PriorityCast: Efficient and Time-Critical Decision Making in First Responder Ad-Hoc Networks

Vartika Bhandari* and Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
vbhandar@uiuc.edu indy@cs.uiuc.edu

***Abstract*— First responders equipped with PDAs in disaster recovery scenarios may need to make decisions that select the best out of multiple options. Each of these options may originate from different parts of the network (e.g., risk assessment of various entrances to a collapsed building). Yet the highest priority option needs to be spread to all nodes within a *time deadline* and in an *efficient manner*, i.e., by preventing lower priority options from spreading too far. We define this problem as *PriorityCast*, and study a wide range of possible solutions. The emphasis is on solutions that are scalable, and resilient to unreliability and unpredictability. Our basic scheme, called Flood and Suppress, suppresses lower priority options from being forwarded by a node that has seen other higher priority options. We then augment this basic scheme using *probabilistic approaches*, *background gossiping techniques*, and *delayed propagation*. We quantify the impact of using various combinations of the mentioned approaches, by presenting mathematical analysis for the Flood and Suppress protocol, and evaluating a suite of composable PriorityCast protocols via simulations. The results provide insight into the feasibility and scalability of this class of solutions. While the focus of this paper is on the PriorityCast problem, these solutions are also relevant as potential building blocks for other distributed operations in ad-hoc networks.**

I. Introduction

This paper addresses a realistic ad-hoc networking problem, called *PriorityCast*, that arises out of a *large-scale project*[1] aimed at building an infrastructure for disaster recovery scenarios. In our infrastructure, we envision that relief and rescue operations in large disaster sites will involve a very large number (potentially hundreds) of *first responders* such as fire-fighters, police officers, medical personnel and engineers, who will be equipped with wireless-enabled communication devices (e.g., PDAs).

The first responders will use the ad-hoc network set up among these PDAs to disseminate, locate, and share information, and to collaboratively make *time-critical* decisions for selection of appropriate courses of action. It is very likely that such decisions will need to be made many times in the course of the entire rescue effort. Thus it is desirable that the dissemination be supported in an efficient manner, while maintaining a high degree of reliability even when individual nodes may behave unpredictably. This paper addresses the issue.

Returning to our example, consider a group of first responders trying to decide which is the best entrance to enter a collapsed structure. This decision is made from among several *options* that may be available. However, each of these options is seen and evaluated by geographically distant first responders (each potentially standing right in front of the entrance). Each option is assigned a score or *priority* by its evaluator. The problem is to design a *message-efficient protocol* that will disseminate the highest-priority option to all the nodes in the ad-hoc network, *within a time deadline*. We term this problem as *PriorityCast*. While similar to problems such as leader-election, and maximum-finding in networks, there is a novel aspect to the problem, in terms of the requirement that the solution scale well, make minimal assumptions about coordination between nodes, and disseminate information within the deadline. A naive solution involves sim-

[1]This work was partly supported by National Science Foundation Grant ITR-0427089: "IT-Based Collaboration Framework for Preparing Against, Responding to, and Recovering from Disasters Involving Critical Physical Infrastructures".
*Also affiliated with Coordinated Science Laboratory (UIUC).

1-4244-0507-6/06/$20.00 ©2006 IEEE

ply flooding each of the options throughout the system - this is prohibitive due to the high message overhead. It also increases network congestion, and can give rise to an acute form of the *Broadcast Storm* problem [1]. In this paper we consider a range of simple yet effective solutions to address this issue. We present a continuum approximation based analysis for a simple Flood-and-Suppress approach, and evaluate a number of alternative solutions via simulation, including approaches based on delayed propagation. Though the techniques comprising these solutions have individually been used in miscellaneous different contexts before, in this paper we treat them as a consolidated suite of protocols that allow for adaptation to different situations, and provide exhaustive evaluation of their performance for a newly emerging and important usage scenario.

While this work was motivated by the need for scalable protocols that require minimum/no prior coordination in a first responder network, the solutions studied in this paper are relevant to the general problem of disseminating information in a wireless ad-hoc network.

II. Problem Statement

PriorityCast Problem: Consider an ad-hoc network of n nodes $V = \{a_1, a_2, ..., a_n\}$. A random subset of V, of size k, generates values $v_1, v_2, ..., v_k$ respectively, where each value is assigned a unique and fixed *priority* by its generator node (source node). Without loss of generality, we assume that the values are in non-increasing order of rank, i.e. $rank(i) \geq rank(i+1)$, or equivalently a lower subscript value indicates a higher rank (or possibly an equal rank in case of consecutive subscripts). Denote the option *generators* as $S = \{s_1, s_2, ..., s_k\}$ respectively.

The PriorityCast problem is to (1) disseminate the highest priority option to all nodes, (2) to do so within a specified time bound of T seconds after the options are generated, and (3) to do so in a message-efficient manner, i.e., by preventing other lower priority options from spreading too far. For simplicity, we will assume henceforth that all options are generated simultaneously, i.e., at the same instant of time. The presented solutions work in the case of staggered option generation too. Requirement (1) is not very rigid in that it may be acceptable if some nodes receive a close-to-best option, rather than the best option.

System Model: We assume a typical ad-hoc network scenario, where each node can broadcast a message to each of its immediate neighbors through the wireless medium. We do not assume any *a priori* knowledge of the network size/topology (although an estimated upper bound on network diameter might be available), or fine-grained synchronization. Since deadline T is $O(seconds)$, we assume the presence of a coarse grained clock synchronization at nodes - this is completely orthogonal to the protocol.

Information Semantics : We assume that there is a global and total order on priorities of options. If each node were to use a different metric for comparing options, then it would be impossible for a node to estimate whether a certain value is relevant to another node or not. In this circumstance, the only way to ensure that each node receives *its* highest priority option is to simply flood all options. Hence, we focus only on the case where the ordering relation among all options is globally consistent. Although we assume a total order in this paper, our protocols generalize to partial orders too. In the latter case, a tie between two unordered values can be broken using arbitrary criteria such as comparison of generator node's IP addresses.

Note that in some cases, geographical proximity may make an option attractive even though it may not have the highest priority. Since options that originate nearby have a higher chance of being seen despite suppression, this concern may be accomodated by allowing a node to ultimately choose between the highest ranked value it received, and other comparatively lower-ranked values that originated nearby and were seen by it. Where proximity is really crucial, another alternative is to have TTL-restricted value propagation within a PriorityCast instance. Thus a node would decide on the highest priority option within a certain locality of itself.

III. Related Work

The *PriorityCast* problem is similar in spirit to the leader-election problem, where a set of participants must agree on a *single* leader from amongst themselves. Leader-election has been extensively studied in the context of distributed algorithms [2]. Algorithms for leader-election in mobile ad hoc networks are proposed in [3]. The PriorityCast problem is also closely related to work on

computation of aggregate information, e.g., maximum, minimum, average etc.There has recently been significant interest in aggregate computation in sensor networks where energy is a major concern. A deterministic method for continuous aggregate monitoring called *digest diffusion* is proposed in [4] wherein values are piggybacked on periodic network/MAC layer beacons. There is no notion of deadlines.

Though determination of a supremum in PriorityCast can be mapped to leader or maximum determination, the specific context of this problem leads to a different (and more relaxed) set of requirements. In PriorityCast, each node is required to choose a supremum by a certain deadline, and while it is desired that nodes choose the best value, it is tolerable if most nodes choose a best or close-to-best value, and only a few nodes make a bad choice. Besides, multiple generator nodes may originate values with the same rank, and nodes may choose any one of these values (tagged with the corresponding originator) without violating the problem requirements (contrast this to leader election, where each participant must choose exactly the same leader node). These relaxed requirements allow one to consider simpler solutions that lead to near-optimal choices by most nodes, and exhibit exceptionally good scaling behavior. Besides, it is desirable that the solution(s) make minimum assumptions about availability of individual nodes, and avoid relying on establishment of any dissemination structure, since there may be a high degree of unpredictability in a disaster scenario.

Epidemics-based algorithms have emerged as a powerful paradigm in distributed systems research, due to their ability to achieve scalable communication with probabilistic reliability guarantees. These algorithms are rooted in epidemiological principles pertaining to the spread of infectious diseases. The use of probabilistic broadcast in wireless ad hoc networks was proposed in [1], where it was aimed at alleviating the broadcast storm problem. Amongst other epidemic protocols, a gossip-based approach toward balancing energy consumption and latency for broadcasts when nodes use a power-save protocol has been presented in [5].

Analysis of efficient gossip-based algorithms for aggregate computation is presented in [6]. Of particular interest is the *Distributed Find* algorithm for finding the ϕ-th order statistic, that converges to the correct value in a logarithmic number of rounds with high probability. However, a designated leader and synchronized leader-initiated phases are assumed in the decentralized algorithm. In contrast, we seek much simpler solutions that do not rely on any coordination amongst participants.

Another relevant notion is that of probabilistic *semantic reliability* which has been considered in the context of multicast protocols [7]. In [7], it is proposed that several messages get superseded by more recent semantically dominant ones while they are yet to be delivered. Such messages should be detected and purged to alleviate network congestion (semantic purging). Some recent work in the domain of energy-efficient routing [8] also uses a notion of delayed forwarding akin to ours.

IV. Analysis of a Pure Semantic Purging based Flood-and-Suppress Protocol

We present a simple mathematical model for the spread of information in a network scenario of the kind described in Section II, when using an unoptimized protocol using only *semantic purging* (referred to in Section I, and in the following sections, as the basic Flood-and-Suppress protocol).

Consider the network model described in Section II, viz., the network comprises n nodes $a_1, a_2, ..., a_n$, and a random subset of these nodes of size k originates values $v_1, v_2, ..., v_k$. A lower subscript value indicates a higher rank. The option generator nodes are $S = \{s_1, s_2, ..., s_k\}$ respectively. Recall that the Flood-and-Suppress protocol is based on the notion of *semantic purging* whereby if a_i receives v_j after having received some v_l s.t. $l < j$, then it suppresses v_j.

We present a simple *continuum* propagation model wherein there are k generator points $s_1, s_2, ..., s_k$ that originate information, and the information propagates out in 2-D space from a generator till it meets the frontier of a higher ranked value, at which point, the lower priority value gets suppressed. We ignore any edge effects, i.e., the network area may be considered toroidal. The generator locations are uniformly distributed over the network area, and the generator location and generated value are uncorrelated. This information spread pattern may be viewed as a multi-tiered variant of a Voronoi Growth Model [9]. We consider the case when all values are emitted at the same instant (call it $t = 0$) and propagate through space

at the same rate. This conforms to the standard definition of Voronoi polygons based on Euclidean distance. The continuum propagation process may be viewed as infinitesimal area elements receiving and re-propagating a value. Henceforth we shall refer to an area element as a site.

Theorem 1: When the number of generators is k, and generated values are uncorrelated with generator locations, $E[$ num values propagated by a site $]$ $=$ H_k, where H_k denotes the k-th harmonic number.

Proof: Consider a site P. We enumerate the generators in non-decreasing order of distance from *P* i.e. $s_{i_1}, s_{i_2}, ..., s_{i_k}$, and call it the distance order w.r.t. *P*. Then *P* will receive and propagate v_{i_j} iff $\forall m < j, rank(v_{i_m}) < rank(v_{i_j})$. Given that the generator locations are uniformly random and uncorrelated with generated value, each permutation of S is equally likely as the distance order w.r.t. *P*. Thus, $Pr[s_m = s_{i_j}] = \frac{1}{k}$, $\forall 1 \leq j \leq k$. Hence, we can determine the probability (over all generator configurations) that point *P* receives *and* propagates value v_m as:

$$Pr[P \text{ receives and propagates } v_m | s_m = s_{i_j}] = \begin{cases} \frac{^{k-m}P_{j-1}(k-j)!}{(k-1)!} & \text{if } j < (k-m+1), \\ 0 & else. \end{cases} \quad (1)$$

Therefore:

$$Pr[P \text{ receives and propagates } v_m] = \frac{1}{k!} \sum_{j=1}^{k-m+1} {}^{k-m}P_{j-1} \cdot (k-j)! = \frac{1}{m} \quad (2)$$

Thus $E[$ num values propagated by P $]$ $=$ $\sum_{m=1}^{k} \frac{1}{m} = H_k$. Since $\ln(k+1) < H_k < \ln(k) + 1$, the expected number of distinct values received by a point is $\Theta(\log k)$.

The same result is also obtainable via a Voronoi polygon argument. Denote by $V_i^{(j)}, (i < j \leq k)$ the Voronoi polygon of s_i in a tessellation having generator-set $s_1, ..., s_j$. The total area of the network is A. Then, the area that receives and propagates v_i is given by $Area(V_i^{(i)})$. Also $E[Area(V_i^{(j)})]$ is $\frac{A}{j}$. Thus, the expected area to which v_i propagates is $\frac{A}{i}$. The probability that a site P chosen at random lies within $V_i^{(i)}$, and therefore receives and propagates it is thus $\frac{1}{i}$. Thus $E[$ num values propagated by P $]$ $= \sum_{m=1}^{k} \frac{1}{m} = H_k$. ■

However, the expected value analysis does not reveal information about the actual distribution. We now state and prove a stronger claim:

Theorem 2: When the number of generators is k, and generated values are uncorrelated with generator locations, the number of values propagated by a given site P is $O(\log k)$ with probability at least $1 - \frac{1}{k}$.

Proof: Consider given site P. The generators now generate values uncorrelated with location (either absolute, or relative to P). Consider a permutation π representing the distance order of the generators w.r.t. P. Since the values are not correlated with location, any distance order is equally likely. Then $\pi(i)$ is the position of the generator s_i in the distance order. Evidently the number of values that may be propagated in this configuration $\leq$ $\pi(1)$. Let $\pi(1) = j$. Let us denote π as $i_1, i_2, ..., (i_j = 1), ..., i_k$. Then, consider the subsequence $i_1, i_2, ..., i_{j-1}$. If $\max(i_1, ..., i_{j-1}) = i_p$, then the number of values propagated $\leq p + 1$. The same argument may be applied recursively to the prefix sequence $i_1, ..., i_{p-1}$, and so on, till the recursion bottoms out. Each step of the recursion adds 1 to the number of values propagated. We thus need to prove that the depth of the recursion is $O(\log k)$ with probability at least $1 - \frac{1}{k}$. Observe that at the beginning of recursive step i, we have a prefix subsequence π_i of π, of some length l_i (where $l_1 = k$). The maximum element of π_i may occupy any of the l_i positions with equal probability $\frac{1}{l_i}$. If the maximum element occupies position j, $l_{i+1} = j - 1$. Thus the prefix subsequence under consideration is reduced in size. We call a recursive step i a *nice* reduction, if $l_{i+1} \leq \frac{1}{2} \cdot l_i$. Then, it may be seen that $\log_2 k$ nice reductions suffice for the recursion to bottom out. The probability that a particular recursive step i comprises a nice reduction is given by $p \geq \frac{1}{2}$. We consider a sequence of M steps in the recursion. We denote by X_i the outcome of each step i, where $X_i = 1$ if the step comprised a *nice* reduction, and is 0 otherwise. Then the X_is are $i.i.d.$ random variables with $Pr[X_i = 1] = p \geq \frac{1}{2}$. Let $X = X_1 + X_2 + ... + X_M$. Thus X denotes the number of nice reductions that occur in the

sequence of M steps, and $E[X] \geq \frac{M}{2}$. We seek to determine the probability that $X > \log_2 k$, as this would imply that the recursion bottoms out within M steps[2], and hence the number of propagated values $\leq M$. For this purpose, it is possible to make use of the following special form of the Chernoff bounds [10]:

$$Pr[X \leq (1-\beta)E[X]] \leq \exp(\frac{-\beta^2 E[X]}{2}), 0 < \beta < 1 \quad (3)$$

We set $M = 9 \ln k$ (note that $\log_2 k < \frac{3}{2} \ln k$), and $\beta = \frac{2}{3}$. Then, we can show that:

$$Pr[X \leq \frac{M}{6}] \leq \exp(\frac{-M}{9}) \quad (4)$$

Thus we obtain:

$$Pr[X \leq \frac{3}{2} \ln k] \leq \exp(-\ln k) \quad (5)$$

This leads to the result that:

$$Pr[\text{ site P propagates } \leq 9 \ln k \text{ values }] \geq 1 - \frac{1}{k} \quad (6)$$

Thus, the number of values propagated by the given site is $O(\log k)$ with probability at least $1 - \frac{1}{k}$. ■

In simulations (see Section VI), we find that this number tends to be much less than $9 \ln k$.

a) Realistic Networks: Realistically, the information does not spread in a continuum. Besides, the time for a value to reach a particular node depends not merely on the hop-distance but on the congestion and channel conditions encountered. However, the continuum model yields useful insights. We thus rest our analysis on such a continuum approximation, and investigate its validity via simulation. The analysis does not handle the receipt of duplicate copies of the same value from different neighbors. The attempt has been to quantify the number of uniques values propagated, and then reduce redundancy by applying solutions similar to those proposed for broadcast.

V. Proposed Solutions to PriorityCast

We have based our PriorityCast protocols on *wireless broadcast* rather than unicasts.

[2]Observe that the argument used by us is very similar to that invoked in algorithmic literature to prove that randomized quicksort runs on an input of size n in time $O(n \log n)$ w.h.p.

We present our protocol descriptions in terms of a number of building blocks from which various hybrid protocols may be composed, as per requirement:

- *Semantic Purging:* This is the most basic mechanism. In our context, *semantic purging* implies supression of lower priority values if a higher priority value has already been seen.
- *Re-propagation Delay:* To reduce traffic, one may stipulate that if a node receives a value v that is currently the supremum, it will re-propagate it instantly with probability p_{inst}, while with probability $1 - p_{inst}$ it will wait for time T_{defer}. Thus if a higher value is received while a node is waiting, the prior one(s) can be suppressed.
- *Parameter Adaptivity:* It is desirable to choose p_{inst} and T_{defer} adaptively so as to reduce overhead while still meeting the deadline. In fact, one could think of them as $p_{inst}(r_i, d_{res})$ and $T_{defer}(r_i, d_{res})$ where r_i is the number of values already received by node i before receiving the current value (observe that in the most general case, the probability of the current supremum being the global supremum is $\frac{1}{k - r_i + 1}$ i.e. increases with r_i), and d_{res} is the residual time to deadline.
- *Probabilistic Re-propagation:* Efficiency concerns dictate that protocols have a *probabilistic* component associated with re-broadcasting, to reduce redundant message delivery.
- *Background Gossip:* To ensure that reliability is not compromised, a lightweight *background gossip* component may also be added.

Protocols considered in this paper are composed of some or all of these mechanisms. The protocols incorporate a notion of deadlines. On deadline expiration the current local supremum is chosen as the decision value by each node. PriorityCast packets carry the value of interest, as well as auxiliary information such as the identity of the generator, and the number of hops traversed so far from the generator.

A. Flood-and-Suppress Protocols

The basic Flood-and-Suppress protocol (F&S) uses only semantic purging. Thus a node that receives a value discards it if its rank is lower than the current supremum, else sets it as its supremum and re-broadcasts it.

A probabilistic variant (pF&S) re-broadcasts only with a probability p_g to reduce redundancy. To maintain reliability, a background gossip component may be incorporated (pF&S+BG) whereby each node periodically (at intervals of T_{bg}) makes a probabilistic decision (with probability p_{bg}) to broadcast its current supremum value.

B. Delayed Propagation Protocols

In the basic Delayed Propagation (Dprop) protocol, a node that receives a value discards it if its rank is lower than the current supremum, else sets it as its supremum. It then sets a timer for time T_{defer}. If no higher-ranked value is received during this interval, the current supremum is broadcast on timer expiration. If a higher-ranked value is received, the running timer is canceled, and a new timer started. Thus the Dprop protocol corresponds to $p_{inst} = 0$.

A probabilistic version of this protocol (pDprop) re-broadcasts a value on timer expiration with probability p_g, rather than deterministically. To handle possible omissions that might hurt supremum quality, a background gossip mechanism may be added (pDprop+BG), as in case of the F&S protocol. Finally, we consider adaptive variants without/with background gossip (pDprop+A and pDprop+BG+A) that choose T_{defer} dynamically, and are hence expected to be better suited for meeting variable deadlines (as they can adapt to different time-scales), while keeping overhead low. For adaptive Dprop variants, we propose the following general formulation for T_{defer} to be chosen by a node at time $t < d$ (where d is the deadline):

$$T_{defer}(t) = c_1 \cdot \frac{d-t}{(D-h)} + c_2 \cdot \frac{1}{v_{seen}(t)} \cdot f(t) \quad (7)$$

where D is a maximum estimate of network diameter, h is the number of hops traversed by the current supremum so far, v_{seen} is the number of values seen by the node so far, and $f(t)$ is a decreasing function of time satisfying $f(d) = 0$. c_1 and c_2 are appropriately chosen constants (typically $c_1 < 1$). The intuition behind this formulation is at follows: at time t, if the value has already traversed h hops, it may need to traverse upto $D - h$ more hops in the remaining time interval of length $d-t$. Without introducing a re-propagation delay, the time taken for each hop traversal is given by queueing and propagation delay + transmission time. Assuming that this time is much smaller compared to the time-scale on which values need to propagate, one may add a delay of $c_1 \cdot \frac{d-t}{D-h} (c_1 < 1)$, while ensuring that the value is able to traverse all remaining $D - h$ hops within deadline. Besides, we bias the delay so that nodes that have seen very few values are more conservative about re-propagating. This yields the second term. The function $f(t)$ ensures that as the deadline approaches, this bias becomes increasingly smaller. A propagation decision for time t_s actually gets scheduled for time $t_s + t_{jitter}$ where $t_{jitter} \in [0, JITTER_{max}]$. This helps prevent synchronization of the transmissions of various propagating nodes, that might lead to collisions and resultant high packet loss.

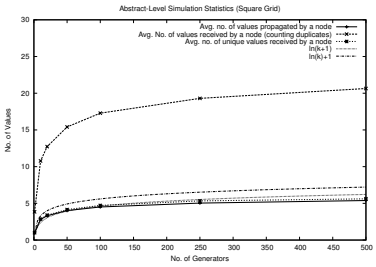

Fig. 1. Statistics from Micro-Benchmarks (Square Grid)

VI. Experimental Results

Our evaluation approach has been two-pronged. We have performed some abstract-level simulations to corroborate the analysis presented in Section IV. To evaluate the protocols we propose for PriorityCast, we have undertaken simulations using the ns-2 [11] simulator. In these simulations, we avoid taking into account the actual semantics of the information that must be PriorityCast by having generators choose $pseudovalues$ that indicate the rank of the item within the global total order.

A. Micro-Benchmarks

In this section, we present results from abstract-level simulations, that assume an ideal physical and MAC layer, and are primarily aimed at validating the analysis. They have been performed on a 25x25 square lattice, with all sites occupied. This corresponds to a network of 625 nodes, where each node is capable of communicating with its four lattice neighbors. k nodes are chosen uniformly at random as generators, and they choose a pseudovalue at random to propagate.

We study the use of a simple Flood-and-Suppress methodology (analyzed in Section IV). The average statistics and the distributions have been computed over 50 runs for each value of k. Fig. 1 depicts the average statistics, as the number of generators is varied from 1 to 500. The average number of values propagated by nodes when the total number of generated values is k, tends to stay within the lower and upper bounds for H_k. The number of unique values received is marginally larger, with the difference not being significant. However the total number of values (including duplicates) received is *significantly* larger, indicating a strong need to reduce redundant broadcasts. The key observation is that the expected value analysis of Section IV seems to have been borne out. It was also found in these simulations that the maximum number of unique values received by any node was well within the $9 \ln k$ mark.

B. Results from ns-2 Simulations

We have implemented PriorityCast in the ns-2 network simulator. The ns-2 shadowing model is used with path-loss exponent set to 3, and the shadowing deviation set to 4 dB. In the simulations, we assume the IEEE 802.11 MAC, and an AODV [12] routing layer. The transmission rate is 2 Mbps, the transmission power is 100 mW, and the receiver sensitivity is -91 dBm. We set the carrier sense threshold to -102 dBm. Currently the routing protocol is not used by the PriorityCast service, which relies exclusively on *hop-by-hop broadcasts*. We have implemented and evaluated a suite of protocols comprising the variants described in Section V. We present results for evaluation of static scenarios comprising 400 nodes in a 2500m x 2500m area. All scenarios were generated using the BonnMotion tool [13]. The total simulation time is 50.0s, and in each simulation, the PriorityCast protocol is invoked once. All presented statistics represent an average over 10 random static topologies. We also introduce very low background traffic comprising 20 constant bit rate (CBR) conversations (between randomly selected source-destination pairs, and starting at random times) [3]. In the presented simulation results, the gossip probability p_g was set to 0.5, and the background gossip probability was 0.2 (wherever applicable). The number of generators k was varied in the range 40-200. In each simulation, node numbers $1, ..., k$ were designated originators, and as each node is placed uniformly at random, this yields random distribution of generators. Recall that all simulations used the same 10 random static topologies. Thus, varying k corresponds to incrementally making additional nodes in the topology originate values. All generators originate the values at the same time. The deadline for dissemination was set so as to allow 10.0s for propagation. For non-adaptive Dprop variants, T_{defer} was set to 1.0s. Adaptive Dprop variants choose $T_{defer}(t) = 0.5 \cdot \frac{d-t}{(D-h)} + 2 \cdot \frac{1}{v_{seen}} \cdot (1 - \frac{t}{d})$ (recall notation from Section V). v_{seen} is taken as the total number of values received, rather than the unique ones. However, as the two are expected to be proportional, this is reasonable. T_{bg} is set to 5.0s, while $JITTER_{max}$ is 100ms. In order to quantify satisfactoriness of the chosen suprema, we examine the number of nodes n_i that choose the value with rank i on deadline expiration.

1) Static Networks: No Spatial Correlation in Values: We discuss the simulation results for static network topologies, when the values generated do not show any location correlation. Fig. 2 depicts the average number of values that a node propagates. It can be seen that for the F&S protocol, this value lies within the bounds for H_k, thus bearing out the analysis of Section IV. The basic Dprop protocol propagates less values than F&S on average, and all protocols with a probabilistic component propagate a very small number of values, with the probabilistic Dprop variants propagating the least. Fig. 3 depicts the average number of values received by a node (counting duplicates). As is expected, F&S receives the maximum number of values, followed by Dprop. The probabilistic protocols show significant improvement, and receive a much smaller number of values. The probabilistic variants of F&S show moderate improvement. pDprop, pDprop+A, pDprop+BG and pDprop+BG+A exhibit a very large reduction.

We have not included graphs for number of unique values received due to lack of space but briefly discuss them. The average number of unique values received by a node is higher than the number of values propagated. However the P.D.F. of unique

[3]We did not study performance of the background CBR flows, but did observe extremely poor CBR delivery ratios, primarily due to failed route discoveries. This happened even in absence of PriorityCast activity, and is thus orthogonal to our work, but merits mention as it raises a relevant issue - routing protocol performance in large, dense networks

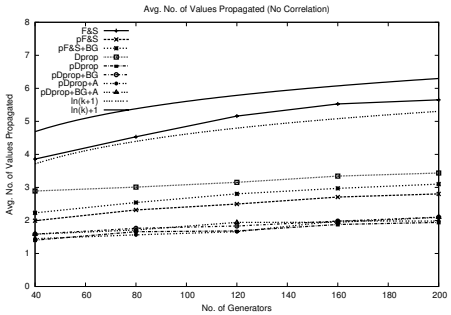

Fig. 2. Avg. No. of Values Propagated (No Correlation)

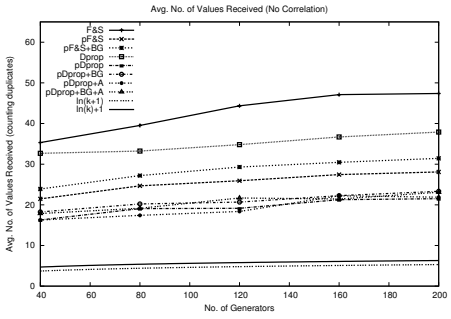

Fig. 3. Avg. No. of Values Received incl. Duplicates (No Correlation)

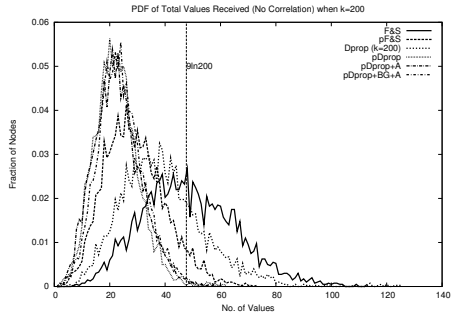

Fig. 4. P.D.F. of No. of Values (incl. duplicates) Received (No Correlation)

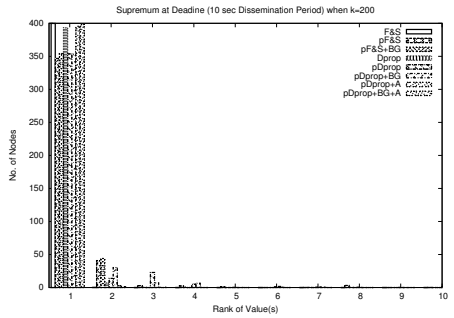

Fig. 5. Rank of Chosen Supremum (No Correlation)

values is still strongly concentrated for all protocols, and most of the probability mass lies below the $9 \ln k$ mark.

Fig. 4 shows the P.D.F. for total number of values received (counting duplicates) when $k = 200$. For clarity, we only reproduce results for six protocols. The F&S protocol has the heaviest tail. In Dprop, the mass is shifted to the left, while the probabilistic variants of both F&S and Dprop show even greater left-shift. The pDprop, pDprop+A, and pDprop+BG+A protocol(s) are the best in terms of this criterion.

The emergent picture is that Dprop is more message-efficient than F&S, and adding a probabilistic component reduces the total number of values received. However, there is not much additional reduction in message overhead obtainable via further sophisticated approaches to pDprop. Nevertheless, the case for background gossip and adaptive Dprop rests on another crucial consideration viz. the quality of the supremum that gets chosen by each node on deadline expiration. While addition of probabilistic rebroadcast reduces the message overhead, it increases the possibility of nodes not receiving the global supremum, due to the *gossip* for that value dying out. Similarly, use of a fixed T_{defer} in Dprop may make it difficult to meet tight deadlines. Adaptive T_{defer} also allows for greater opportunity to supress values when there is a lot of time left before deadline expiration. We consider statistics regarding the number of nodes that chose values of rank 1, 2, 3, ... etc. as their supremum on deadline expiration (averaged over 10 runs). Recall that rank is determined by the pseudovalue. Fig. 5 depicts these statistics (a few outlying points are not shown in this figure to keep the scale reasonable) for $k = 200$. The x-axis indicates the rank of the value, and the y-axis indicates the number of nodes that chose it as supremum. All protocols show fairly good quality of suprema. Though there is not a significant performance difference between various protocols here, there is indication that any possible loss of supremum quality on using pDprop can be potentially compensated by using the additional mechanisms we have considered. There is some variability in the trends for other values of k (not shown here). However, one general conclusion can be derived, viz., that any loss in supremum quality by using pDprop can be compensated for by adding both adaptivity and background gossip (i.e. using

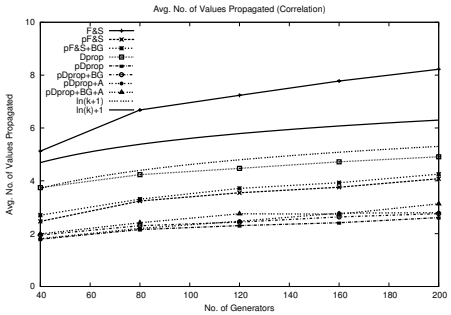

Fig. 6. Avg. No. of Values Propagated (Correlated Values)

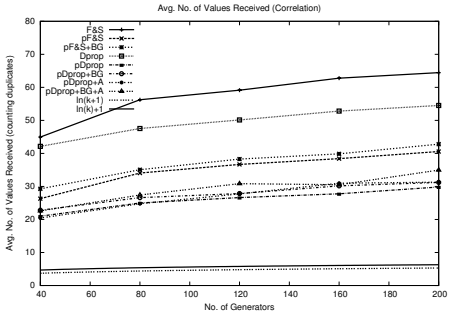

Fig. 7. Avg. No. of Values Received incl. Duplicates (Correlated Values)

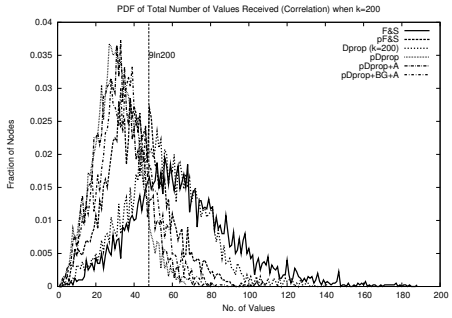

Fig. 8. P.D.F. of No. of Values (incl. duplicates) Received (Correlated Values)

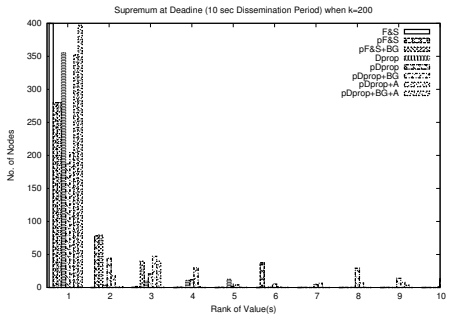

Fig. 9. Rank of Chosen Supremum (Correlated Values)

pDprop+BG+A); thus leading to low overhead as well as good quality.

2) Static Networks: Spatial Correlation in Values: To investigate the effect of spatial correlation in values, we have performed another set of simulations. We consider the same set of static scenarios as in Section VI-B.1. The generator nodes are chosen as before. However, in this set of simulations, a generator node located at (x, y) originates a *pseudovalue* $= \lfloor x + y \rfloor$. This yields a strong physical gradient in the generated values, and leads to the likelihood that higher priority values will be generated by nodes that are close together.

Fig. 6 depicts the average number of values that a node propagates. This number is only marginally higher than in the absence of correlation. Particularly, the number of values propagated by F&S is only slightly higher than H_k. Thus, it appears that the expected value analysis based on lack of correlation is not too far off even when there is strong spatial correlation. Fig. 7 shows the average number of values a node receives (counting duplicates). While the relative trends again remain the same, it is noteworthy that the number of values received for each protocol is higher in the presence of correlation. This is intuitive, as the higher priority values originate from certain specific regions of the network, and lower priority values are often able to propagate further before getting suppressed. Similar trends are observed for the average number of unique values received, which is higher when there is correlation (figure omitted due to space constraints). Due to lack of space, we have also not included the P.D.F. of unique values received. It also exhibits a right-shift in probability mass compared to the no-correlation case. However, most of the probability mass is still much below the $9 \ln k$ mark. Fig. 8 shows the distribution for the total number of values received with $k = 200$. For the sake of clarity, we only reproduce results for six protocols. F&S exhibits the heaviest tail, very closely followed by Dprop. pF&S performs much better, but pDprop and pDprop+A are best. pDprop+BG+A shows only a marginal right-shift compared to pDprop and pDprop+BG, and is still quite message-efficient.

Fig. 9 depicts the number of nodes that chose a particular ranked value as their supremum when $k = 200$ (some outlying points are not shown in the figure due to scale issues). F&S per-

forms best. pDprop+BG+A is next-best. Dprop and pDprop+A also perform quite well. However, pF&S and non-adaptive pDprop variants have comparatively poorer performance. pDprop+BG+A peforms somewhat better than pDprop+A signifying that the background gossip component may help. Overall, the quality of the achieved suprema is worse in presence of correlation when augmented variants of the basic F&S and Dprop protocols are used. Once again, the supremum statistics for other values of k do show some variability in trends; however the same general conclusion emerges from all, i.e., adding both adaptivity and background gossip to pDprop allows for achieving a reasonable trade-off between efficient dissemination and good supremum quality.

VII. Discussion

The results obtained from the ns-2 simulations conform well to the analysis and the micro-benchmarks. The PriorityCast problem has an inherent scalability in that even with a trivial F&S protocol the number of unique values propagated tends to scale logarithmically. However, inclusion of a gossip component is important for reduction in redundant receipts. Delayed Propagation further reduces communication overhead by providing greater opportunity for semantic purging. Additional optimizations can help reduce message overhead without overtly degrading quality of chosen suprema. The benefits of having a suite of protocols using different combinations of the considered mechanisms lie in the ability to choose the appropriate combination for a given situation. More specifically, the considered protocols have flexibility to handle a much wider range of situations/requirements than those analyzed/simulated in this paper. Adaptive T_{defer} allows modulation of dissemination speed to suit a wide-range of timescales (depending on deadline), thereby allowing effective suppression without degrading supremum quality. It is also expected to be useful in case of staggered value generation, since values generated earlier will be propagated after larger delays, thereby allowing opportunity for their suppression by higher-ranked values generated later. Background gossip not only helps compensate for supremum degradation due to probabilistic propagation, but can also be effective in overcoming missed packets when nodes use power-save mechanisms. Inclusion of auxiliary information allows nodes the freedom to choose a supremum based on criteria involving both rank and other desirable properties.

VIII. Conclusion

We have considered an important problem called PriorityCast, that seeks to disseminate critical prioritized information in a wireless ad hoc network, subject to deadlines, and in an efficient manner. We have presented some analysis based on a continuum-propagation approximation, and evaluated a number of solutions via simulation. Overall, the results are extremely promising, and indicate that simple but effective gossip-based protocols are a suitable solution for the PriorityCast problem.

Acknowledgements

The authors thank Cigdem Sengul and Matthew Miller for useful comments.

References

[1] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen, and J.-P. Sheu, "The broadcast storm problem in a mobile ad hoc network," in *Proc. of Mobicom '99*, 1999, pp. 151–162.

[2] H. Attiya and J. Welch, *Distributed Computing*. McGraw-Hill, 1998.

[3] N. Malpani, J. L. Welch, and N. Vaidya, "Leader election algorithms for mobile ad hoc networks," in *Proc. of DIALM '00*, 2000, pp. 96–103.

[4] J. Zhao, R. Govindan, and D. Estrin, "Computing aggregates for monitoring wireless sensor networks," in *Proc. of SNPA 2003*, 2003.

[5] M. J. Miller, C. Sengul, and I. Gupta, "Exploring the energy-latency trade-off for broadcasts in energy-saving sensor networks." in *Proc. of ICDCS 2005*, 2005, pp. 17–26.

[6] D. Kempe, A. Dobra, and J. Gehrke, "Gossip-based computation of aggregate information," in *Proc. of IEEE FOCS*. IEEE Computer Society, 2003, p. 482.

[7] J. Pereira, L. Rodrigues, R. Oliveira, and A.-M. Kermarrec, "Probabilistic semantically reliable multicast," in *Proc. of the IEEE NCA '01*, 2001.

[8] S. Gobriel, D. Moss, and R. Melhem, "Mitigating the flooding waves problem in energy-efficient routing for manets," in *Proc. of ICDCS 2006*, 2006.

[9] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: Concepts and applications of Voronoi diagrams*, 2nd ed., ser. Probability and Statistics. NYC: Wiley, 2000, 671 pages.

[10] M. Mitzenmacher and E. Upfal, *Probability and computing*. Cambridge University Press, 2005.

[11] Information Sciences Institute, "NS-2 network simulator," Version 2.27, 2003.

[12] C. E. Perkins, E. M. Royer, and S. R. Das, "Ad hoc on demand distance vector (AODV) routing," IETF, Internet-Draft Version 13, February 2003.

[13] "Bonnmotion: A mobility scenario generation and analysis tool," University of Bonn.