Reliable Collaborative Decision Making in Mobile Ad Hoc Networks

Theodore L. Willke[1,2] and Nicholas F. Maxemchuk[1]

[1] Columbia University, Dept. of Electrical Engineering
1312 S.W. Mudd
500 West 120th Street
New York, NY 10027
tlw24@columbia.edu, nick@ee.columbia.edu
[2] Intel Corporation, Enterprise Platforms Group
2800 Center Drive, M/S DP3-307
DuPont, WA 98327
theodore.l.willke@intel.com

Abstract. Mobile units, such as vehicles in a traffic flow or robots in a factory, may use sensors to interact with their environment and a radio to communicate with other autonomous units in the local area. Collaborative decision making can be carried out through information sharing amongst these units and result in cooperative problem solving. Examples of solutions include coordinated vehicle control for collision avoidance and executing complementary path plans for robots on a factory floor. We propose an application-level protocol that enables units to contribute their local knowledge and actions to a shared global view of the problem space. The protocol uses a time-driven token ring architecture to permit any unit to reliably broadcast a message to the entire group. The protocol ensures that all units commit the same set of received messages in the same order by a deadline, and it upholds these guarantees in the presence of channel failures, hidden units, and a changing set of collaborators. Units in the network are made aware of when others fail to maintain the global view. Failing units may be required to operate autonomously, pending information recovery.

1 Introduction

The growing ubiquity of both mobile computing and wireless networking will motivate new applications for this technology, including cooperative problem solving. Mobile units, such as vehicles in a traffic flow, tanks on a battleground, or robots in a factory, may be equipped with sensors to interact with their local environment, a radio to share information with other autonomous units in the immediate area, and a computer to execute a program. These resources enable the units to collaborate in the collection and dissemination of environmental information that can be used to make decisions that ultimately result in a single coherent solution to a problem. Examples of solutions include coordinated vehicle control for collision avoidance and computing non-intersecting trajectories for robots on a factory floor.

Units are enabled to act coherently and achieve common goals if they use a communication protocol that permits their local knowledge and actions to contribute to a shared global view of the problem space. Units may either carry out local, decentralized decisions or be directed by one or more units making centralized decisions. In the former case, the global view supports local decision making so that globally-optimal behavior results. In the latter case, the global view permits any unit to take the role of decision maker if communication with a centralized resource is lost. Both approaches must contend with an environment where communication channel limitations and unit failures can result in a continuously changing set of collaborating units.

We propose an application-level protocol, the Mobile Reliable Broadcast Protocol (M-RBP), that is particularly well-suited to collaborative decision making in this environment. The protocol permits any unit to reliably send a message to every other unit in the group and is time-driven to ensure that all units commit the same set of received messages in the same message order by a maximum delay following initial message acknowledgement; this enables the global view to be kept up-to-date in a timely fashion. Units also learn when they have lost the global view and may be required to operate autonomously using a different, more conservative set of assumptions until sufficient information is recovered. Other units in the group are made aware of these failures and can react appropriately.

The remainder of this paper is organized as follows. The scope of the network considered is described in Section 2. Section 3 introduces the architecture of M-RBP and its operation. Section 4 presents a comparison of M-RBP with other existing reliable broadcast and multicast protocols for this application. Section 5 briefly discusses protocol scalability and performance tradeoffs, and Section 6 concludes the paper.

2 Scope of the Networking Problem

Figure 1 is an example of the scope of networking that we will address for collaborative decision making. A number of mobile units (e.g., *a* through *f*) are collaborating in a localized geophysical region, or LAN. Some units may be within direct communication range of all other units, while others may separated (i.e., hidden) from each other by greater than one hop distance and need to rely on neighboring units to relay information (e.g., *a* to *c*, *e*, or *f*). The relaying of information may also be required when an obstruction prevents direct communication between units (e.g., *d* and *e*).

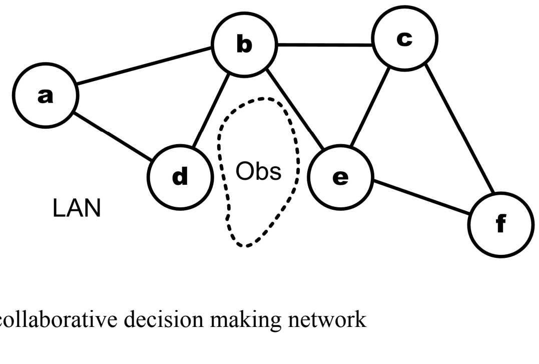


Fig. 1. Scope of the collaborative decision making network

Our protocol provides delivery guarantees for broadcast transmissions from any of the units to the rest of the group. Some of the guarantees, such as maximum delay, scale with the network hop diameter and number of units. Although this protocol will function properly on larger multi-hop networks, it is best suited to implementations in which most units can directly communicate with one another. Providing a global view for the group remains difficult, even for small-diameter networks, due to a changing set of collaborators and the unreliability of wireless network connectivity.

3 M-RBP Architecture

M-RBP is an application-layer protocol that accepts data from user applications and transmits it using the IP broadcast address. We assume a networking stack consisting of UDP/IP services and an IEEE 802.11 MAC and PHY design [1] with the distributed coordination function. This medium provides physical signaling, a broadcast addressing scheme, and carrier sensing with collision avoidance. It does not provide a request-to-send, clear-to-send handshake or a data transfer acknowledgement.

Mobile units that share the global view participate in a token ring protocol. Previous work on token ring protocols for reliable broadcast and multicast focused on wired network implementations [2], [3], [4]. Figure 2 is an illustration adapted from the Reliable Broadcast Protocol [4] of a token ring comprised of receivers in a broadcast group. In our application, *n* mobile units can serve as both message sources and receivers. Sources transmit messages at will into the medium, with an identification of the source unit, *s*, and a source-specific sequence number, M_s. M_s is incremented for each unique broadcast message so that duplicate transmissions may be identified.

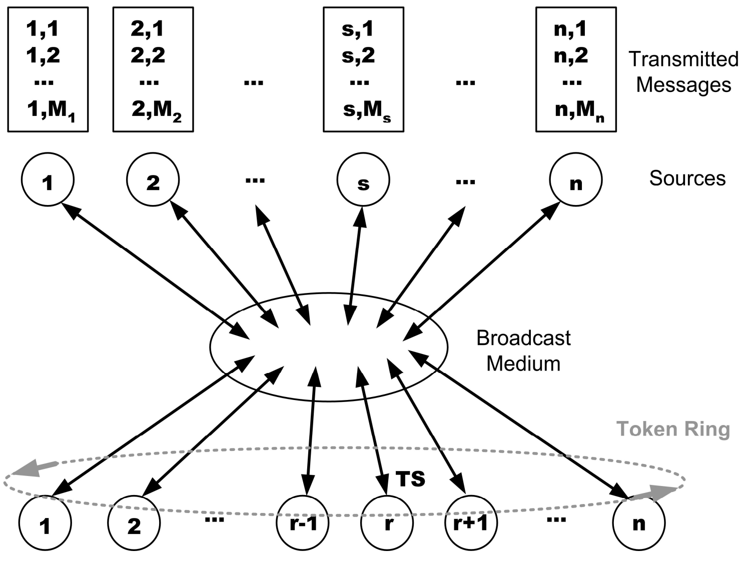


Fig. 2. Receivers in the broadcast group belong to a token ring

A token is passed amongst the units in the receiver group on a timed schedule. The receiver with the token is referred to as the token site, and it is responsible for acknowledging any source messages received from the source unit, *s*, while in possession of the token, as well as some additional messages described in Section 3.5. The token site acknowledges messages using a single bulk acknowledgement (ACK) that references the messages and assigns them a relative sequence order. Sources retransmit messages until they receive an acknowledgement from a token site. Units that do not receive the token site's ACK shortly after its scheduled transmission time are permitted to broadcast a retransmission request. Units that receive the ACK broadcast a negative acknowledgement (NACK) for any messages that they are missing. The NACK is repeated until a peer services the retransmission request or other criteria described in Section 3.3 is met. Broadcast retransmissions include the original message identifier (s, M_s) and the retransmitter's source identifier; this prevents retransmissions from being mistaken for transmissions from the message source.

On a timed schedule, the group collectively determines what ACKs to use for global message ordering and what messages should be committed. By a specific deadline, all surviving units learn what messages are committed by their peers.

In the following sections, we describe aspects of the protocol in more detail.

3.1 Implicit Time-Based Token Passing

The token site is responsible for acknowledging messages and initiating global sequencing. In several token ring protocols [2], [3], [4], an explicit handshake is used to: 1) acknowledge source messages, 2) confirm acceptance of the token from the previous token site, and 3) request transfer of the token to the next token site, as shown in Figure 3. This approach prevents continuous circulation of the token in the presence of frequent unit and communication failures because, in these cases, the required handshake may not transpire.

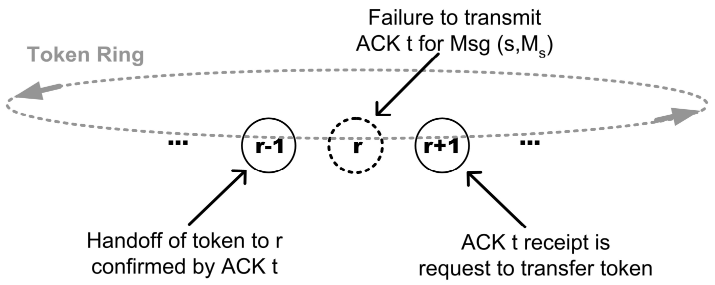


Fig. 3. Explicit ACK handshake used by many token ring protocols. If the token site, *r*, fails then the token ceases to circulate until the ring completes a lengthy repair process [2]

M-RBP, by contrast, uses a token that is passed based on time, without further qualification. Each receiver holds the token for a duration of Δ_T seconds in a time slot specified by a token passing list (TPL) and is expected to transmit a single ACK at the end of its assigned token passing interval. In addition to its role in message acknowledgement, the ACK is used to indicate continued participation of the acknowledging unit and enables each receiver, through an algorithm described in Section 3.2, to keep identical, and perform the same maintenance on, local copies of the TPL. Because no explicit handshake is required to pass the token, the communication with the token site may fail without disrupting token circulation. Relative synchronization of units is required, but is not addressed in this work.

3.2 Maintaining the Token Ring using a Distributed Time-Driven Algorithm

A unit can infer that a token site has failed if its scheduled ACK broadcast is not recovered by a deadline. Individual units, however, may disagree on the failure, depending on how successful their own recovery efforts are. Units could use a gossip protocol [5], [6] to spread the ACK or the token site could require positive acknowledgement of its ACK by every other unit in the group, but these approaches would only provide highly probable agreement by a deadline. We have devised a distributed time-driven token ring repair process that ensures agreement by a deadline.

A conceptual timeline for the ring repair process is shown in Figure 4. Since all units have a copy of the TPL and can identify when a specific unit is scheduled to transmit its ACK as the token site, they all know when to expect the ACK and can begin attempting recovery (described below) shortly thereafter. Each unit that does not recover the ACK by a deadline assumes the token site has failed. After making a determination, each unit in the group broadcasts a "yes" or "no" vote to drop the unit in question. All units attempt to recover as many of these votes from peers as possible and, at a prescheduled deadline, they each attempt to determine a group consensus using an agreement function. The consensus will either be to take no action or to remove the unit from the TPL. If the unit is removed, its TPL entry is deleted and the entries below are shifted up to fill the void.

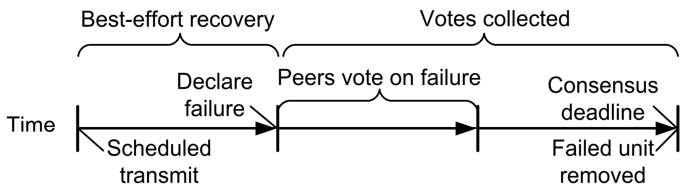
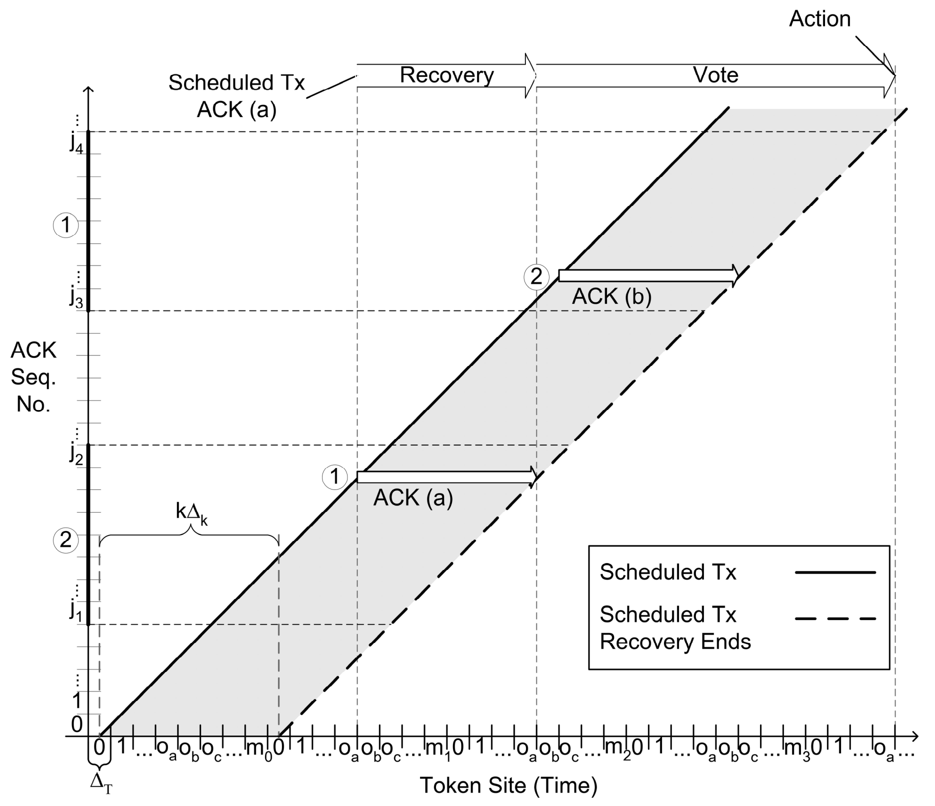


Fig. 4. Timeline for the time-driven distributed token ring repair process

In M-RBP, the votes are transmitted in ACK messages to minimize control overhead. The relationship between scheduled ACKs and the votes they carry is shown in Figure 5. A token site is assigned a time slot of Δ_T seconds based on its offset in the TPL. The offset numbers are shown on the x-axis, with token round x having m_x slots. As shown on the y-axis, each token site transmits an ACK with a sequence number j at the end of its time slot. The set of scheduled ACK transmissions is delineated by the *solid line* on the chart. The units attempt to recover each ACK using a time-driven process involving k iterations, each of length Δ_k. The parameters for the process are chosen so as to guarantee a high probability of ACK recovery by the final iteration. The recovery period for each ACK is shown by the *gray band* in the chart, the end of which is delineated by a *dashed line*. Each unit that did not receive an ACK by the end of its recovery phase indicates this by including a drop field in its next ACK transmission that references the source and sequence number of the missing ACK (a vote against unit removal is implied by the absence of this field). In the example shown in Figure 5, the votes applying to the ACK transmitted by unit *a*, at the position labeled 1, are transmitted by peers in subsequent ACKs with sequence numbers j_3 through j_4.



Action
Scheduled Tx ACK (a)
Recovery
Vote
ACK Seq. No.
j_4
1
j_3
j_2
2
j_1
1
0
$k\Delta_k$
ACK (a)
ACK (b)
Scheduled Tx
Scheduled Tx Recovery Ends
0 1 … o_a o_b o_c … m_0 0 1 … o_a o_b o_c … m_1 0 1 … o_a o_b o_c … m_2 0 1 … o_a o_b o_c … m_3 0 1 … o_a …
Δ_T
Token Site (Time)

Fig. 5. Timeline for scheduled ACK transmission and recovery

Because each peer transmits its ACK, including votes, only once per token ring cycle, each ACK carries votes pertaining to all ACKs that reached the end of the recovery phase in the last token ring cycle. For example, the ACK transmitted by unit *b* at the position labeled 2 includes votes relevant to ACKs with sequence numbers j_1 through j_2.

When all of the ACKs associated with a particular vote (e.g., the range labeled 1 and associated with the ACK transmitted by unit *a*) have themselves completed their recovery phase, the group decides on whether to take action and remove the respective unit from the TPL. If all units take the same action at this point in time, their

TPLs will be adjusted in the same manner and time slot reassignment can take place to fill in the vacant slot. However, attaining unanimous agreement is complicated by the fact that each unit may recover a different subset of the ACKs associated with a particular vote.

The units ensure unanimous agreement on TPL changes through the use of a majority agreement function that returns one of three possible values. If a unit recovers a sufficient number of ACKs to determine that the group consensus is “yes” or “no” on an action, the function returns a value of *Y* or *N*, respectively. If a unit fails to recover enough ACKs to determine the group consensus, the function returns a value of *U* for “unknown”. In this case, the unit must forfeit its time slot in the token ring until it has rejoined the token ring with a new time slot assignment (see Section 3.4).

To make a decision based on a majority, each unit must determine that either greater than 50% of the expected vote transmissions, *V*, are “yes” votes or at least 50% of the transmissions are “no” votes. The total number of votes received by each unit is less than or equal to the number transmitted because some of the units may not transmit a vote (e.g., due to failure) and some votes may not reach a particular unit by the time the consensus deadline is reached. Given that unit *i* recovers Y_i “yes” votes and N_i “no” votes, the tri-valued function *F* is expressed as:

$$F(Y_i, N_i, V) = \begin{cases} Y & \text{if } Y_i > \left\lceil \frac{V}{2} \right\rceil \\ N & \text{if } N_i \geq \left\lceil \frac{V}{2} \right\rceil \\ U & o.w. \end{cases} \tag{1}$$

We have chosen to use threshold values that require either a majority of “yes” votes or a majority of “no votes to be recovered to determine group consensus with certainty. Our choice minimizes the maximum number of “yes” or “no” votes that a unit must recover in order to avoid an “unknown” determination. Other threshold choices may better maximize a unit’s probability of survival, especially if the selection is based on estimated probabilities of receiving a “yes” vote or a “no” vote. In any case, the thresholds chosen must guarantee a mutually-exclusive “yes” or “no” determination by *F*, no matter how many votes an individual unit successfully recovers. This can be accomplished by ensuring that the following holds for the choice “yes” and “no” thresholds, T_Y and T_N:

$$T_y + T_N \geq V + 1 \ , \tag{2}$$

In summary, by using *F* to determine the voting consensus and by acting in accordance with the consensus at the prescribed deadline, all units remaining in the token ring maintain identical copies of the TPL.

3.3 Reliable and Consistent Messaging with a Delay Guarantee

The distributed protocol used to maintain each unit’s TPL can also be used to provide reliable and consistent message delivery between all sources and receivers. For real-time collaborative decision making, we desire to provide: 1) global sequencing of messages; 2) consistent commitment of messages across the group of receivers; and 3) notification of message commitment between each unit and its peers. Because the protocol provides units with a concept of time and units take action on a schedule, we can offer reliable and consistent message delivery with a delay guarantee.

A typical approach to defining the probability of reliable message delivery, $P_r(t)$, is to state that it monotonically increases to a value sufficient to meet application requirements at some time τ following a number of retransmission attempts, as shown in Figure 6. We desire a definition of reliability that is more suited to a continuously changing receiver set and that offers a specific reliability guarantee at a deadline. To this end, we pursue a guarantee that a receiver remaining in contact with its peers and participating in the token ring protocol for $> \tau_1$ seconds after a message is initially acknowledged will commit that message if the group reaches a consensus to do so. Furthermore, if the receiver remains in the group for $> \tau_2$ seconds, where $\tau_2 > \tau_1$, its peers can verify that it has committed the message.

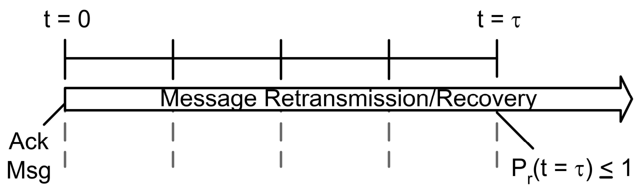


Fig. 6. Typical reliability guarantee

The timeline for providing the new reliability guarantee is shown in Figure 7. The timeline starts at time t with acknowledgement of message (s, M_s) by scheduled ACK j. The process proceeds in three phases, with each phase involving information recovery followed by a group consensus vote on the success of the recovery. Since the token ring length may increase or decrease by one unit each Δ_T, the number of expected votes associated with each phase is labeled uniquely as m_a, m_b, and m_c. A unit that cannot determine the voting consensus for any vote must relinquish its place on the TPL.

The first phase of the process involves recovery of the scheduled ACK j that acknowledges the source message of interest, in this case message (s, M_s). Since the ACK is also used to infer the token site's continued operation, the first vote both determines whether the associated token site is considered operational and whether the ACK will be used for message sequencing.

Some messages referenced by dropped ACKs may never be referenced again by future ACKs. These may be discarded $> \tau_1$ seconds after their reception without concern that they need be committed.

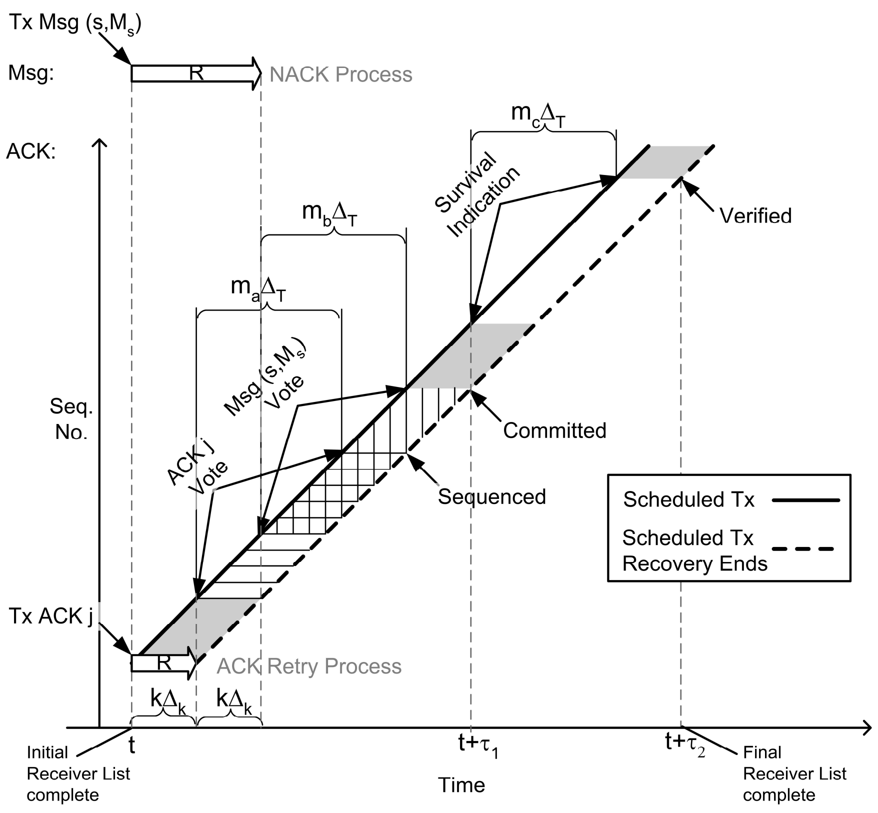


Fig. 7. Time-driven process for global message ordering, commitment, and delivery confirmation

The second phase involves message recovery. Units that successfully recover the ACK attempt to recover the associated message(s) until one of the following events occur: 1) the message is recovered; 2) the group reaches a consensus to not use the ACK; or 3) the deadline for message recovery is reached. The deadline for message recovery follows the deadline for ACK recovery by $k{\cdot}\Delta_k$ seconds. At the deadline for message recovery, each unit that recovered ACK *j*, but not message (*s*, M_s), conveys this in its next scheduled ACK transmission by voting to drop (*s*, M_s). In general, the ACK may indicate that one or more messages be dropped in a variable-length field. If the voting unit did not recover ACK *j*, it votes generically to drop all messages referenced by the ACK. Message reception is implied for all messages referenced in ACK *j* that are not listed in the drop field of the ACK used in the vote. If a majority verifies message reception, the message is committed. If the majority votes to drop the message, it is discarded.

In the third phase, ACK recovery is used to determine the set of receivers that survived to commit the message. A unit determines that a peer has survived to receive the message if the peer transmits its scheduled ACK in the token round starting at $t+\tau_1$. If the unit either directly receives the peer's ACK or if any other peer indicates that it received the peer's ACK via a vote, then the peer in question survived to commit the message. A unit can verify the set of peers that received the message by applying this test to each peer in the TPL after time $t + \tau_1$. This verification is complete no later than time $t + \tau_2$. τ_1 and τ_2 are a deterministic function of the number of units in the group during several token ring cycles as well as a few protocol time constants.

This three-step process ensures that every unit that continues to collaborate in decision making has committed a message within, at most, three recovery periods and one token ring cycle of when it was initially acknowledged and that every unit knows what peers have committed the message within, at most, four recovery periods and two token ring cycles.

3.4 Join Requests

Units that intend to join the ring for the first time, or that were dropped and want to rejoin, must transmit a source message with a Join Request field. The token site that acknowledges this message will respond with an ACK that includes a copy of the TPL and protocol parameters. The unit is neither admitted to the token ring, nor does it begin sharing the global view, until its source message is committed by the group. The new unit is assigned the TPL entry corresponding to the first token passing interval that begins after the time of message commitment.

The unit must be ready to participate in the ring repair process as soon as it is added to the TPL. Therefore, it must begin to recover ACKs and monitor vote outcomes as soon as its message is acknowledged. Since all transmissions, scheduled or not, include source identifiers, the unit can maintain an updated list of one-hop neighbors. It may transmit unicast ACK Retry messages to these neighbors in a round-robin fashion each Δ_k seconds. The addressed neighbor may service the request by retransmitting it or, if the requested ACK was dropped by the group, respond with a unicast ACK that includes a drop field for the requested ACK. Changes to the TPL are inferred through the ACK history.

Units that are rejoining the group after being recently dropped and that desire to maintain a continuous global view must, in addition, request missing source messages via unicast NACK messages. The addressed neighbor may service the request with a message retransmission or respond with a unicast ACK that includes a drop field for the requested message(s).

3.5 Dealing with Hidden Units

The presence of hidden units challenges the delivery of messages to receivers hidden from a source and the acknowledgement of messages when the token site is hidden from a source. To solve these problems, ACK and source message retransmission requests are serviced by nearest-neighbor peers using a time-based recruitment process.

During the recovery window for a particular ACK, the units on the TPL are progressively recruited to service retransmission requests for the ACK and to request its retransmission themselves. The recruitment starts with the unit on the TPL that was originally scheduled to transmit the ACK and continues until the entire TPL is recruited. The number of units recruited per iteration may adhere to any number of profiles. For example, one strategy may be to exponentially recruit units until the entire TPL is recruited, so that a total of 2^j units are recruited $j{\cdot}\Delta_k$ seconds following the scheduled time of the ACK transmission. Another strategy would be to recruit one unit in the first iteration and all remaining units in the second iteration. In any case, the process halts when some k, where $k > j$, number of recovery iterations of length Δ_k have occurred. This mechanism grants all units an opportunity to service and request ACK retransmissions while lowering initial contention for the broadcast medium. Carrier sensing and collision avoidance is the MAC layer's responsibility and is outside the scope of this research.

Source message retransmissions, driven by NACKs, use the same mechanism, with the only difference being when the recovery window starts, and completes (see Figure 7). The recruitment process for message retransmissions begins when the ACK recovery process completes, and continues for a period of $k{\cdot}\Delta_k$ seconds. The process is executed independently for each source message, and starts with the unit that originally acknowledged the message.

An example of message recovery between units hidden from one another is shown in Figure 8. All four of the units shown participate in the token ring, and unit 3 is the token site when unit 4 transmits message (4, M_1). The message is received by units 2 and 3. Unit 3 acknowledges message (4, M_1) with ACK Y. A specified delay after the scheduled transmission of ACK Y, unit 1 realizes that it missed the transmission and repeatedly transmits a retry for ACK Y. This retry is eventually received by unit 2, which services the retry by transmitting the ACK. Unit 1 then repeatedly transmits a NACK for (4, M_1). This NACK is eventually received by unit 2, which services the NACK by retransmitting the source message.

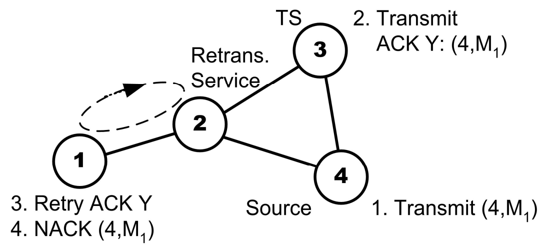


Fig. 8. ACK and source message recovery for hidden units

Another problem that arises with hidden units is that a source transmitting a message may be hidden from the token site. To deal with this, the source retransmits the message at regular intervals of Δ_T seconds until it receives an ACK that references the message. The ACK may be either a scheduled ACK transmitted by a token site or an unscheduled ACK transmitted by any unit within one hop. Units other than the token site may be recruited to respond to the transmission if it is repeated one or more times. The neighbors are recruited in growing numbers (e.g., exponentially) according to their relative offset in the TPL from the unit that was token site during the original message transmission. Unscheduled ACKs (i.e., those not transmitted by the token site) do not have an ACK sequence number and are not used for message ordering;

they simply inform the source that the message will be acknowledged later when the responding unit becomes the token site.

3.6 Global Message Ordering

As described previously, each ACK includes a sequence number *j*, indicating that it was transmitted at scheduled time $t_j = j \cdot \Delta_T$. ACK *j* assigns each message received, identified by the label (s, M_s), the relative sequence number *k*. Messages that were either acknowledged by a unit during the previous token round using an unscheduled acknowledgement, or received during its token passing interval, are assigned relative sequence numbers $k = 1, 2, ...$ in the order received. The 2-tuple (j, k) is used to assign message (s, M_s) a global order.

The received messages associated with ACK *j* are not assigned a global sequence number until all messages associated with ACK *i*, for $\forall i < j$, are sequenced. Duplicate references to message (s, M_s) may arise because sources can retransmit messages and because messages may reach units after a variable amount of propagation delay. To resolve this, all units commit messages according to the ACK with the lowest sequence number that references the message, and discard duplicate references. Then, the remaining messages associated with ACK *j* are ordered by increasing, not necessarily contiguous, relative sequence number *k*. This procedure results in the same message sequencing in all units that received the same acknowledgements.

4 Attributes of M-RBP

Token ring protocols have been studied for application to mobile ad hoc networks (MANETs) [7], [8]. The work described in [7] is limited to the study of several algorithms that permit a token to circulate amongst all members of a graph. WTRP [8] was developed for communication between unmanned vehicles. To provide bandwidth guarantees, each source is only permitted to transmit source messages in a time slot assigned by its position in the token ring. WTRP does not support mechanisms for reliable message delivery.

Numerous reliable broadcast and multicast protocols have been proposed for MANETs that do not use token rings [5], [6], [9], [10], [11], [12], and cannot provide global message ordering for many-to-many communication in multi-hop networks. These protocols are compared with M-RBP in Table 1, where attributes important for the support of collaborative decision making in a MANET are listed. None of these protocols were specifically developed for this application, and they all lack support for either message delivery confirmation with the source, or peer retransmission service. Furthermore, the gossip-based protocols, which do provide peer retransmission service, provide relatively weak reliability guarantees due to their probabilistic nature.

Because M-RBP was specifically developed to support collaborative decision making, it possesses all of the desirable characteristics.

Table 1. A comparison of M-RBP with other reliable broadcast and multicast protocols. Support of attributes that enable collaborative decision making in a MANET is indicated

Protocol		**Support**				
Class	**Scheme**	**Many-to-Many**[1]	**Peer Service**[2]	**ACK-Based Delivery**[3]	**Confirm Delivery w/ Source**	**Global Ordering**[4]
MAC	BMW w/ ODMRP [9]	✓	✓	✓		
Source Service	RALM [10]	✓		✓	On retry	
	RMA [11]	✓		✓	✓	
Hierarchical Service	FAT [12]	In omni mode	✓	✓		
Gossip Service	AG [5]	✓	✓			
	RDG [6]	✓	✓			
Time-Driven Token	M-RBP	✓	✓	✓	✓	✓

5 Comments on M-RBP Performance

Units that fail to recover ACKs and messages used in the global view are temporarily removed from the token ring. Therefore, the more effectively units recover information, the more likely they are to remain participating members of the group. As previously discussed, hidden units recover information using time-based retransmission protocols. Using a larger iteration interval, Δ_k, in these protocols provides units with more time to retransmit information and deal with congestion. However, the delay guarantees, τ_1 and τ_2, relax as Δ_k increases. This trade-off between delay guarantees and probability of unit failure will be analyzed in future work using the QualNet simulator [13].

The number of iterations, k, required of the time-based recovery processes has not yet been discussed. Independent of the profile used to qualify units to retry, or service retransmissions (e.g., one unit in the first iteration and all remaining units in the second iteration, exponential unit recruitment, etc.), the worst-case number of iterations occurs when: 1) all units, other than the token site, did not receive the original transmission, 2) the last unit permitted to retry is the only 1-hop neighbor of the token site, and 3) the network diameter is maximal, given the number of units (i.e., m - 1).

A worst-case network topology is illustrated in Figure 9. In this example, unit 1 is required to recover information from unit m by the end of the recovery period.

---

[1] Assuming a single instance of the protocol on each unit
[2] Information may be recovered from any peer, even if source fails
[3] Retry until positive acknowledgement
[4] For many-to-many broadcast or multicast

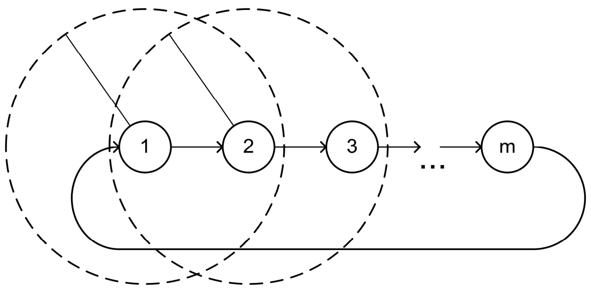


Fig. 9. Worst-case network topology (token ring shown) for a recovery process

The worst-case recovery process delay for the network depicted in Figure 9 using an exponential unit recruitment example is depicted in Figure 10. For this example and *m* units, the maximum number of iterations, *k*, required is:

$$k = \lceil \log_2 m \rceil + m - 2 \,. \tag{3}$$

In contrast, for a recruitment policy of a single unit in the first iteration and all units in the second, the relationship between iterations required and unit count is simply $k = m$. However, with this policy, medium contention may increase.

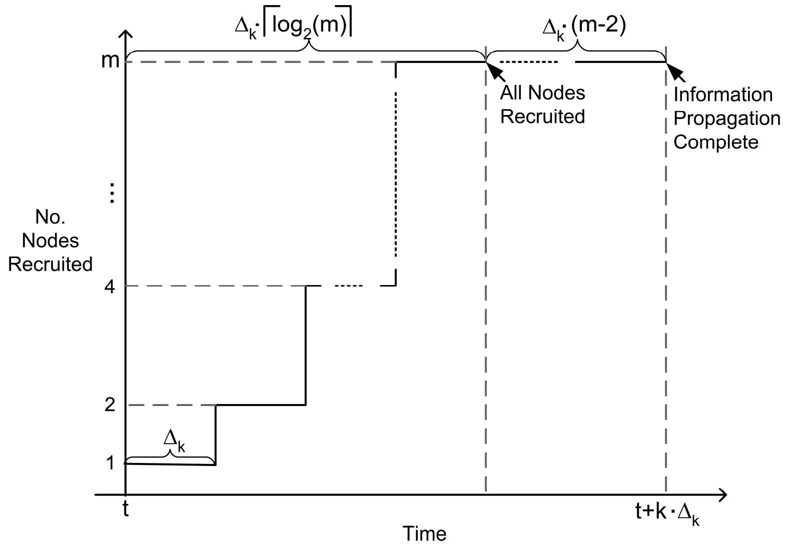


Fig. 10. Worst-case recovery delay for the scenario in Figure 9, using exponential recruitment

As an example of ACK recovery using the described network and exponential unit recruitment process, if $m = 9$ and unit 9 is the token site, it takes 11 iterations of the recovery process for unit 1 to recover unit 9's ACK. Thus, recovery is complete $11 \cdot \Delta_k$ after the scheduled transmission of the ACK, if the network remains connected.

6 Concluding Remarks

In this paper, we have discussed the problem of collaborative decision making in a mobile ad hoc networking environment. We have introduced an application-level protocol, the Mobile Reliable Broadcast Protocol, that provides the services necessary for collaborating mobile units to construct and share a real-time global view of the problem space, despite the potential for communication failures, a continuously-changing set of collaborators, and units that are hidden from one another. To the best of our knowledge, this support is not offered by any other existing protocol.

M-RBP uses a time-based token ring protocol to provide specific reliability and delay guarantees for source message commitment, something that is difficult, if not impossible, to accomplish with an event-driven protocol. It was shown that the delay guarantees provided can be tuned to alter the scalability of the protocol. It was also shown how novel time-based decision processes incorporated into M-RBP enable collaborating units to globally commit and order messages.

References

1. IEEE Standard for Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, P802.11, (1999)
2. Chang, J.-M., Maxemchuk, N.F., Reliable Broadcast Protocols, ACM Trans. Comp. Sys., vol. 2, no. 3, (1984) 251-73
3. Maxemchuk, N.F., Shur, D., An Internet Multicast System for the Stock Market, ACM Trans. Comp. Sys., vol. 19, no. 3, (2001) 384-412
4. Maxemchuk, N.F., Reliable Multicast with Delay Guarantees, IEEE Comm. Mag., vol. 40, no. 9, (2002) 96-102
5. Chandra, R., Ramasubramanian, V., Birman, K., Anonymous Gossip: Improving Multicast Reliability in Mobile Ad-Hoc Networks, Proc. IEEE ICDCS, (2001) 275-283
6. Luo, J., Eugster, P. Th., Hubaux, J.-P. , Route Driven Gossip: Probabilistic Reliable Multicast in Ad Hoc Networks, IEEE Proc. INFOCOM, (2003) 2229-2239
7. Malpani, N., Chen, Y., Vaidya, N.H., Welch, J.L., Distributed Token Circulation on Mobile Ad Hoc Networks, to appear in IEEE Trans. Mobile Computing, (2004)
8. Lee, D., Attias, R., Sengupta, R., Tripakas, S., A Wireless Token Ring Protocol for Ad-Hoc Networks, Proc. IEEE Aerospace Conf., (2002)
9. Tang, K., Gerla, M., MAC Reliable Broadcast in Ad Hoc Networks, IEEE MILCOM, (2001) 1008-1012
10. Tang, K., Obraczka, K., Lee, S.-J., Gerla, M., Reliable Adaptive Lightweight Multicast Protocol, Proc. IEEE Intl. Conf. on Comm., vol. 2, (2003) 1054-1058
11. Gopalsamy, T., Singhal, M., Panda, D., Sadayappan, P., A Reliable Multicast Algorithm for Mobile Ad Hoc Networks, Proc. ICDCS, (2002) 563-570
12. Liao, W., Jiang, M.-Y., Family ACK Tree (FAT): Supporting Reliable Multicast in Mobile Ad Hoc Networks, IEEE Trans. Veh. Tech., vol. 52, no. 6, (2003) 1675-1685
13. QualNet User's Manual, version 3.6, Scalable Network Technologies, Inc., (2003)