Parameterless broadcasting in static to highly mobile wireless ad hoc, sensor and actuator networks

Adnan Afsar Khan[1], Ivan Stojmenovic[2,1] and Nejib Zaguia[1]
[1]SITE, University of Ottawa, Ontario, Canada {akhan083, ivan, zaguia} @site.uottawa.ca
[2]EECE, University of Birmingham, UK

Abstract—**In a broadcasting task, source node wants to send the same message to all the other nodes in the network. Existing solutions range from connected dominating set (CDS) based for static networks, to blind flooding for moderate mobility, to hyperflooding for highly mobile and frequently partitioned networks. The only existing protocol for all scenarios is based on some threshold parameters (which may be expensive to gather) to locally select between these three solution approaches. Here we propose a new protocol, which adjusts itself to any mobility scenario without using any parameter. Unlike existing methods for highly mobile scenarios, in proposed method, two nodes do not transmit every time they discover each other as new neighbors. Each node maintains a list of two hop neighbors by periodically exchanging 'hello' messages, and decides whether or not it is in CDS. Upon receipt of the first copy of message intended for broadcasting, it selects a waiting timeout and constructs two lists of neighbors: neighbors that received the same message and neighbors that did not receive it. Nodes not in CDS select longer timeouts than nodes in CDS. These lists are updated upon receipt of further copies of same packet. When timeout expires, node retransmits if the list of neighbors in need of message is nonempty. 'Hello' messages received while waiting, or after timeout expiration may revise all lists (and CDS status) and consequently the need to retransmit. This provides a seamless transition of protocol behavior from static to highly mobile scenarios. Our protocol is compared to existing solutions. It was shown to be superior to all of them in number of retransmissions and reliability.**

Keywords—Broadcasting, wireless networks, ad hoc networks, sensor and actuator networks

I. INTRODUCTION

We consider all network scenarios with respect to mobility. Each node, or its neighborhood, can be *static*, *moderately mobile*, or *highly mobile*. The distinction between static, moderately and highly mobile networks is debatable, and we present here one possible view. In *static* networks, network topology does not change during broadcasting task (thus nodes may still move slowly). Roughly speaking, the distinction between moderate and high mobility is based on the percentage of neighbor changes of a node while broadcasting is in progress. A network is moderately mobile if all nodes in it are either static or moderately mobile. Thus even single moderately mobile node among static nodes classifies the network as moderately mobile. Finally, a network is highly mobile if at least one node in it is highly mobile. Therefore few high-speed vehicles among lot of pedestrian along the road may classify the network as highly mobile.

The primary goal of a broadcasting task is to deliver the message to all nodes in a network. We calculate the nodes that could have somehow received the message in the following way. Initially the source node is colored black, and all the other nodes are white. In each slot, color of a node is changed from white to black if at least one of its neighbors is colored black in earlier time slot (recall that nodes can move between slots). We assume that the broadcasting process must complete within a finite time *T*. When time *T* expires, black nodes are exactly those that could have received the message from the source node. Then the reliability of a particular protocol is the percentage of black nodes that received the message. This gives more accurate results for reliability since it is impossible for nodes that are always disconnected to receive a message and therefore they are not considered. Moreover, it also considers nodes that may not be connected to the source at any given moment in time but could receive a message from the source. For instance, if another node moves between the areas where a source and destination node is located and carries the message. Our primary goal is hieve high reliability while minimizing the total number of retransmissions.

We consider only *localized* broadcasting protocols. There exists a body of knowledge about centralized broadcasting where source node knows the whole network topology and can determine the whole broadcast process. However, collecting required global knowledge demands unacceptable communications overhead for dynamic networks. Such knowledge gathering may have to be done by applying a variant of localized broadcasting. In localized protocols, node has only local knowledge about the network. One extreme is lack of any awareness of neighbors. That is, nodes do not send control 'hello' or beacon messages to inform neighbors about their presence. If each node periodically transmits 'hello' message then 1-hop knowledge can be gained. If subsequently, each node transmits the list of 1-hop neighbors then the knowledge of 2-hop neighbors (direct neighbors and their neighbors) can be gained. Our proposed algorithm is based on 2-hop knowledge. One option is to transmit also geographic position with 'hello' messages (positional information), in addition to node IDs. We however do not consider this option and restrict our investigation to topological knowledge. The proposed new algorithm PBSM (Parameterless Broadcasting from Static to Mobile), is designed for 2-hop topological knowledge. It also works with 1-hop positional knowledge, with appropriate modification in neighbor set definitions. This is important to observe for applications in sensor networks, since sensors normally do not have their IDs. Geographic position of sensors may be used then instead of their IDs, to derive whether or not any two neighbors are neighbors themselves. Nodes can make backbone decision for connected dominating set (CDS), but have no awareness of 2-hop neighbors which are not direct neighbors. PBSM follows same steps except applying them on different local knowledge.

Most existing solutions address a single scenario. In *blind flooding*, each node, upon receiving the message for the first time, will retransmit it, and ignore further copies of the same message. This is a traditional broadcasting protocol that does not require neighbor knowledge. In dense networks it can cause lot of redundancy, collisions and contentions and reduce rather than increase reliability [10]. Improved solutions aiming at full network coverage require 2-hop neighbor topological knowledge and are based on connected dominating sets and neighbor elimination [19]. However the mobility makes the maintenance of such knowledge expensive and therefore blind flooding is considered still as the favorite protocol for scenarios with moderate mobility. Blind flooding may not suffice in networks with temporary partitions and/or high mobility. *Hyper-flooding* was proposed for such scenarios [3, 7, 21]. *Hyper-flooding* is a flooding technique similar to plain flooding. In hyperflooding, there are additional retransmissions whenever a node discovers a new neighbor. Reliability is thus increased at the cost of high message overhead.

We found only one protocol that attempts to describe a single broadcasting protocol suitable for all scenarios [21]. That protocol is based on applying high and low thresholds. Each node calculates its low and high threshold value based on past relative movements in its neighborhood. Each node keeps track of relative movements in its neighborhood, and compares it with low and high thresholds. Thus different nodes can make different decisions and run different protocol modes. Each node decides to run scoped (restricted) flooding, blind flooding or hyper-flooding based on its own threshold values. Two threshold types were considered: mobility and traffic based. The protocol has a number of problems. First, the requested parameter value may be difficult or impossible to gather. For instance, the protocol uses speed and direction of movement, which requires accurate position information and adds some hardware to nodes and overhead to hello message exchanges. Traffic parameters are based on measuring collisions but they reduce reliability in high volume traffic and increase unnecessarily overhead in low volume traffic.

Our objective is to describe a broadcasting protocol that will adapt itself to any mobility scenario automatically, without calculating, tracing and applying any type of thresholds; that is, without using any parameter. Therefore our protocol will not measure speed or direction of movement, and will not monitor traffic for the purpose of deciding retransmission behavior. Furthermore, we are looking for a protocol that will have fewer retransmissions and higher reliability. The proposed protocol is named "Parameterless Broadcasting from Static to Mobile networks" (PBSM). Each node maintains two lists, *R* (neighbors that received message) and *N*, (neighbors that did not receive the same message). These lists are updated upon the receipt of each copy of the message. Nodes in CDS select shorter timeouts than nodes not in CDS. When waiting period expires, node retransmits if *N* is nonempty. 'Hello' messages may refresh *R* and *N* and subsequently cause further retransmissions. This protocol provides a seamless transition of protocol behavior from static to highly mobile scenarios without monitoring or measuring any speeds or knowing what mobility scenario is. Furthermore, we expect that this protocol will have much better tradeoffs between communication overheads and reliability than the sole existing threshold based protocol. It should be also superior to any other existing broadcasting protocol since they address single mobility scenario and perform poorly in other scenarios while our protocol is adaptive and remains always competitive.

This article is organized as follows. Literature review is in section 2. Our new protocol is described in section 3. Section 4 describes the simulation results of comparison with existing methods. Conclusion and references complete this article.

II. LITERATURE REVIEW

There exist a plethora of proposed broadcasting protocols. Their survey is given in [14, 17]. We describe here only protocols that are relevant to our proposed method and the adaptivity goal.

2.1 Broadcasting in static networks

Wu et al described, in a series of articles (starting from [23]), a lightweight backbone construction scheme. We will use a modified definition from [19, 20] of basic concept [23], because of its reduced message overhead. A node is an *intermediate* node if it has two unconnected neighbors [23]. A node *A* is covered by neighboring node *B* if each neighbor of *A* is also neighbor of *B*, and *key(A)<key(B)*. Nodes not covered by any neighbor are *inter-gateway* nodes. A node *A* is covered by two connected neighboring nodes *B* and *C* if each neighbor of *A* is also a neighbor of either *B* or *C* (or both), *key(A)< key(B)*, and *key(A) < key(C)*. An intermediate node not covered by any neighbor becomes an *inter-gateway* node. An inter-gateway node not covered by any pair of connected neighboring nodes becomes a *gateway* node. Dai and Wu [5] introduced a *generalized dominating set*, where coverage can be provided by an arbitrary number of connected neighbors. The definition was modified in [17] to avoid similar message exchanges between neighbors, as follows. Node *A* is covered by its direct neighbors *B, C, D,...* if the neighbors *B, C, D, ...* create connected subgraph, any neighbor of *A* is a neighbor of at least one of nodes *B, C, D, ...* and *key(A) < min (key(B), key(C), key(D),...)*. It is computationally simplified by Carle and Simplot-Ryl [4], as follows. First, each node checks if it is an intermediate node. Then each intermediate node *A* constructs a subgraph *G* of its neighbors with higher *key* values. If *G* is empty or disconnected then *A* is in CDS. If *G* is connected but there exists a neighbor of *A* which is not a neighbor of any node from *G* then *A* is in CDS. Otherwise *A* is covered and is not in CDS. Dijkstra's shortest path scheme can be used to test the connectivity. In *enhanced* CDS by Dai and Wu (elaborated in [8]), 2-hop neighbors can be used to cover 1-hop neighbors for smaller CDS. In this method, an intermediate node *u* is not in CDS if there exists a connected set *A* of its 2-hop neighbors with higher priorities, such that each neighbor of *u* either belongs to *A* or is a neighbor of a node in *A*. Otherwise *u* is in CDS.

Wu's concepts require either 1-hop knowledge of neighbors with their position, or 2-hop neighbor topology information. Experimental data from several sources confirm that Wu's concepts provide small size CDS on average. It was proven in [5] that generalized CDS concept has constant approximation ratio on average, and very low probability of having infinitely large approximation ratio. Each node makes decisions about CDS membership (in Wu's concept) without communications between nodes beyond the message exchanges that nodes use to discover each other and establish neighborhood information.

In [19], the following framework and general algorithm were established for a reliable broadcasting. The algorithm is based on two concepts: connected dominating set (CDS) as the particular type of backbone that provides reliability, and neighbor elimination scheme. In *neighbor elimination* scheme [12, 19], a node does not need to rebroadcast a message if all its neighbors are believed to be covered by previous transmissions. After each received copy of the same message, a node eliminates, from its rebroadcast list, neighbors that are assumed to have received correctly the same message (based on the local knowledge). If the list becomes empty before the node decides to rebroadcast, the re-broadcasting is canceled.

The general Dominating Sets and Neighbor Elimination Scheme (DS-NES) [19] for intelligent flooding proceeds as follows. The source node transmits the packet. Nodes not in CDS do not retransmit the packet. Upon receiving the first copy of the packet, node in the CDS will select a timeout period to wait. It will also eliminate from its forwarding list (originally containing all 1-hop neighbors) all neighbors that received the same copy of the message. While waiting, more copies of the packet could be received. For each of them, all neighbors receiving it are eliminated from the forwarding list. When timeout expires, the node will retransmit if its forwarding list is nonempty, otherwise it will cancel retransmission. This framework was applied in [19] using clustering based and Wu's concept based backbones.

2.2 Broadcasting in network of moderate mobility

In [9], Lou and Wu study two environments to handle mobility. In the "static" one, mobile hosts are allowed to roam freely. However the broadcast process is done quickly so that both 1-hop and 2-hop neighbor sets remain the same during the process for each host. In addition, each host has an updated and consistent 1-hop and 2-hop neighbor sets when the broadcast process starts. Delivery of the broadcast packet is guaranteed as long as the selected forward nodes cover all hosts. In the "dynamic" environment, the broadcast process is still done quickly as in the static environment; so that both 1-hop and 2-hop neighbor sets remain the same during the process for each host. However, a node may not update its neighbor sets in a timely and consistent manner because of mobility. Under this model, the broadcasting is no longer fully reliable. A simulation [9] shows that the broadcast delivery rate still remains high in an ad hoc network with slow to moderate speed (with respect to the transmission range) mobile hosts using an ideal MAC layer without contention and collision. This high delivery rate is partly because of the broadcast redundancy in selecting the forwarding nodes. Therefore, while excessive broadcast redundancy is harmful and will cause the broadcast storm problem, some degree of redundancy is useful for reliability purpose.

Zhang and Jiang [24] compared, analytically and experimentally, the performance of several broadcasting algorithms in mobile scenarios. The selected algorithms are MPR (multipoint relay) and dominating set based on generalized definition by Dai and Wu [5]. Both schemes are improved by considering relative stability levels of links in neighborhood and averaged over the network. This level may alter the decision on forwarding. Thus the modification adjusts level of flooding based on mobility, but does not provide the opportunity for multiple retransmissions by the same node (the same is valid for protocol [9]), which may be required in highly mobile scenarios.

Walker, Glenn and Clancy [25] proposed and analyzed the following simple broadcasting algorithm. Whenever two nodes meet, they exchange IDs of messages they carry. Each of them has two counters for each message it carries: NC (number of consecutive non-carriers of the message) and ND (number of consecutive carriers of the message). There are two fixed thresholds $C\geq 0$ and $D>0$. If $NC\geq C$ then node hands off a copy of the message and resets $NC=0$. If $ND\geq D$ then node drops its copy of message and resets $ND=0$. This algorithm assumes that every meeting between two nodes can be accounted for and does not interfere with meetings with other nodes. Details of hello message dynamics and conflict resolutions were not discussed. A node may encounter a number of new neighbors simultaneously, and decide to drop the packet before continuing on a journey to another area. The message then expires (higher node density will contribute to it) before that area is reached. One possible advantage is that overhead of communicating neighbors is replaced by overhead of communicating merely message IDs. This protocol was not considered here because of different assumptions.

2.3 Broadcasting in highly mobile network

Cooper, Ezhilchelvan and Mitrani [3] proposed encounter based protocol, as follows. Upon receiving a new message, node u stores it, together with counter c set originally to 0. If the current neighborhood contains nodes other than sending one, u broadcasts message and increase c by 1. Encounter is discovery of a new neighbor that was not present in the previous list of neighbors (thus a node that was neighbor and disappeared for just one 'hello' cycle is still considered as encountered upon return). At every encounter, if $c< T$ (T is a parameter) then broadcast message and increase counter c by 1. Authors show that, under certain simplified assumptions, a high coverage is achieved by making a total of $O(\log n)$ messages per node, where n is the number of nodes in the network, and the network diameter is $O(\log n)$. Encounter based protocol [3] has two differences with hyper-flooding [21]. Unlike hyper-flooding, there are no additional transmissions when a node receives data message from a node that is not in the current neighbor list. The second difference is that in this protocol message expires upon certain number of encounters rather than after certain time.

In [11], O'Dell and Wattenhofer designed flooding algorithms for dynamic networks with dynamic edges and fixed nodes. Among the assumptions used are that the network stays connected at all time, and evolves slower than the message transit time between adjacent nodes. Their model, therefore, differs significantly from the model considered here. Their CounterFlooding algorithm requires one input *T* and intercepts two types of events: the connectivity-driven events and the message-driven events. When the message is first received at a node *u*, the counter *c* is reset to 0, and the message is retransmitted. If *u* is notified of a neighborhood change with the arrival of connectivity-driven event, it broadcasts the message and increments the counter *c* by one until it reaches the maximum *T*.

In [13], Soedarmadji and McEliece relax the flooding algorithm's assumptions by removing the requirement that the network stays connected at all time, and extend the algorithm to solve the problem where dynamic nodes are also involved. The extended algorithm is reliable: it guarantees message-passing to all the destination nodes and terminates within a time bounded by a polynomial function of the maximum message transit time between adjacent nodes, and the maximum number of nodes in the network.

Protocols [3, 11, 13] are designed specifically for highly mobile scenarios. They perform similarly to blind flooding in static networks and produce excessive messages in well-connected networks with moderate mobility.

Hahner, Becker, and Rothermel [7] modified hyper-flooding scheme [21] in two ways. Instead of sending full message as in [21], a short message containing advertisement is sent. The short message uniquely identifies the full message, and the receiving node may request the full message or ignore it if it already has it or does not want it. While [21] is designed for one-to-many medium access layer, [7] has one-to-one model in mind, where message sent by one node to another node is assumed not to be simultaneously received by other neighbors of sender node. More precisely, advertising can reach all neighbors at once, but these neighbors in need of listed items need to respond by separate messages, and receive separate copies of the same message. It is possible to consider also the model where one response by any neighbor triggers message transmission to all neighbors, including perhaps few more in need of same message. Thus, to disseminate information across partitions, an approach similar to hyper-flooding [21] is added: whenever a node discovers a new neighbor, it is allowed to re-advertise observations as long as TTL (time to live) has not expired. When TTL reaches 0, the message is erased from database. The authors propose few variants on the advertising priorities and ordering of several messages that may be broadcast simultaneously (but not necessarily from the same source or start time). When full message is relatively short, this protocol is inefficient.

2.4 Adaptive Broadcasting

Viswanath and Obraczka [21] proposed different heuristics to deal with broadcast reliability in static to highly mobile environments. Each node decides between three modes for broadcasting task, based on two thresholds. In the *scoped flooding* [21], periodical hello messages contain 1-hop neighbors list. If the receiving node's neighbor list is a subset of the transmitting node's list, then it does not re-broadcast the packet. We note that this is a special case of the neighbor elimination scheme [12, 19]. The *plain flooding* mode is the same as blind flooding. In the *hyper flooding* mode, additional re-broadcasts can be triggered upon receiving a data packet or hello message from a new neighbor. Note that nodes running scoped flooding or plain flooding do not retransmit message again after discovering any new neighbor.

Viswanath and Obraczka [21, 22] described several variants for thresholds determination, based on measuring the *network load* (MAC layer collisions) or *relative velocity*. The computation and storage of relative velocity or the number of collisions introduces additional hardware, software and/or communication overheads. Measuring relative speeds poses additional problems as positional information or special hardware is required. Network load as threshold parameter makes the protocol behavior dependent on other traffic. One option for selecting thresholds based on local movement velocity is to choose two fixed values [21]. However, [21, 22] favored other ways with presumably better performance. Nodes send velocity (speed and direction) information as part of their hello messages. Each node is then able to compute its velocity relative to all of its immediate neighbors, and to maintain a running average as well as the minimum and maximum value of relative velocity for the past five time windows. Based on the current value of relative velocity and its past history, each node adaptively chooses a low threshold and high threshold value for the current time window. Details were not given precisely. In our implementation, time window corresponds to the time between two hello messages, and the span between minimum and maximum past averages is divided into three equal span ranges by two thresholds. Thus, different nodes can make different decisions and run different protocol modes. It is possible that nodes moving at slow speed run hyper-flooding protocol only because neighbors are static. Similarly, highly mobile nodes may behave in static mode only due to faster neighbors. Next, when all nodes slow down or accelerate simultaneously, they all move to static or highly mobile modes simultaneously. That is, protocol performance becomes dependent on past history as variable, which introduces sub optimality scenarios.

III. PARAMETERLESS BROADCASTING FROM STATIC TO MOBILE NETWORKS

3.1 Protocol overview and illustrations

We propose an adaptive broadcasting protocol that does not require nodes to monitor and exchange their position, movement and/or traffic information, and yet performs better than existing threshold based protocol [21]. Protocol is localized, and is based on applying CDS and neighbor elimination concepts on currently available neighborhood information. Thus two nodes do not transmit every time they discover each other as new neighbors. The proposed PBSM protocol does not rely on any threshold and provides smooth transition of protocol behavior based on network dynamic.

Unlike other methods, in the proposed method, two nodes do not transmit every time they discover each other as new neighbors. This is the main novelty of proposed PBSM protocol. The protocol should behave in a way such that success rate is nearly preserved while reducing flooding rate (overall number of messages sent). The other change is not to always rebroadcast first time message is received, as in the blind flooding protocol, considered primarily for moderately mobile scenarios. This will create excessive messaging in dense networks with no or slow topology changes. In these networks local knowledge information is nearly preserved while broadcasting is in progress, and therefore retransmissions can be first made by backbone nodes with neighbors still in need of message.

Nodes periodically exchange hello messages to update local knowledge up to two hops. CDS is calculated after each hello message round. Source node transmits the message. Upon receiving the message for the first time, each node initializes two lists: receiver list *R* containing all nodes (up to 2-hop distance) believed to have received the packet, and list *N* containing neighbors in need of message. Node sets a timeout waiting period. If node is not in CDS then it selects longer timeout than node from CDS so that nodes in CDS react first. For each further message copy received, and its own message sent, every node updates *R, N* and the timeout. At the end of timeout period it transmits if *N* is nonempty. Message is memorized until *T* hello messages are received. For each hello message received, *N* is updated. Nodes that are no longer 1-hop neighbors are eliminated from the list, while new neighbors, not present in *R,* are added. Regardless of previous decisions, all nodes that so far received broadcast packet check whether new *N* is nonempty. If so, they start fresh timeout. Nodes not in CDS also run timeouts if their *N* lists remain nonempty.

Before giving algorithmic details, we will illustrate protocol behavior on several examples. Consider first its behavior on static networks. In Fig. 1, *S* is the source node while *A, F, G* and *H* are backbone nodes (following generalized dominating set definition). *F* is the only backbone node to receive the transmission from the source *S* and thus it will retransmit first. Its backbone neighbors *A* and *G* choose timeout proportional to their number of uncovered neighbors 1/3 (neighbors *D, I* and *H* of *A*) and ½ (neighbors *H* and *L* of *G*), respectively. Node *E* is not in CDS and runs a longer timeout 6 (=5+1/1, one neighbor *L*). Thus *A* retransmits next. At that moment *G* extends its timeout proportional to 1/1-1/3 (one uncovered neighbor *L* left, 1/3 time elapsed), while *H* sets its timeout in proportion to 1/3 (neighbors *I, J, K*). Thus *H* retransmits next. *G* and *I* cancel timeout as their *N* list becomes empty. *E* further reduces its timeout to 6-1/3-1/3 (1/3 more time elapsed). Node *L* (non-CDS) sets a longer timeout 6 (i.e. 5 + 1/1, one uncovered neighbor *E*). *E* retransmits since it is not aware of coverage for L, and broadcasting completes successfully after retransmissions from nodes *F, A, H* and *E*. For static networks PBSM behaves close to the DS-NES protocol [19], with few added transmissions. Note that scoped flooding [21] (assuming same waiting timeout function) would require retransmissions also from all nodes *G, L* and *I* (all nodes except those covered by single neighbor). Other methods will perform blind flooding.

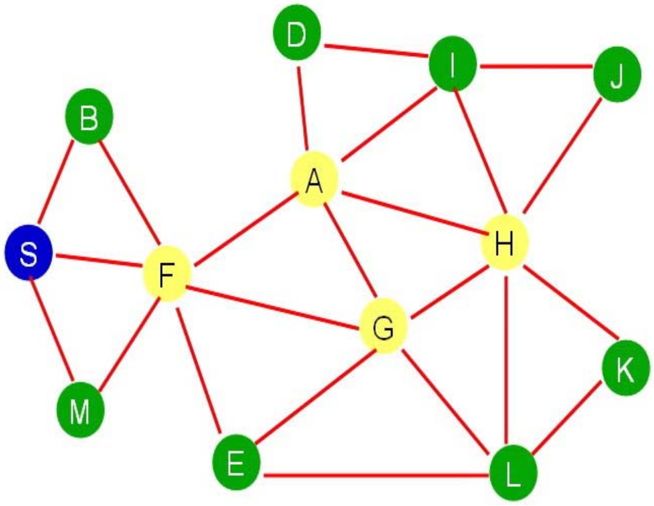

Figure 1. Source *S* and backbone nodes *A, F, G, H* in a static network

Consider example in Fig. 2, where network is initially partitioned into two components. After one round of 'hello' messages node 12 was discovered (top of figure) as new neighbor of nodes 16 and 10 making network connected. Node 12 continues to move downward (all other nodes remain static all the time) and in the next hello message becomes neighbor of five nodes. Broadcasting is initiated by node 14, and blind flooding would propagate in the left portion of the network, while the other partition never receives the message, since there is no action of 'refreshing' retransmissions after 'hello' message discovered new node. Hyper-flooding would 'bridge' the message propagation but will cause additional retransmissions by all nodes that discover node 12, from both network parts. Consider now adaptive broadcasting [21] with the historical relative velocities as thresholds. Nodes 16, 8 and 14 may not retransmit due to arrival of node 12 if themselves and their neighbors on the left partition 4, 6, 7, and 13 slow down, thus compensate for the movement of node 12, and therefore do not declare hyper-flooding mode. Similar examples can be constructed for any type of threshold determination. PBSM will cause node 16 to retransmit after discovering node 12 since it becomes CDS member (bridging two partitions) and its list *N* becomes nonempty. Retransmission is similarly done by node 12 and the network on the right side then receives the message. Similar example is movement of a node between two 'cities'. Upon arrival, it needs to be above threshold value to transmit the message originated in one city to the new city. However, for any type of threshold setup it is straightforward to find parameter value when this will not happen. For example, nodes in the new city may be highly mobile, or having lot of traffic, or new node slows down upon arrival etc.

3.2. Details of PBSM algorithm

Before source node decides to initiate broadcasting of a message, nodes gain local knowledge and CDS status from the previous hello message round. Nodes react to two events:

receiving message intended for broadcasting, and sending/receiving hello messages at fixed intervals. These activities proceed until a fixed number *T* of hello messages occurred after the initial source transmission. *T* is a parameter that determines lifetime of message being broadcasted. The remaining lifetime can be carried with the message if synchronization of lifetime is desirable. It may be beneficial that each node starts its own *T* at the moment of receiving the first copy of message, to reduce message size by not informing neighbors about the remaining lifetime.

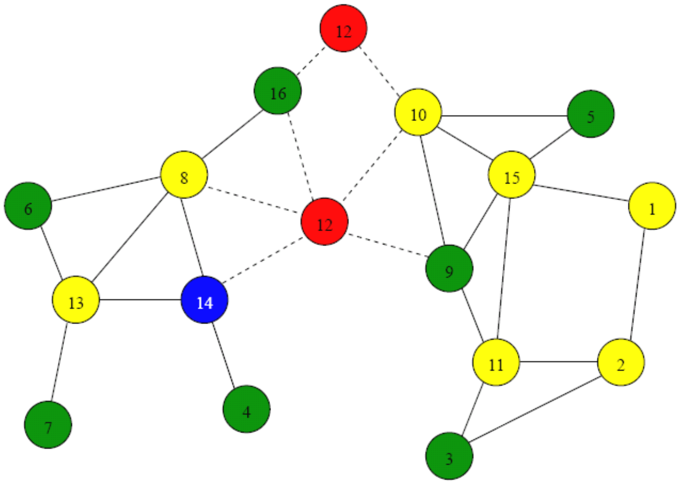

Figure 2: Improvements in transmission and reliability when node joins two partitioned network

When a node *X* receives message for the first time, it initializes two lists, *R* and *N*. List *R* consists of nodes that are believed to have received the same message based on available and current local knowledge of *X*. This includes the sender and its known neighbors. The remaining 1-hop neighbors of *X* are included in list *N* (nodes that would benefit from possible transmission of *X*).

We assume that there exists a timeout function that assigns waiting time to each node that decides to evaluate the need for retransmission. It can be selected in various ways. For example, it could be a random number in interval (0, maxtime] or it could be proportional to $1/|N|$ ($|N|$ is the number of nodes in *N*).

If the list *N* is empty then *X* cancels the retransmission but retains and later possibly updates *R*. If *N* is nonempty then *X* selects a waiting timeout. If *X* is further not in CDS it adds maximal possible timeout value (e.g. maxtime) to the timeout (this means that it waits for nodes in CDS to decide first).

Whenever a node *X* receives message for broadcasting, it updates its lists *R* and *N* by adding (eliminating, respectively) all known nodes believed to have received the same message. Note that *R* is being updated even if timeout at node *X* expired or has not even started. If the list *N* becomes empty then *X* can terminate running timeout and decide not to retransmit. If node *X* is running a timeout that depends on $|N|$, it updates it. If timeout at *X* expires while its list *N* remains nonempty then *X* retransmits the message. It also updates its list *R* by moving nodes from *N* there, leaving *N* empty.

During each of *T* upcoming 'hello' message rounds, the following updates occur, which may cause additional retransmissions from otherwise terminated process. All nodes that have stored unexpired messages for broadcasting consider new or perhaps very first retransmission (including nodes that have active timeouts). Former 1-hop neighbors that are not anymore listed as direct (1-hop) neighbors are eliminated from *N*, since *X* cannot cover them anymore. Note that this may cause a running timeout to terminate (if *N* becomes empty). Newly reported 1-hop neighbors are verified if they are in accumulated list *R*. If so, no changes are made with respect to such neighbors. Otherwise they are added to list *N*. This may cause (re)activation of timeouts at *X* if *N* was empty before adding new node.

Hello messages may occur while some nodes were running timeouts already. These nodes behave as if hello message did not occur; timeout are impacted only by possible changes in N lists. That is, they continue running the remaining timeouts (more precisely, it is new timeout value reduced by already expired waiting time). Note that 'refreshed' or restarted timeouts can also be considered. However, we opted for described version to avoid problems with relatively short hello message intervals that may delay retransmissions for quite long time leading even to its expiration. Detailed code of PBSM protocol is given in the master thesis of the first author.

IV. SIMULATION

The proposed PBSM protocol was implemented and compared with the only known competing protocol VO [21] addressing all mobility scenarios. To distinguish between gains based by threshold avoidance and gains made by replacing scoped flooding with CDS-NES approach, we added such protocol VO-CDS. We also added protocol CEM [3] in our comparison as a benchmark for realistically achievable reliability and upper bound on message overhead. The simulator is developed in Visual C++.

A simplified IEEE 802.11 MAC protocol has been implemented. It is based on *Carrier Sense Multiple Access with Collision Avoidance* (CSMA/CA) scheme. After receiving message, node waits for *W* message free slots. Afterwards it retransmits for *p* consecutive slots. In our experiments *W* is random integer in [0,25] while $p=128$.

We assumed the unit disk graph model. There is a single broadcasting task in the network. Nodes within fixed transmission radius receive message correctly unless it collides with message from other neighbor. These are in fact favorable assumptions for competing methods because they send more messages. There are no acknowledgements for any received message. We simulated 2-hop variant of PBSM without using position information. However, we gave to VO and VO-CDS the privilege of using the position information to derive speeds and directions of movements.

The following independent variables are used in simulation: *n* (the number of nodes), *d* (average number of neighbors, or density), *B* (the number of messages that can be transmitted between two hello message rounds), *D* (the average number of transmission radii traveled by nodes between two hello messages). We fixed T (the message lifetime) to 10 hello message intervals, believing it has no major impact on main findings. Most experiments use values n=50 and d=4. All nodes are randomly generated in a square of side a. That is, x- and y- coordinates of initial node placement are chosen at random between 0 to a (a=470 is used for viewing movement and transmissions on screen). The transmission radius r used is $r^2 = (d * a * a)/ ((n-1) * \pi)$. This provides approximate degree d during the experiment. The generated graphs were disconnected most of the time. Each of the presented data is average of 20 runs with same parameters. For each scenario, there are $T=10$ fixed intervals (each of length H) of hello messages. Sending broadcast message to immediate neighbor requires fixed time B', and thus $B=H/B'$ is the number of retransmissions that can be made between two hello messages. B' is divided into p slots. Broadcasting process begins at a random time between two hello messages but terminates simultaneously at all nodes after TH time.

Nodes move following the random waypoint mobility model [2]. Nodes move toward randomly selected turning point with constant speed selected at random between 0 and the maximum speed. Upon reaching the turning point, node chooses a random new speed and moves to new randomly selected turning point. Under maximal speed, node can travel distance D' between two hello messages. If $D'=D*r$, then the maximal speed is $D*r/H$ and average speed is half of that.

The following metrics were used to evaluate the performance of compared protocols: the total number of transmissions per node (*trans_count*), and the *reliability* (the number of nodes that received the message originated at the source divided by the total number of nodes that could have received the message).

Three types of mobility scenarios, referred to as static, moderate and high mobility, are simulated. In a static scenario all nodes were static $(D=0, B=10, n=50, d=4)$. All methods were 97% reliable (note that graphs were not forced to be connected). *Trans_counts* were 1.22 for *PBSM* and *VO-CDS, 1.42* for VO and *2.18* for CEM protocol. Thus replacing scoped flooding by CDS-NES resulted in about 16% saved messages while CEM in fact behaved like blind flooding with one message per connected node.

In moderate mobility scenario, n=50, d=4, 25 nodes move with moderate speed (D=1/2, or maximum speed $r/(2H)$), whereas the remaining nodes are static. B was varied from 3 to 21 in increments of 2 for ten runs, to account for its impact. The *trans_count* values were 2.1 for PBSM, 2.5 for VO-CDS, 2.92 for VO, and 6.18 for CEM protocol. Thus PBSM has about 28% less transmissions. On average, in VO-CDS protocol, 42% of nodes are in scoped/CDS mode, 22% of nodes are in plain and 36% of nodes are in hyper mode. Because of the way thresholds are determined, when average speeds are decided independently, it is in fact expected that about one third of nodes are in each of mobility categories, regardless on speed maximal speed distributions. This is valid even for static and highly mobile scenarios. The reliability of protocols are: 91% for PBSM, 70% for VO-CDS, 70% for VO and 91% for CEM. We can conclude in this scenario that PBSM provides reliability as high as possible, matching CEM, and winning over VO and VO-CDS, while reducing transmission counts considerably.

In high mobility scenario, almost all nodes move, n=50, d=4, D=1, so maximum speed is r/H. B was varied from 3 to 21. The *trans_count* values were 3.64 for PBSM, 3.32 for VO-CDS, 5.04 for VO, and 11.98 for CEM protocol. PBSM and CEM methods were 99% reliable. VO and VO-CDS methods were 70% reliable. The differences are not large since nodes are highly mobile and discover new neighbors frequently. Still PBSM made savings compared to others, without sacrificing any reliability. On average, 28% of nodes are in scoped/CDS mode, 32% of nodes are in plain mode, and 40% of nodes are in hyper mode. This confirms expectations of about one third of nodes in each class regardless of mobility ranges.

We next considered a corridor scenario, where B=4, n=25, D=1, d=4, and 12 nodes lie in each of two partitioned 'cities' while one node moves slowly (so that it does not enter hyperflooding mode upon arrival) between the two with sufficient speed to retransmit before message expires but after blind flooding terminates. Nodes are highly mobile ($D=1$) within their cities. Trans-counts were 0.84 for PBSM, 0.184 for VO-CDS, 0.32 for VO and 2.08 for CEM. The difference is due to lack of transmissions in the second city. The reliability was 100% for CEM and PBSM, 52% for VO and VO-CDS. Note that similar scenarios can be created with arbitrary ratio of nodes in two cities and therefore arbitrary reliability (or failure) for threshold based protocol VO. Further, similar scenarios can be constructed for arbitrary choices of thresholds.

The impact of mobility parameter D was studied for B=5, n=50, d=4, and D ranging from .15 to 1.5 with .15 increments. PBSM has on average the lowest transmission counts. PBSM and CEM were 96% reliable whereas VO and VO-CDS were 75% reliable because at lower mobility, most of the nodes in VO and VO-CDS were not in hyper mode.

The impact of density was studies for B=10, n=50, D=0.75. PBSM has small advantage for sparse graphs (d=2, 3, 4) but large advantage for $d \geq 5$. However, the difference in reliability decreases as d increases.

We also considered a 'disaster' scenario to show further weaknesses of studied threshold determination method in VO and VO-CDS. Selected parameters were B=3, n=25, D=0.75, d=4. The speed was increased in each of B slots between two hello messages, but obviously new neighbor speeds were learned only at 'hello' message time (at the beginning of new time window). That caused most nodes to be suddenly above high threshold. Indeed, on average, 20% of nodes are in scoped/CDS mode, 30% of nodes are in plain mode, and 50% of nodes are in hyper mode. All protocols are around 98% reliable. However, the number of transmissions per node in PBSM is around 30% less than methods of [21, VO-CDS]. In a similar 'traffic jam' scenario, all nodes analogously reduce their speed. Now, in an inverse behavior, on average, 60% of nodes are in scoped/CDS mode, 28% of nodes are in plain mode, and 12% of nodes are in hyper mode. The number of transmissions per node in PBSM is now around 15% more than in the methods of [21, VO-CDS]. But reliability remains over 98% while for VO and VO-CDS they are below 80%.

V. CONCLUSION

One of protocols from [7] considers short advertisements, sent whenever new node is found, followed by full message if any neighbor responded requesting it. The approach is justified when full message is much longer than short one. The protocol for disseminating short messages alone can be considered as a broadcasting task in itself. We propose two modifications from the protocol [7]. The first one is to introduce waiting periods before sending advertisements. The second is that it is not necessary for all nodes to send advertisements. Only those that are in CDS, and with nonempty *N* lists, can do so, as in the protocol we proposed. Thus we can apply our proposed protocol to disseminate short messages. Because of protocol similarity, there was no need to simulate this proposed variant and compare it with the one in [7]. The cost of sending full messages appears similar in both protocols, because each node will request it at most once, so it is naturally quite restricted and efficient.

We described and simulated 2-hop neighborhood variant of the protocol, with topological information. We already stated that 1-hop variant with positional information is very similar. Additionally, it can be further improved since geographic location, speed and direction of movements can be used to estimate how long each link will last [16, 18]. When message arrives, node can then estimate current neighborhood in a more precise way, and make better retransmission decisions.

We advocate for the development of further parameterless protocols. Some examples for geographic position based routing were given in [15]. It is our belief that threshold based approaches are generally inefficient. For example, typical algorithms normally call for message dropping when thresholds are violated. However, the purpose of communication protocol is not to drop but to deliver packet, making best possible effort to do so. Thus best available links and routes need to be selected among available ones.

ACKNOWLEDGMENT

This article has been supported by NSERC CRDPJ 319848-04 and UK RSWRMA grants.

REFERENCES

[1] A. Amis, R. Prakash, T. Vuong, and D. Huynh, “Max-min D-cluster formation in wireless ad hoc networks”, in *Proceedings of the IEEE Infocom*, Anchorage, Alaska, April 2001.

[2] J. Broch, D. A. Maltz, D. B. Johnson, Y. Hu, J. Jetcheva, “A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols”, ACM *MobiCom*, 1998.

[3] Cooper, D.E., Ezhilchelvan, P. and Mitrani, I., “High Coverage Broadcasting for Mobile Ad Hoc Networks”, *IFIP-TC6 Networking Conference*, Athens, Greece, May 2004, Lecture Notes in Computer Science Volume 3042, 100-111, 2004.

[4] J. Carle and D. Simplot-Ryl, “Energy efficient area monitoring for sensor networks”, *IEEE Computer Magazine*, 37, 2, 40-46, 2004.

[5] F. Dai and J. Wu, “ Distributed dominant pruning in ad hoc networks”, *IEEE Int. Conf. Communications ICC,* Anchorage, Alaska, 2003.

[6] S. Guha and S. Khuller, Approximation algorithms for connected dominating sets, *Algorithmica*, 20(4): 374 – 387,1998.

[7] J. Hahner, C. Becker, K. Rothermel, “A protocol for data dissemination in frequently partitioned mobile ad hoc networks”, *IEEE Int. Symp. Computers and Communications ISCC,* Turkey, July 2003, 633-640.

[8] François Ingelrest, David Simplot-Ryl, Hong Guo and Ivan Stojmenovic, Broadcasting in ad hoc and sensor networks, in: *The Handbook of Computer Networks,* (H. Bidgoli, EIC), Wiley, to appear.

[9] W. Lou and J. Wu, “On reducing broadcast redundancy in ad hoc wireless networks”, *IEEE Trans. on Mobile Computing,* 1, 2, 111-122.

[10] S. Ni, Y. Tseng, Y. Chen, and J. Sheu, “The broadcast storm problem in a mobile ad hoc network”, in *Proc. ACM Int.Conference on Mobile Computing and Networking (MobiCom)*, Seattle, USA, August 1999.

[11] Regina O’Dell, Roger Wattenhofer, “Information Dissemination in Highly Dynamic Graphs”, *ACM DIALMPOMC’ 05,* September 2, 2005, Cologne, Germany.

[12] W. Peng and X. Lu, “ On the reduction of broadcast redundancy in mobile ad hoc networks”, ACM International Symposium on Mobile Ad Hoc Networking and Computing *MobiHoc,* Boston, USA, 2000.

[13] Edwin Soedarmadji, Robert J. McEliece, “A Dynamic Graph Algorithm for the Highly Dynamic Network Problem”, Proc *FAWN at IEEE PerCom,* March 2006.

[14] D. Simplot-Ryl, I. Stojmenovic, and J. Wu, Energy efficient backbone construction, broadcasting, and area coverage in sensor networks, in: *Handbook of Sensor Networks: Algorithms and Architectures* (I. Stojmenovic, ed.), Wiley, 2005, 343-379.

[15] I. Stojmenovic, Localized network layer protocols in sensor networks based on optimizing cost over progress ratio, *IEEE Network,* Vol. 20, No. 1, January/February 2006, 21-27.

[16] I. Stojmenovic, M. Russell, and B. Vukojevic, Depth first search and location based localized routing and QoS routing in wireless networks, *IEEE International Conference on Parallel Processing,* August 21-24, 2000, Toronto, 173-180.

[17] I. Stojmenovic and J. Wu, Broadcasting and activity scheduling in ad hoc networks, in: *Mobile Ad Hoc Networking* (S. Basagni, M. Conti, S. Giordano and I. Stojmenovic, eds.), IEEE Press, 2004, 205-229.

[18] W. Su, S. J. Lee, and M. Gerla, “Mobility prediction and routing in ad hoc wireless networks,” *International Journal of Network Management*, vol. 6, no. 4, pp. 351–360, January 2001.

[19] I. Stojmenovic, M. Seddigh, J. Zunic, “Dominating sets and neighbor elimination-based broadcasting algorithms in wireless networks”, *IEEE Transactions on Parallel and Distributed Systems,* 13(1): 14 – 25, 2001.

[20] I. Stojmenovic, Comments and corrections to "Dominating Sets and Neighbor Elimination-Based Broadcasting Algorithms in Wireless Networks", *IEEE Trans. Par. Distr. Systems,* 15, 11, 2004, 1054-1055.

[21] K. Viswanath and K. Obraczka, “An adaptive approach to group communications in multi-hop ad hoc networks”, *International Conference on Networking (ICN 2002)*, Atlanta, 2002; *IEEE Int. Symp. Computers and Communications ISCC*, Taormina, Italy, 2002, 559-566.

[22] K. Viswanath, Adaptive routing for group communications in multi-hop ad hoc networks, *Ph.D. thesis,* Computer Engineering, Univ. California, Santa Cruz, June 2005.

[23] J. Wu and H. Li, A dominating set based routing scheme in ad hoc wireless networks, *Proc. DIAL M,* Seattle, Aug. 1999, 7-14; *Telecommunication Systems,* 18, 1-2, 2001, 13-36.

[24] H. Zhang and Z.P. Jiang, “Performance evaluation and improvement of broadcasting algorithms in highly mobile ad hoc networks”, Workshop on *Performance of Wireless Networks and Communication Systems (WiNCS),* Philadelphia, PA, July 23-28, 2005.

[25] B.D. Walker, J.K. Glenn, T.C. Clancy, Analysis of simple counting protocols for delay-tolerant networks, ACM CHANTS Workshop, at MOBICOM, Montreal, September 2007, 19-26.