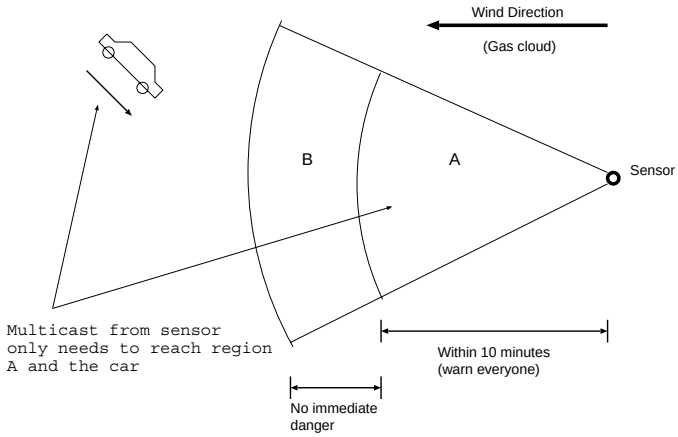


Fig. 2. Multicast coverage of information generated by a sensor.

C. Effect of the Nature of the Threat/Resource on the Multicast

The *type* of threat and its *reach* are also relevant in determining the conduct of the multicast. For example, a disaster relief team equipped with gas masks would not be concerned with gas leaks while people with no gas masks need to know about the progress of the cloud to get out of the way. Similarly, in a battlefield, a tank is not necessarily threatened by soldiers on foot while it would be concerned if there were any enemy tanks in the area. In other words, the type of threat/resource is a factor in determining the receiver set for a multicast. The second feature of a threat is its reach. For instance, a fire can be threatening up to distances as great as one hundred meters away (because of the heat and the possibility of freak explosions that spout flames). Similarly, in a battlefield, an aeroplane poses a threat tens of miles away because of the type of weapons it carries. Thus, a receiver only needs to obtain information about those threats that *do indeed pose a danger* within its t and d specifications or about *useful* resources it will have access to within t time or that are d distance away.

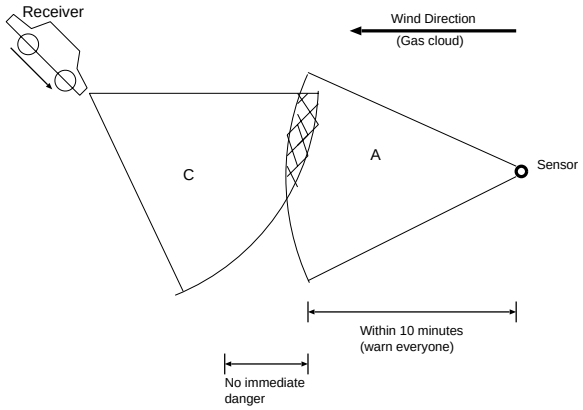


Fig. 3. Multicast data collected by a receiver via receiver-pull.

D. Our Approach

Based on the above discussion we can summarize the problem as follows. A person needs to have information about threats/resources that are within time t seconds or within distance d. This information is affected by two factors:

- The *relative velocity* of each threat/resource w.r.t. the node and,
- The reach of the threat.

The information regarding threats/resources is collected by sensors and then needs to be multicast to all those who need it (based on t and d specifications). This problem of multicast is made difficult because:

- The sensors do not know the composition of their multicast groups since group membership is based on the t and d specifications of individual receivers.
- Changes in the resource/threat deployment changes the t and s specifications.

Our approach for developing a solution to the CBM problem discussed above is to use a *sensor-push receiver-pull* approach. Here, sensors push the information out into the network to some distance and receivers then pull relevant information from the network satisfying their time-space mandates. In Figure 3, for instance, the sensor pushes out the information about the gas cloud into region A. The car sends a request for information *ahead* of it to nodes within distance vt meters (the node's speed is v meters/sec). In Figure 3 this is represented by region C. All multicast information available to nodes in region C is forwarded to the car. Thus, if there is any node in the intersection of regions A and C, information about the gas cloud will get forwarded to the car. We will expand this idea in section III.

E. Overview of the paper

In section II we describe other multicast protocols for ad hoc networks and describe how our time-space model differs from them. Section III describes our multicast protocol in detail. We present results of simulations in section IV. The work reported in this paper is ongoing and we describe our current research focus in section V.

II. Related Work

Recently several authors have begun developing multicast protocols for ad hoc networks. The primary challenge they have attempted to solve has been to construct and maintain multicast *trees* or *flows* (see [6]) despite topology changes. Some examples of these protocols include:

1. *AMRoute (Adhoc Multicast Routing)* [1]. In this protocol, a shared multicast tree is built for multicast group members to forward packets. The tree construction is completed in two steps: mesh construction and tree construction. In this model, the receivers know the group they want to join, thus the identity of the senders. It is the senders' responsibility to start building the multicast tree. The senders do not know about the receivers' identity.
2. *ODMRP (On-Demand Multicast Routing Protocol)* [7], [8]. This is an on-demand protocol, in which a mesh structure is build to flood packets from the source node to the group members. Like the AMRoute, receivers know about the senders' identity.
3. *AMRIS (Ad hoc Multicast Routing protocol utilizing Increasing id-numberS)* [13], which again, uses a shared tree to forward multicast packets. The tree is built through the broadcasting of NEW-SESSION messages, and is maintained by group members with the help of a beacon mechanism. A group member can dynamically join a multicast session in the multicast group. Similarly in this model, the receivers know about the senders' identity, but not vice versa. The creation of shared trees is initiated from one special node, and then the tree is maintained by all group members.
4. *CAMP (Core-Assisted Mesh Protocol)* [5], [9] which, like the ODMRP, builds a mesh to forward multicast packets. Some core nodes are responsible for accepting initialization requests. Nodes can dynamically join the multicast group and help maintain the multicast mesh. Group members use a packet cache to monitor their connectivity and rebuild connections. Also, in this model, receivers know the senders' identity but not vise versa.

It is easy to see that the CBM model proposed in this paper is very different from any of the above models. This is because the *multicast receivers* of a data stream are determined based on the *data contents* (recall that a receiver only wants to receive data that informs it of impending threats defined in time or space). As the data content from a sender changes, so does the multicast group! This feature of our model makes it unique as well as extremely powerful in the military context where the goal is to maximize message efficiency while ensur-

ing that the *sensor-to-shooter distance* is minimized.

III. Description of the CBM Multicast Protocol

Our multicast protocol is based on the idea of *sensor-push* and *receiver-pull*. Essentially, sensors detecting threats send a limited broadcast message into a small region that lies in the path of the threat and individual receivers then pull threat warnings from nodes that lie in the direction of their travel. This push-pull strategy works efficiently because we reduce the number of unnecessary threat warning messages by ensuring that the only nodes that need to receive the warning receive them.

For an efficient implementation of the push-pull approach, we view the playground as being divided into geographic regions as shown in Figure 4. These regions are virtual and are only used for improving the message efficiency of our protocol. It is important to note that the regions do not need to be of the same shape or even size. The specific form of regions will be based on the density of nodes within the region, terrain characteristics, and node mobility constraints. We assume that all nodes are GPS equipped.

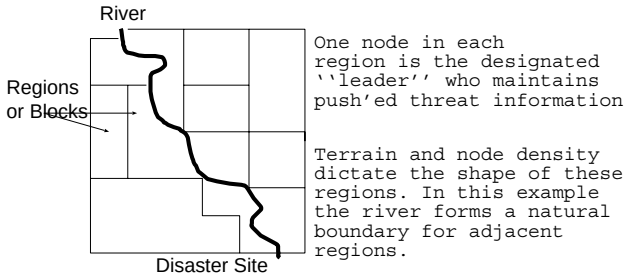


Fig. 4. The battlefield is viewed as being made up of regions.

Within each block one node is chosen to be the "leader". This node maintains a list of all threat warnings and resource updates received via push messages and is responsible for responding to pull requests as well. We will discuss these two parts of the protocol in sections III-A and III-B. When a leader leaves a block, however, the responsibility for maintaining threat warnings relevant to that old block passes on to a new leader. We describe the leader maintenance portion of the protocol in section III-C. Finally, we note that that *routing protocol* used here is MFR (Most Forward with Fixed Radius) described in [11].

A. Push Protocol

Warning messages regarding threats are generated by sensors as and when a new threat is detected or the disposition of a known threat changes (e.g., it changes direction). Thus, when a sensor or a collection of sensors detects the presence of a threat, the sensor generates a limited broadcast message for nodes that will lie in the *projected path of the threat*. Consider Figure 5 where, for the sake of clarity, we assume that the blocks are all identical rectangles. When a tank threat is detected, the sensor(s) that detected the threat need to inform nodes that lie in the path of the tank of a possibly imminent attack by the tank. In the figure we indicate that the extent of this warning push message extends out to blocks that lie within distance $s(\tau_k + \Delta\tau)$. Here s is the speed of the tank, τ_k is a constant and denotes the *time* specification that is used by tank sensors to determine the extent of the push. In our simulations we use a value of $\tau_k = 300$ seconds and the interpretation is that the push informs nodes that are no more than 5 minutes away from encountering the tank threat. The $\Delta\tau$ factor is an error factor (and is also a constant) that ensures that more rather than fewer nodes get informed of the threat.

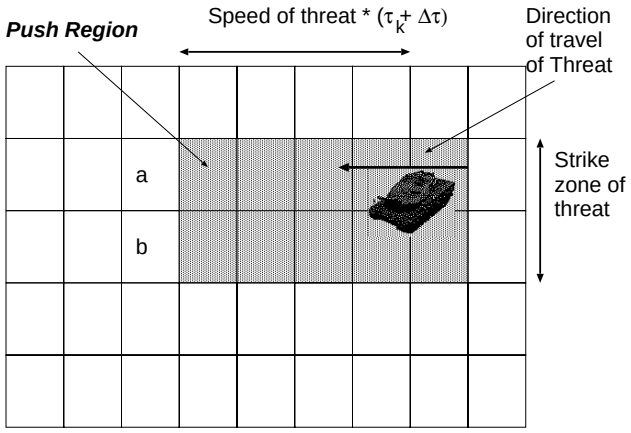


Fig. 5. A *push* is generated ahead of the tank.

We can now state the push algorithm as follows:

Algorithm : *Push*

- When a sensor detects a threat it determines the projected velocity vector for the threat k. Let s denote the speed of the threat.
- Let τ_k represent the time specification used by the sensor to push information for the threat. The value τ_k might be different for different types of threats.
- The sensor (or sensors) send

one THREAT_WARNING message to each of the leaders of the blocks (the routing protocol used is MFR [11]) that lie within the rectangular area with length $s(\tau_k + \Delta\tau)$ and width equal to the *reach* of the threat. The rectangle is oriented in the direction of motion of the threat.

- Each leader receiving the THREAT_WARNING message broadcasts it within their groups. The THREAT_WARNING message includes the nature of the threat, velocity, and any other information (e.g., confidence level of the sensor in projecting the threat's motion).
- Whenever the threat changes direction or moves a distance such that the furthest block warned is less than $s\tau_k$ distance away from the threat, a new push message is sent to blocks that are within $s(\tau_k+\Delta\tau)$ distance of the threat. In Figure 5, after the fire spreads through one (or two) blocks in the indicated direction, a new push is generated to cover blocks a and b.

B. Pull Protocol

The push protocol as described above succeeds in sending threat warnings to nodes that are within τ_k of the threat. However, nodes that are moving towards the threat (and are further away than $s(\tau_k + \Delta\tau)$ of the threat's position) or nodes that are far away but have a large *time/space* specification need to pull the threat information in order to be appropriately warned. For the sake of clarity we will use the example illustrated in Figure 6. Here, the node on the bottom left is moving at some speed s_n and in time t_n it will be in block A. Assume that threats located anywhere within the dotted box can pose danger to nodes located in block A. The algorithm we use for generating pulls is as follows:

Algorithm: *Pull*

- Let a node's *time* specification be t_n (the same algorithm works if we use a node's distance specification). In other words, the node needs to be warned of all threats it considers threatening that it could encounter within time t_n.
- Let the node's speed be s_n. Therefore in time t_n it will be located in a block that is $s_n t_n$ distance away (in Figure 6 this corresponds to block A).
- The node sends a PULL_REQUEST to the leader of the block it expects to be in after time t_n. In Figure 6 this corresponds to block A. The PULL_REQUEST contains the node's velocity vector as well a specification of threats the node needs to be warned about (for example the pull request may only indicate gas as potential threats).
- The leader from block A sends back all the information it has about threats that will lie within the dotted region of Figure 6 at time t_n hence. It is, however, likely that the leader may only have incomplete knowledge about threats that are t_n time away. In this case it needs to initiate additional pulls as follows:
  - Let s_k^{max} represent the maximum speed of threats of type k.
  - The leader sends a pull request to leaders of blocks that are located distance $s_k^{max} t_n$ away (as shown in Figure 6).
  - Leaders of these blocks determine if threats they are aware of will lie in the dotted area (Figure 6) t_n time hence. If so they respond with that information to the leader of block A. The leader of block A then forwards this information to the requesting node.
- Finally, all pull requests are **sticky** with a TTL field (Time To Live). This means that once a leader gets a pull request, it holds on to it until the TTL field expires. Meanwhile, if there is any new threat information, it forwards that to the node that generated the sticky pull request. This mechanism ensures that pull requests only need to be sent infrequently.

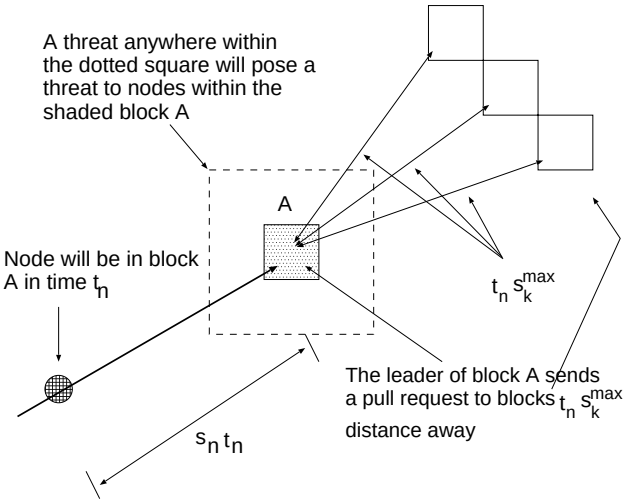


Fig. 6. A *pull* is generated ahead of the node.

C. Leader Maintenance

In order for our protocol to work, we need a leader maintenance protocol as well. This protocol does the following:

- A departing leader selects a new leader in the old block and passes on all the collected threat information to that leader. The identity of the new leader is also broadcast within that block.
- When any node leaves a block, it informs the leader of the old block as well as the leader of the new block.
- If the leader is the last node remaining in a block, when it leaves the block it transmits a message to leaders of all neighboring blocks informing them of all its gathered threat information as well as informing them that the block is now empty. Thus, any pull requests for the empty block will be replied to by one of the neighboring block's leaders (note that pull messages have to traverse a neighboring block to get to the empty block, thus the leader of the neighboring block can respond).
- When a node enters a empty block, it collects threat information from the neighboring blocks and declares itself the leader.

There is a special case that we do not consider in the current version of the protocol – the neighboring blocks may also be empty. We do not consider this case for now because the probability of this happening is very small. However, we will develop a solution for this case in the next version of the protocol.

IV. Performance Evaluation

We evaluated the performance of our CBM multicast protocol via extensive simulations. In order to quantify the performance of our protocol, we concentrated on two metrics:

1. *Message overhead:* The metric here is the number of messages/second/node that are exchanged in the course of the simulation run. This includes all push/pull messages as well as control messages exchanged in order to maintain the blocks.
2. *Coverage:* It is possible that some nodes do not receive threat warnings (or resource availability information) in time. Thus, we measured the percentage of nodes not receiving this information. Ideally we would like this number to be zero. However, periodic network partition and fluctuations of routes make it difficult to achieve this value.

For comparison, we ran the same set of experiments using a smart flooding algorithm. Here, each node transmits any new information it receives to each of its neighbors (except the one it received the information) exactly once. Clearly this protocol will maximize the coverage but will have a high overhead.

The remainder of this section is organized as follows. We first describe the experimental set up and then describe our experimental results.

A. Simulation Parameters

For the simulation we assume that the playground is a 50km by 50km square region. The scenario we simulate is a battlefield where threats consist of enemy tanks, soldiers and gas. The friendly nodes are either tanks or soldiers. Thus, the goal of the CBM protocol here is to deliver threat information quickly to the friendly tanks/soldiers. Tanks move at an average speed of 72kmph, soldiers move at an average speed of 9.6kmph and wind speed is constant at 18kmph. The battlefield has sensors capable of detecting tank and gas threats. In the simulations we use 64 gas sensors and 64 tank sensors evenly distributed throughout the batlefield. A gas sensor can detect a gas threat within 1km and a tank sensor can detect tank threats within 1km as well. Soldier threats can only be detected by other (friendly) soldiers. Thus, it is possible that a soldier threat may go undetected because of human error. We use a probability of 0.2 that a soldier threat will not be detected. Finally, we assume that tanks, soldiers and sensors have radio capability. The transmission radius of the sensors is assumed to be 2km, tanks have a transmission radius of 5km and a soldier can transmit to a distance of 1km. The data rate available is 1Mbps.

The time-space parameters we use were the following. Tanks needed to know about tank threats within 6km and soldiers needed to know about tank and soldier threats within 900m. Each simulation is run for 300 seconds of real-time (a couple hours of simulation time) and the simulation time step is 5 msec of real-time. We run each case ten times and compute 95% confidence values. In each case the, the confidence intervals were very tight (less than 5% of the point values).

B. Discussion of Results

In all of the following plots we use a constant number of enemy threats. Specifically, we use 10 tank, 10 gas and 30 soldier threats. In Figure 7 we plot the total mes-

sage overhead as a function of the number of friendly tanks (no soldiers) and in Figure 8 we plot the message overhead as a function of the number of soldiers (no tanks). First, it is noteworthy that smaller block sizes result in greater message overhead. This is because the number of responses to a pull are greater (one per block) and the overhead of maintaining blocks is also greater (since the time a node spends in a smaller block is smaller). The second interesting observation is that the message overhead increases sub-linearly with increasing numbers of friendly nodes. This indicates that our CBM protocol is scalable to large networks. Intutively this makes sense since the extent of a sensor's multicast is geographically limited to nodes in immediate danger. Thus, even if the network size grows, the message overhead ought to remain the same.

When we compare the cost of smart flooding with the cost of the CBM protocol we observe that CBM with a block size of 500m has a higher message overhead than flooding in the case when there are only 25 tanks (Figure 7). This is due to the high cost of leader maintenance in this case (frequent arrivals and departures from blocks have a message cost but the greater cost results from the fact that many blocks are empty and this causes the neighboring blocks to take over the responsibility of responding to pull requests for the empty blocks). Furthermore, as the number of tanks grows, the message cost comes down because the cost of leader maintenance increases relatively slowly (the playground size remains constant even though the number of tanks has increased) with an increase in the number of nodes. In the case when we only have soldiers (Figure 8), the flooding cost is much higher. This is because soldiers move at a slow speed and thus the leader maintenance cost is very small which results in a small overall CBM cost. The cost of flooding, on the other hand, increases with an increase in the number of nodes.

Figure 9 plots the success rate as a function of the number of tanks. It is noteworthy that with small block sizes the percentage of nodes warned in time is only about 80% for the case when there are 25 tanks. This number changes to 96% with a block size of 2.5km. Thus, using larger block sizes is good because the message cost is lower and the percentage of nodes warned is higher. Interestingly, the success rate does not appear to vary much with block size in the event that we only have soldiers (Figure 10). The explanation for this is that pull messages do not always reach the desired blocks

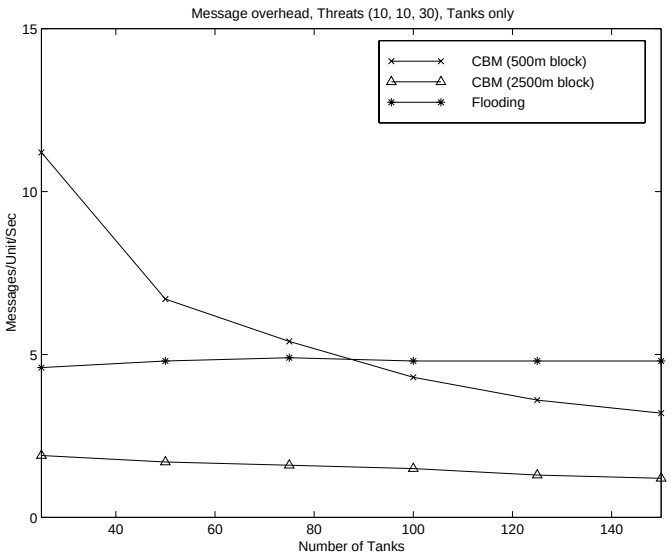


Fig. 7. Message overhead – Tanks only.

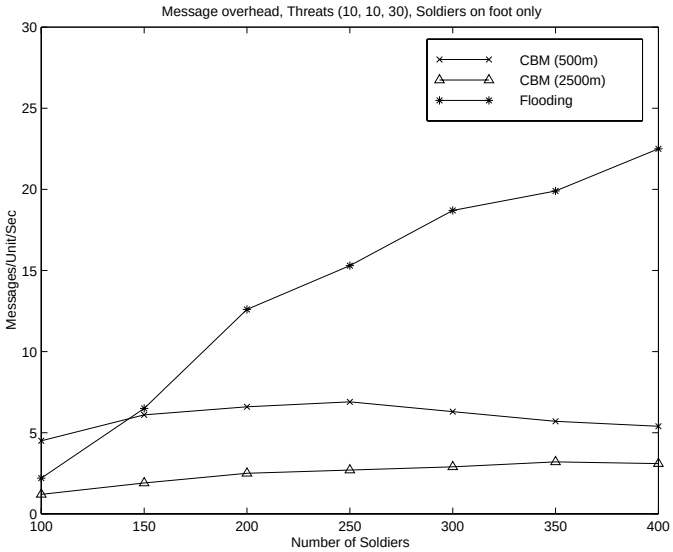


Fig. 8. Message overhead – Soldiers only.

because the network does not have a geographically direct path (we use geographical routing to send messages so if there is no next hop in the direction of the destination, the packet is dropped). However, as the number of soldiers increases, the probability of finding a path increases and hence the percentage of nodes warned in time also increases. This is also the reason that with fewer soldiers the success rate is so small (65%). In the case of tanks, their high speed and greater transmission radius ensures that there is a greater probability of finding routes.

Figures 11, 12 plot the relative contribution of pulls

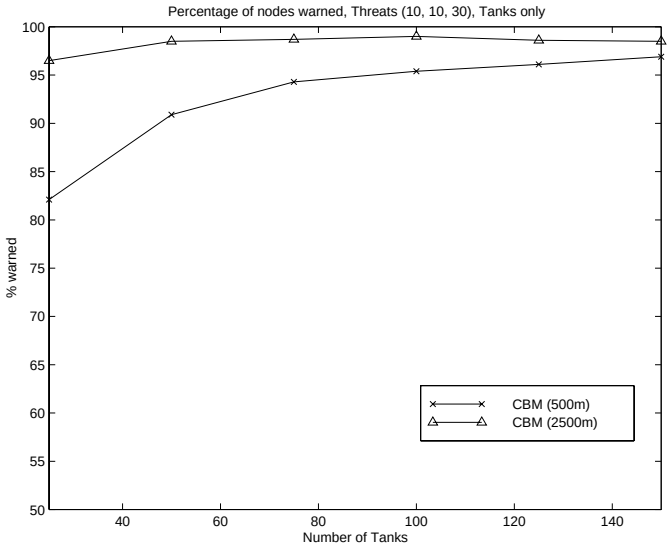


Fig. 9. Success Rate – Tanks only.

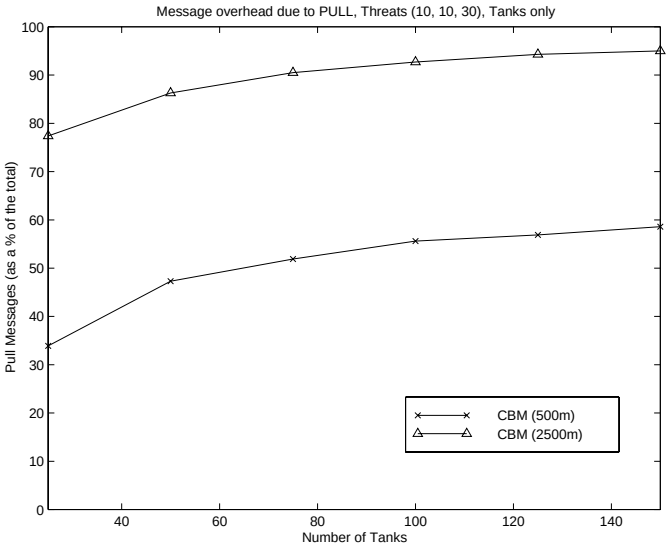


Fig. 11. Pull contribution – Tanks only.

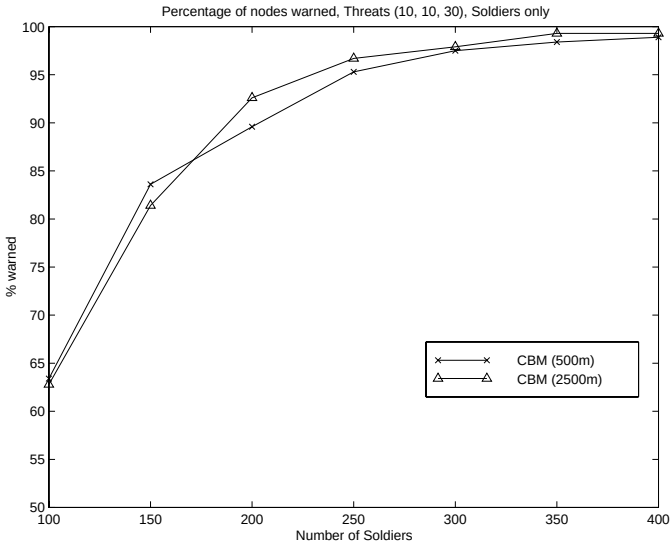


Fig. 10. Success Rate – Soldiers only.

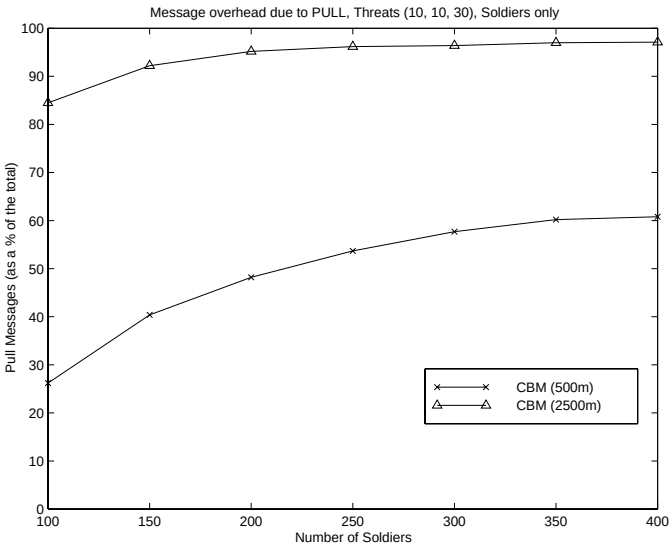


Fig. 12. Pull contribution – Shooters only.

to the overall cost of CBM. As we can see, increasing numbers of nodes result in a greater number of pulls.

V. Future Work

The work reported in this paper is ongoing and we are making the following enhancements to our simulation to better understand the CBM model.

- We are extending our simulation to take into consideration the path loss models of the terrain as this affects connectivity and hence the success rates.
- Threat mobility is unpredictable. Thus we are including a parameter in our model that denotes the sensor's *confidence* level in predicting the threats future velocity.
- Thus far our protocol only deals with threats. In the next iteration we will implement the same push-pull idea to allow nodes to collect information about allies and resources. Unlike the threat case, allies and resources can be queried directly for information about their location.
- In the case of gas threats, the gas cloud disperses over time. This means that the cloud expands its reach and later becomes benign. We will include diffusion models for gas into our simulation to have

a more realistic understanding of how the cloud affects the multicast.

- Finally, we are developing analytical models that will allow us to better understand the energy/message efficiency tradeoffs in this domain.

Acknowledgment

This work was supported by NSF grant ANIR-9902714 and ONR grant N000149910499.

REFERENCES

[1] E. Bommaiah, M. Liu, A. McAuley and R. Talpade, "AM-Route: Ad Hoc Multicast Routing Protocol", *Internet-Draft*, draft-talpade-manet-amroute.txt, Aug. 1998, Work in progres.

[2] C-C. Chiang and M. Gerla, "On-demand multicast in mobile wireless networks", *Proc. IEEE ICNP'98*.

[3] C-C. Chiang and M. Gerla, "Adaptive shared tree multicast in mobile wireless networks", *Proc. IEEE GLOBECOM'98*.

[4] C-C. Chiang, M. Gerla and L. Zhang, "Forward group multicast protocol (FGMP) for multihop, mobile wireless networks", *ACM-Baltzer J. Cluster Computing: SPecial Issue on Mobile Computing*, Vol. 1(2), 1998.

[5] J.J. Garcia-Luna-Aceves and E.L. Madruga, "A multicast routing protocol for ad hoc networks", *Proc. IEEE Infocom'99*, New York, NY, Mar. 1999, pp. 784-792.

[6] Chalermek Intanagonwiwat, Ramesh Govindan and Deborah Estrin, "Directed Diffusion: A Scalable and Robust Communication Paradigm for Sensor Networks", *IEEE/ACM Mobicom'00*.

[7] S.-J. Lee, M. Gerla and C.-C. Chiang, "On-Demand Multicast Routing Protocol", *Proc. IEEE WCNC'99*, New Orleans, LA, Sept. 1999, pp. 1298-1304.

[8] S.-J. Lee, W. SU and M. Gerla, "Ad hoc wireles multicast with mobility prediction", *Proc. IEEE ICCCN'99*, Boston, MA, Oct. 1999, pp. 4-9.

[9] E.L. Madruga and J.J. Garcia-Luna-Aceves, "Multicasting along meshes in ad hoc networks", *Proc. IEEE ICC'99*, Vancouver, Canada,Jun. 1999, pp. 314-318.

[10] E. Pagani and G. P. Rossi, "Reliable broadcast in mobile multihop packet networks", *Proc. ACM MOBICOM'97*, 1997, pp. 34-42.

[11] H. Takagi and L.Kleinrock, "Optimal transmission ranges for randomly distributed packet radio network", *IEEE Trans. Comm.* Vol. COM-32, pp. 246-257, March 1984.

[12] C. L. Willamson, T. G. Harrison, W. L. Mackrell and R. B. Bunt, "Performance evaluation of the MoM mobile multicast protocol", *Mobile Networks and Applications*, Vol. 3(2), 1998, pp. 189-202.

[13] C.W. Wu, Y.C. Tay and C.-K. Toh, "Ad hoc Multicast Routing protocol utilizing Increasing id-numberS (AMRIS) Functional Specification", *Internet-Draft*, draft-ietf-manet-amris-spec-00.txt, Nov. 1998, Work in progress.