Enabling reliable many-to-many communication in ad-hoc pervasive environments

Einar Vollset and Paul Ezhilchelvan
School of Computing Science
University of Newcastle
{einar.vollset, paul.ezhilchelvan}@ncl.ac.uk

Abstract

In order for pervasive computing to realize its full potential, pervasive applications have to be able to operate without support from fixed communication infrastructure at least some of the time. Sophisticated applications emerging in this domain will have to rely on the cooperation of groups of wireless devices in a peer-to-peer (P2P) fashion to accomplish their task. A number of such applications are likely to require coordination between devices. Such coordination requires deterministic reliability guarantees from the communication protocol in order to be efficient. In this paper we present a novel P2P many-to-many communication protocol which provides such deterministic guarantees. Our approach differs radically from previously proposed deterministic protocols in that it does not rely on routing structures. This allows the protocol to provide its guarantees under a wide range of network conditions and in an efficient manner.

1 Introduction

Managing applications in an infrastructure-less ad-hoc environment is exceptionally hard due to low network bandwidth, dynamic network topologies, and limited device capabilities. The task of building sophisticated applications (e.g., replica management for highly available content distribution) is simplified if certain fundamental problems that are commonly encountered are solved efficiently at the network level rather than at the application level. Solutions at this low-level are commonly called *middleware* solutions in the literature (see Bacon *et. al.* [18]). This paper presents an efficient *many-to-many dissemination* protocol, which can form the basis of such middleware solutions.

A number of many-to-many dissemination protocols have been proposed for ad-hoc networks. These can be broadly divided into two categories, *deterministic* and *best-effort*, depending on the type of delivery guarantees they provide. The best-effort category includes optimized[10, 9][1] and probabilistic [1, 11, 2] protocols in addition to the better known best-effort protocols of [8, 14]. Best-effort protocols deliver a message to all concerned devices in a group with a high probability. Deterministic protocols on the other hand provide absolute guarantees on message delivery. Proposed protocols in this category includes both multicast [15, 5, 6] and broadcast[13]. In this paper we present a deterministically reliable protocol, called *Scribble*, which guarantees that a message gets delivered to at least k devices in a group. Scribble's design is novel in that unlike the existing deterministic protocols, it does not rely on routing structures, and thus can operate in a very wide range of network conditions. In the limit, the only property the network must satisfy is that it is not permanently partitioned (that is, any partition that occurs should be transient and must heal within some finite but unknown time). The implication of this is that if Scribble cannot operate under a given set of network conditions, no deterministic protocol will. Our radical approach to designing Scribble, in particular the absence of routing structures, naturally raises concern about bandwidth overhead. The paper alleviates such concerns through representative simulations. A further analysis of the design of existing, structure-based protocols, found in section 3, and the experience of structure-less protocols in [12] also suggest that such concerns are misplaced, and in fact that a structure-less approach might be preferable in a number of scenarios.

2 Protocol Definition and Assumptions

A group $\mathcal{G}$ consists of an arbitrary number n of devices, N_0, N_1, ..., N_{n-1}, which communicate using an omni-directional wireless transmission mechanism (such as 802.11b). Any device in $\mathcal{G}$, say N_0, can initiate at any time, say t_0, the dissemination of a data packet, denoted as m, to

[1] These protocols are sometimes called reliable.

a specified number of devices, denoted as $k, 1 < k \leq n$. A deterministically reliable many-to-many dissemination protocol satisfies the following three properties:

(i) Integrity: a device that delivers *m* (to a higher level application) delivers it exactly once,

(ii) Termination: at least k devices in $\mathcal{G}$ deliver *m* within some bounded time after t_0, where *k* is a protocol parameter.

(iii) Network Subsidence: within some bounded time after t_0, transmissions of *m* or any packet related to the dissemination of m stop.

By these definitions, exactly-once delivery of a packet *m* is guaranteed to at least *k* devices within some bounded amount of time and with a bounded transmission overhead. If *k* = *n*, the protocol becomes a *broadcast* protocol.

2.1 Assumptions

A1. Each device has a unique ID that is not necessarily known to others.

A2. At least *k* devices in $\mathcal{G}$ exist for the duration of the protocol operation,

A1 is necessary to guarantee that at least *k* distinct devices have received *m*; it can be trivially met by most devices. The need for A2 is also intuitive; if *m* is to be delivered to *k* devices, $\mathcal{G}$ must contain at least *k* devices. Ensuring A2 is met requires that N_0 choose k appropriately, which requires some form of knowledge of group membership. Throughout this paper we assume that a this knowledge is provided to the protocol (as in [13]). Further, we note that the primitives provided by Scribble should enables this form of group membership service to be built in an efficient manner suitable for ad-hoc pervasive environments.

2.1.1 Network assumption

In the pervasive computing environment, due to user- or device mobility, interference and various obstacles or power-saving measures it is possible for a subset of devices in a group to get partitioned from the rest of the group at any moment in time, i.e. none of the devices in a given subset is able to communicate with any other device not in the subset. If partitions can be permanent, then clearly the termination property may not be met: if N_0 (that initiates the dissemination of *m*) and fewer than *k-1* other devices are in one subset, and if this subset is isolated permanently from any other subset, then no device in the latter subset can ever receive *m*, and no protocol can ever succeed. Therefore, the minimal requirement for solving the deterministically reliable many-to-many dissemination problem is that at least one subgroup that contains N_0 and at least *k-1* other devices of $\mathcal{G}$, must not suffer a *permanent* partition. We call this requirement the *minimal liveness property* and assume that the network satisfies this property.

3 Design

Any deterministically reliable many-to-many communication protocol, not just ours, must address the following three design issues:

1. *Message dissemination*: What mechanism is used to attempt to deliver *m* to enough devices?

2. *Coverage deduction*: Given that enough number of devices have received *m*, how is this fact deduced so that the protocol can terminate (c.f. network subsidence)?

3. *Protocol termination*: As per assumption A2, the desired coverage *is* possible and the protocol should not terminate until and unless that happens. What measures are in place to ensure that the protocol achieves the desired coverage?

Existing deterministically reliable protocols [13, 15, 5, 6] address the design issues 1 and 2 above by using a *routing structure* imposed on the network topology. The routing structure may be a tree[6][5], cluster[13] or a set of unicast routes[15]. Ho *et. al.* [7] observe that as the volatility of the network topology increases, the likelihood of a routing structure accurately reflecting the current topology diminishes. If the dissemination based on a routing structure fails to achieve the desired coverage, a deterministic protocol, unlike its best-effort counterparts, is obligated to take remedial actions for the on-going message dissemination. This involves the routing structure being patched (as in [6]) or being recreated if patches are deemed not effective. These efforts can be expensive in terms of bandwidth usage.

More seriously, however, most existing reliable protocols rely on the same structure for coverage deduction as well. This is typically in the form of aggregated acknowledgments being sent back up the structure to the originator node or some statically chosen "core node" as in [6]. This naturally implies that the structure needs to remain valid much longer than if it were used only for message dissemination. The structure is more likely to break during coverage deduction, as this happens after the message dissemination. If it does fail and if patches do not work, the structure needs to be recreated or acknowledgments have to be routed (or flooded) back to the originating or core devices so that the protocol can terminate.

The apparent high cost of remedial actions, and the high likelihood of them being applied often, led us to take a different, *structure-less* approach to 1 and 2, which entirely removes any dependence on fragile routing structures. In our protocol, the responsibility for message dissemination is passed on from one device to another, beginning with the initiator N_0, in a decentralized manner. The resulting arrangement keeps the transmission overhead small while yielding high coverage. It is elaborated in section 3.1. Further, since any device can hold the responsibility for message dissemination, it should also be able to autonomously deduce whether or not the desired coverage has been achieved. We describe in section 3.2 how Scribble provides this ability.

The final design issue is one of ensuring that the coverage is achieved. Existing protocols, as discussed earlier, rely on the cooperation of the network so that the remedial actions are not needed (the most optimistic case) or eventually succeeds if carried out in full (the most pessimistic case). Our protocol is optimistic that the structure-less message dissemination and coverage deduction mechanisms swiftly achieve and deduce the desired coverage, while its pessimistic expectation is that the minimal liveness property is satisfied. The minimality of the liveness property allows our protocol to operate under a wide range of network conditions. To put it differently, if our protocol cannot terminate in a given network condition, no deterministic protocol will. Section 3.3 explains the (extreme) measures our protocol gradually resorts to, when the network satisfies just the minimal liveness property in the least helpful manner (like an adversary) to the protocol.

3.1 Message dissemination

The responsibility for message dissemination initially rests with the originator and is subsequently passed on to other devices (as in a relay-race). Consequently, a device can be in one of two states regarding the dissemination of m: *responsible* or *passive*. A responsible device transmits *m* once, and then repeats this transmission every β seconds if required. Obviously, the number of devices simultaneously responsible should be kept low, particularly when increasing that number would not provide any further coverage. The scheme employed for passing on the responsibility achieves this objective by striving to keep at most one device responsible in any subset of devices that are in each others wireless range (a *fully connected* subset); moreover, at least one device is kept responsible in the group as a whole at any time. The scheme makes use of the following known results:

K1: The additional coverage expected from a device's transmission drops exponentially with the number of transmissions that have already occurred in the device's wireless range immediately before that transmission [12].

K2: Consider two devices with clocks whose values never decrease. Let them transmit a message timestamped with their local clock values. If the devices receive each other's message, it is not possible for both the devices' local clocks to be smaller than the timestamp on the received message.

In our protocol, timestamps are issued based on a *logical clock* while timeouts are set using (unsynchronised) physical clocks. A logical clock is just an integer counter whose value can only increase, though not necessarily in relation to the passage of real-time. A device N_i constructs a logical clock $L_i(m)$ with the initial value of zero when N_i first knows of *m*. Recall that the initiator is the first device to know of, and to be responsible for, m.

A responsible N_i transmits m soon after becoming responsible. It repeats this transmission every β seconds until it becomes passive. A transmission, however, is preceded by the following activities: N_i chooses a Random Assessment Delay (RAD) uniformly distributed on (0, MAX_RAD), and *schedules* a transmission of *m* after the RAD has expired. If the RAD expires, N_i increments $L_i(m)$ by 1, time-stamps *m* with the value of $L_i(m)$ and transmits *m*. This (logical) time-stamp is denoted as *m.l* of the transmitted *m*. Suppose that a device N_j is passive on *m* and receives *m*. N_j becomes responsible for *m* if

R1 : *m.l* of the received *m* is greater than, or equal to $L_j(m)$, and

R2 : N_j has not received *m* in the past β-δ seconds.

If R1 and R2 are true, N_j becomes responsible after setting $L_j(m)$ = *m.l* of the received message *m*. A responsible device N_i becomes passive if it receives *m* such that:

P1 : *m.l* of the received $m > L_i(m)$.

Upon becoming passive, N_i cancels any active RAD and thereby any pending transmission of *m*.

Remark 1. If N_j that has just become responsible, receives *another* transmission of *m* that meets **P1**, then it instantly becomes passive canceling any RAD that it had just set and any transmission scheduled. This is in conformance with **K1**, and means a responsible device could become passive before ever transmitting m.

Remark 2. When two responsible devices receive each other's *m*, **K2** and **P1** do not permit both devices to become passive. However, one of them will become passive except in the unlikely case of both having identical logical clock values and transmitting *m* nearly at the same instant despite RAD.

3.2 Coverage deduction

Scribble ensures that *any* responsible device is able to deduce the achievement of the desired coverage once the latter is obtained, and does so using only local information. This is done in the following manner: The protocol requires that there be a dedicated header field, *m.K*, in each data packet m. This fields contains the "signatures" of devices which have already received m. A signature is a compact representation of a node id. In addition, each device N_i maintain a data structure, $K_i(m)$, which contains the signatures of the devices it knows to have received m. When N_i receives m, it merges its knowledge about which devices have received the message, $K_i(m)$, with the knowledge already in the message header, $m.K$. If the received *m*.K does not contain its signature and if it subsequently decides not to transmit *m*, it transmits a small *acknowledgment packet* for *m*. Upon reception of such an acknowledgment, a node updates its $K_i(m)$ appropriately.

The following rules enable N_i to realise *m*:

1. N_i realises *m* when $K_i(m)$ contains *k* signatures.
2. On realisation by (1) or upon receiving *m* thereafter, N_i transmits a small, 1-hop *realisation packet* for *m* which only contains the id of *m*.
3. N_i realises *m* on receiving a realisation packet for *m*. Realisation by this rule does not cause N_i itself to transmit a realisation packet.

A signature is essentially a device's acknowledgment tagged onto *m*. It needs to be unique and device-specific as it is counted to assess coverage. Signatures, and thus K(*m*), can be represented in numerous ways, and someone implementing the protocol should decide what is most appropriate depending on the likely size of the group and the processing and storage ability of the devices in it. In relatively small groups within an administrative domain such as a university or a company, it might be sufficient to use the use the last byte of the IPv4 or MAC address. This is the approach taken for the simulations presented in section 4

3.3 Protocol termination guarantees

In adversarial network conditions, a deterministic protocol may not be able to achieve the required coverage simply by passing on the dissemination responsibility from device to device. Consider for example the fully-connected subset $\mathcal{C}$ which contains c_0 which initiates the dissemination of *m*. If $|\mathcal{C}| < k$, some more devices, such as $d \notin \mathcal{C}$, must receive *m*. In the absence of any such *d* which enters the wireless range of some $c_i \in \mathcal{C}$, it is easy to see that the desired coverage cannot be obtained. Putting the above differently, the desired coverage is guaranteed only if some $c_i \in \mathcal{C}$ is directly connected to one or more devices, such as $d \notin \mathcal{C}$, *and* if c_i is responsible during this period of direct connectivity. The former will occur since the network meets the minimal liveness property (see § 2.1.1) and when it does occur, the protocol must ensure the latter. Note that the network can choose *any* $c_i \in \mathcal{C}$ and place the chosen c_i and *d* in direct connectivity at arbitrarily chosen timing instants. Indeed, the network can behave *like an adversary*, enabling the direct connectivity between c_i and *d* only when the former is passive. This means that a protocol which keeps only a subset of devices in $\mathcal{C}$ responsible, however cleverly designed, cannot guarantee that the right devices in $\mathcal{C}$ are responsible at the right time. So, the 'desperate' measure taken by our protocol involves gradually allowing *all* devices in $\mathcal{C}$ to become responsible when the protocol appears not to be terminating. This is achieved by requiring each device N_i to have a parameter θ_i. If $L_i(m) \geq \theta_i$ or if N_i receives *m* with $m.l \geq \theta_i$, then N_i becomes responsible (if it is not already) and does not become passive until it realises *m*. N_i is then said to be in the *ANGRY* (Actively eNGaged to ensure ReliabilitY) state, and θ_i is called the *ANGER* (Active eNGagement to Ensure Reliability) threshold whose value can be chosen by N_i autonomously.

The discussion above indicates that the desperate measure of Scribble is necessary for termination in the most pessimistic cases. In [17], we show that it is also sufficient to achieve termination and for any responsible device to achieve realization. This also allows us to claim that *any* deterministic protocol must allow every device that has *m* to enter the *ANGRY* state; otherwise, termination cannot be guaranteed even when the network satisfies the minimal liveness property (but behaves like an adversary).

4 Protocol Performance

We have studied the performance of Scribble with $k = n$ to ODMRP, a *best-effort* multicast protocol, with all devices in the network being members of the multicast group. The simulation parameters used were as in Table 1, unless otherwise stated. We used the default parameters for ODMRP as suggested by its authors. In all experiments Scribble set aside 64 bytes for signatures in each data header. All simulations used a byte for each device signature (c.f. section 3.2). θ_i was chosen to be a high value of 666(!) so that no device enters the ANGRY state.

The parameters of interest were: *Transmission overhead*, which was measured by counting all the bytes put on the network and dividing by the number of nodes in the system and the size of the data payload in each message, and *Percentage Successful Runs (PSR)*, defined as the percentage of protocols runs where a message gets delivered to all nodes. This will of course always be 100% for Scribble.

Table 1. Simulation parameters

Simulation Parameters	
Simulator	Glomosim v. 2.03
No. nodes	48
Area size	1000m x 1000m
Mobility model	Random Waypoint
Min. speed	1m/s
Pause time	0s
Total simulation time	3000s
Packet generation rate	1pkt/s
Total number of packets	100
Packet size	512bytes

A wide range of wireless networking scenarios was modelled by varying average device speed from 0.5 to 17.5m/s, and varying the density by varying the wireless range of devices between between 150 and 350m while keeping the simulation area and number of devices constant Figure 1

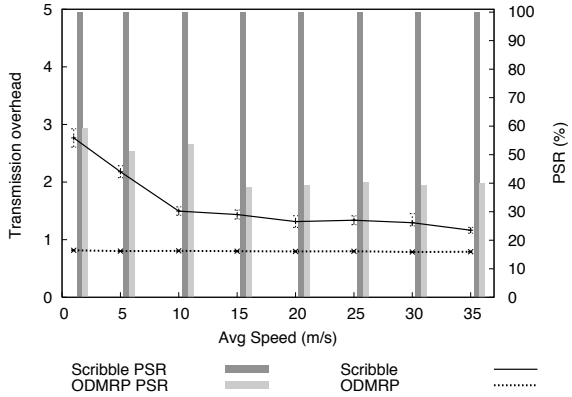

Figure 1. Transmission overhead and PSR vs Average speed with wireless range = 250m

shows the impact that relative change of topology has on the two protocols, when the wireless range of the devices is set to 250m. PSR is shown as bars on the right y-axis, while the transmission overhead is drawn as lines relative to the left y-axis. As expected, ODMRP performs relatively well in terms of overhead with these parameters (these conditions are in fact very similar to those chosen by the ODMRP authors themselves in [8]), with mobility having little impact on the transmission overhead, as there are no reconstruction efforts in place in case the routing mesh breaks. However the average number of times ODMRP is able to deliver a message to all intended recipients is below 60% even in the most static scenario (0.5 m/s), with the PSR dropping to just under 40% in the most mobile case (17.5m/s). Further, when the number of successful recipients of individual packets is studied, one can observe that in extreme cases, when the originating device is partitioned from the rest of the network, the protocol terminates with only the originating device itself having received the packet. Scribble, on the other hand, provides its delivery guarantees with little additional overhead compared to ODMRP, with the additional overhead being higher when the mobility is relatively low. The reason for this is that Scribble guarantees delivery to all devices including those which might be transiently partitioned from the rest. When the mobility is low, these partitions takes longer to heal, so the cost of guaranteeing delivery to partitioned devices is higher, thus increasing the overall cost of guaranteeing delivery. Figure 2

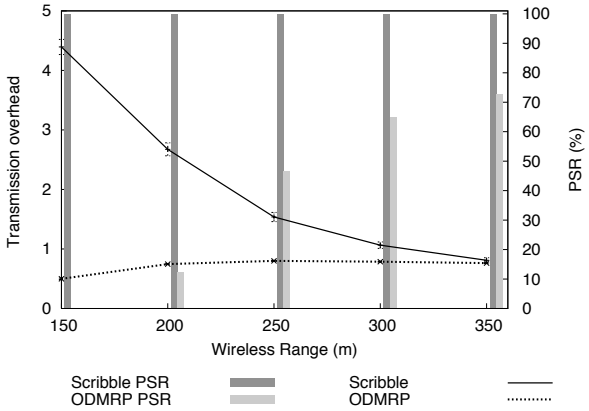

Figure 2. Transmission overhead and PSR vs Wireless range with average speed = 5 m/s

shows how transmission overhead and PSR vary as function of density. These results show very clearly the cost of attaining certainty that a message will be delivered to enough devices. As one would expect, the cost of guaranteeing delivery is not excessive when the network is fairly dense (remember we do not consider very congested conditions). In fact ODMRP and Scribble have almost identical overheads in the densest case considered, though note that even here ODMRP is not able to provide delivery to all devices in more than 72% of cases. As the density decreases, the PSR ODMRP achieves drops dramatically, going as low as 0.2% of cases in the sparsest network (this is too small to show up on the graph). Scribble on the other hand maintains its 100% PSR, though at what can be considered considerable costs in the sparsest cases.

5 Conclusions and future work

Providing deterministic delivery guarantees in mobile P2P systems is a critical requirement in order to enable a numberof fundamental problems to be solved effectively in the infrastructure-less pervasive computing context. In this paper we have presented a k-deterministic reliable many-to-many dissemination protocol, called Scribble, which is able to provide such guarantees in an efficient manner by decentralizing responsibility for message dissemination. Decentralization leads to the requirement that the ad-hoc net-

work meet only a minimal liveness property. Consequently, Scribble is able to operate in a wide range of network scenarios, and efficiently so. Future work includes making Scribble crash-tolerant.

Acknowledgments

Research supported by a UK-EPSRC grant GR/S02082/01, a UK Overseas Research Scholarship, and the School of Computing Science, University of Newcastle, UK.

References

[1] R. Chandra, V. Ramasubramanian, and K. Birman. Anonymous gossip: Improving multicast reliability in mobile Ad-Hoc networks. pages 275–283.

[2] D. Cooper, I. Mitrani, and P. Ezhilchelvan. High coverage broadcasting for mobile ad-hoc networks. In *The Third IFIP-TC6 Networking Conference*, 2004.

[3] P. Ezhilchelvan, A. Mostefaoui, and M. Raynal. Randomized multivalued consensus. In *Proceedings of the 4th IEEE International Symposium on Object-Oriented Real-Time Computing*, pages 195–200, May 2001.

[4] D. Ganesan, B. Krishnamachari, A. Woo, D. Culler, D. Estrin, and S. Wicker. Complex behavior at scale: An experimental study of low-power wireless sensor networks. Technical report, UCLA, 2002.

[5] T. Gopalsamy, M. Singhal, D.Panda, and P. Sadayappan. A reliable multicast algorithm for mobile ad hoc networks. In *Proceedings of ICDCS*, 2002.

[6] S. Gupta and P. Srimani. An adaptive protocol for reliable multicast in mobile multi-hop radio networks. In *IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[7] C. Ho, K. Obraczka, G. Tsudik, and K. Viswanath. Flooding for reliable multicast in multi-hop ad hoc networks. In *Proceedings of the 3rd International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*, pages 64–71, Seattle, WA, 1999.

[8] S. Lee, W. Su, and M. Gerla. On-demand multicast routing protocol in multihop wireless mobile networks, 2000.

[9] J. Lipman, P. Boustead, and J. Chicharo. Reliable optimised flooding in ad hoc networks. In *IEEE 6th CAS Symposium on Emerging Technologies: Frontiers of Mobile and Wireless Communication (MWC'04)*, May 2004.

[10] W. Lou and J. Wu. Double-covered broadcast (DCB): A simple reliable broadcast algorithm in manets. In *IEEE Infocom*, 2004.

[11] J. Luo, P. T. Eugster, and J.-P. Hubaux. Route driven gossip: Probabilistic reliable multicast in ad hoc networks. Technical report, EPFL, 2002.

[12] S. Y. Ni, Y. C. Tseng, Y. S. Chen, and J. P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *ACM/IEEE MobiCom*, 1999.

[13] E. Pagani and G. P. Rossi. Reliable broadcast in mobile multihop packet networks. In *Proceedings of MobiCom*, 1997.

[14] E. M. Royer and C. E. Perkins. Multicast operation of the ad-hoc on-demand distance vector routing protocol. In *Mobile Computing and Networking*, pages 207–218, 1999.

[15] K. Tang, K. Obraczka, S.-J. Lee, and M. Gerla. Reliable, congestion controlled multicast transport protocol in multimedia multi-hop networks. In *Proceedings of WPMC*, 2002.

[16] E. Vollset. EPSRC PACE project website: http://vollset.org/pace.html. July 2004.

[17] E. Vollset and P. Ezhilchelvan. The design and evaluation of an efficient reliable manycast protocol for ad-hoc networks. Technical Report CS-TR-838, School of Computing Science, University of Newcastle upon Tyne, 2004.

[18] E. Yoneki and J. Bacon. An adaptive approach to content-based subscription in mobile ad hoc networks. In *IEEE International Conference on Pervasive Computing and Communications - Workshop on Mobile Peer-to-Peer Computing*, 2004.