# **ROCC: A Communication Overlay Abstraction for Wireless Users**

Seth Holloway and Christine Julien
The University of Texas at Austin
Mobile and Pervasive Computing Lab
Austin, Texas, USA
{sethh, c.julien}@mail.utexas.edu

#### **Abstract**

As wireless devices become more popular, the potential for new collaborative applications has emerged. For example, a group of coworkers can opportunistically come together and share project documents and other data. Such applications are characterized by the fact that information needs to be distributed among all members of a group. Supporting such coordination currently requires using lowlevel multicast protocols that are complex and resource intensive to initialize and maintain without the support of an infrastructure. In this paper, we introduce the ring overlay for collaborative coordination (ROCC), which offers a fair, reliable coordination service for wireless environments. ROCC specifically addresses wireless application situations requiring group coordination. ROCC is influenced by traditional token ring and overlay communication protocols but leverages unique properties of the broadcast nature of the wireless environment to improve performance. The result is a minimal, fair, dynamic group coordination service. This paper introduces the ROCC abstraction and demonstrates its potential through analysis.

#### 1 Introduction

The combination of heightened numbers of networked electronic devices and users' acceptance of pervasive computing hardware and applications has enabled new classes of distributed applications. These new applications demand increased levels of coordination as we strive to maintain connectivity in an always-on world. Collaborative applications can be envisioned in lecture halls, where students and teachers share notes and lecture information, to coffee shops, where an ad hoc group of customers can initiate a multi-player game that requires dynamically sharing changing state information.

These collaborative applications share characteristics that engender them to the creation of a generic coordina-

tion infrastructure. In addition to requiring low-level communication capabilities (typically provided by the wireless medium), collaborative applications require a group coordination component that determines how and when information is shared among participants. While existing protocol approaches can be manipulated to support collaborative applications, either their performance suffers because they are not tailored for this domain or applications have to perform extra communication tasks because the protocols are not particularly suited to the application's characteristics.

In this paper, we create a novel group coordination abstraction targeted to supporting the kinds of information sharing necessary in collaborative applications. This abstraction and middleware service, the ring overlay for collaborative coordination (ROCC), takes advantage of properties both of the application (e.g., the definition of the communicating partners) and of the physical communication channel (e.g., the wireless medium) to provide guaranteed delivery, fairness, and reduced overhead. ROCC is suited to supporting applications in which all information shared within the collaboration group must be received by all members of the group, and the ability to transmit data to group members must be allocated fairly. To achieve these goals, the ROCC abstraction leverages aspects of token ring protocols, application-level overlay protocols, and existing coordination languages.

The remainder of this paper is organized as follows. In Section 2, we provide a description of the ROCC protocol. In Section 3, we briefly describe our prototype implementation and present an analysis of its performance. Section 4 places the ROCC protocol in the context of related work, and Section 5 concludes.

## 2 Coordinating Collaboration Participants: The ROCC Abstraction

Collaborative applications must create and coordinate groups of participants. Groups must remain connected so that users can reliably exchange information necessary to support collaborative activities. Within any single collaborative application, multiple groups may exist and must be maintained independently. In a classroom, one group may include all students and the teacher; other groups may support teams working together on class projects.

As wireless computing continues its amazing acceptance rate, ad hoc networks will be increasingly used to support collaborative applications. Ad hoc networks form opportunistically in response to devices' movements in physical space. When devices are close enough to create a wireless link, a new connection is added. In such environments, existing communication protocols provide minimal point to point and multicast abstractions. The latter can support coordination among collaborative groups when all members of the group have joined the multicast group, but the implementations in ad hoc networks incur significant overhead, as they are based on protocols that functioned well in networks with significant centralized infrastructure but are not well suited to dynamic networks. In addition, when many participants try to send information simultaneously, such communication strategies can result in significant contention for the wireless medium, requiring additional negotiation protocols for group communication.

The principle motivation of this work is to create a communication abstraction that directly reflects the requirements of collaborative applications in dynamic environments. Our approach raises the level of abstraction for the application developer by directly reflecting the structure and communication pattern of the group. The fundamental structure of our abstraction is a ring overlay, reminiscent of token ring networks. Each node in the collaboration group can exchange information only when it possesses the token. Imposing such an overlay gives each group member a chance to transmit and, as described below, can aid in ensuring that every member receives every other member's transmissions.

Figure 1 depicts such an overlay and shows that it may not necessarily directly reflect available physical connections. Instead, the overlay's implementation may use multiple hops in the physical network to provide what appears to be a single

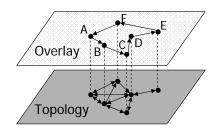


Figure 1. A sample network demonstrating connectivity between nodes.

hop connection in the overlay. If the ring does reflect a subset of the network's true topology, we avoid unnecessary transmissions, which can serve to increase efficiency and speed. Overall, providing the application with the appearance of this ring structure directly matches communication capabilities to collaborative applications' expectations of underlying group communication schemes and prevents collaborative application developers from having to awkwardly manipulate existing protocols to achieve their goals.

In the remainder of this section, we describe the ROCC model. We first discuss how the ring is used to support group coordination. In our subsequent discussion of ROCC, we assume the presence of only one application group. If multiple application groups exist, we assume the ROCC overlay runs separately for each one, on a different (non-interfering) channel to prevent collisions between the groups. For now, we assume the assignment of these channels to groups is performed out of band; future work will investigate how to handle dynamic channel assignments.

#### 2.1 Basic Operation in the Ring Overlay

Before looking at how ROCC constructs and maintains its overlay, we first look at how collaborative applications use ROCC to ensure fair and reliable collaboration. ROCC establishes and maintains an overlay on top of the true topology; in ROCC, this overlay is presented to the application as a directed token ring network. Once the node is part of a ring, the node can communicate only when it possesses that ring's token. By imposing ring-like behavior, ROCC significantly reduces the likelihood of competition for the physical medium and the potential for interference between transmissions from members of the same group.

In ROCC's overlay ring, each node has a predecessor (the previous node in the ring) and a successor (the next node in the ring). When a ring consists of only a single node, the node is its own predecessor and successor. It is important that nodes adhere to this structure even in the simplest cases to ensure that adding and removing nodes can be done smoothly.

The goal of data transmission in ROCC is to ensure that after each complete token rotation, every node in the ring has the exact same picture of the data as every other node. When forwarding the token, each node forwards a list of all other nodes in the group. Token carrying messages are structured in the following format:

 $\langle ring\_address, source, destination, participants, data_i \rangle$ 

The *ring\_address* identifies the ring, while the *source* and *destination* addresses indicate the sender and targeted receiver of this packet. The *participants* list contains the identities of the members of the ring. This portion of a ROCC message is therefore variable in length, depending on the number of ring participants and has the following format:

 $participants \equiv \langle node\_address_1, \dots, node\_address_n \rangle$ 

The final portion of the message,  $data_i$  contains this round's data for the sending node (i.e., node i), if the node has data to send

To optimize its performance, ROCC takes advantage of the inherent broadcast nature of wireless links by storing data contained in overheard packets. These overheard packets include other nodes' token exchanges, which contain the sending node's data. When a node receives the token, it compares the token's list of ring participants to this buffered overheard data. The token holding node then proactively solicits any missing data from its predecessor. When the predecessor sends this missing data, this transmission may also be overheard by other nearby nodes, potentially further reducing the overhead. Consider Figure 1. If node A transmits data within its token message to its successor (B), D and F will overhear the message because these nodes are within communication range of A. Since C is not within range of A, when C receives the token, it will request A's data from B (its predecessor). Node D will not have to request this same data since it overheard the original transmission. Especially in densely connected networks, this overhearing drastically reduces the communication overhead and, consequently, the speed at which the token can circulate. This is particularly practical in collaborative applications that demand that every node receive every other node's data transmissions. Implementing such a situation using traditional communication protocols at the application level results in numerous retransmissions due to a significant potential for collision, as evaluated in the next section.

If a packet that a node overhears contains a ring address other than the node's ring address, the node remembers this new address, and it raises an internal flag to merge the ring with this newly discovered ring. After receiving the token but before updating the neighbor information as described above, the node begins the ring join phase, which will be described in the next subsection.

The ROCC protocol as described above does not completely prevent all collisions, as simultaneous transmissions in neighboring rings may compete with each other. In overhearing these transmissions, ROCC nodes can discover new nearby rings; the detriment is the expense of the overhead of recovering from collisions. However, the degree of collisions is significantly decreased in comparison to straightforward contention for the media; this aspect will be analytically evaluated in Section 3.

## 2.2 Building Ring Overlays

Because a node by itself is also a ring, adding a new node to an existing ring and joining rings together uses the same process. To discover a new ring, a node within a ring must receive or overhear a transmission from a nearby ring. Because rings containing only a single node do not require data

transmission, such self rings periodically signal their presence with a beacon. Nodes in rings with more than one node do not have to do this because their standard transmissions can be overheard by nearby nodes. When a nearby node hears either type of transmission (a beacon from a single-node ring or a standard packet from a ring other than its own), it sets a flag to join the neighboring group.

Within the standard ring operation described above, each node will eventually receive and acknowledge reception of the token, at which point it checks the join flag. If the join flag is set, the node sends a join request to the neighboring ring. This join request contains the sending node's address, ring address, predecessor, and successor, and the destination ring's id. Since only a ring's token holder is allowed to transmit information, the receiving node in the newly discovered ring must also hold its ring's token for a join operation to commence. For this reason, the join request is followed by a brief delay to give the node in the neighboring ring some time to capture its ring's token. If the neighboring node does not reply within the allotted time, the requester returns to normal operation by transmitting data and forwarding the token.

This join process does not ensure that nearby rings join in a timely fashion, but it avoids the deadlock that arises when a node in one ring has blocked its ring to start a join, while a different node in the target ring has also blocked its ring to start a (different) join. This could also be avoided using extra control messages, but these messages increase coordination overhead and the possibility of collisions.

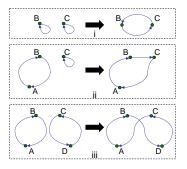


Figure 2. The three possible configurations for B joining C: i) single node to single node, ii) single node to group, iii) group to group.

If the discovered node in the neighboring ring captures the token in time, it will send an acknowledgment with its predecessor and successor. At this point, the ring that has the lower ring identifier becomes the control point for the join operation. The control node swaps the successors from the joining nodes by sending update messages to the affected nodes (both the new successors and new predeces-

sors). Figure 2 demonstrates the three cases that can occur during the join operation. In all of these cases, assume nodes B and C moved within communication range of each other. Each case exchanges B and C's successors. The third case is the general case for joining two rings, since there can be arbitrarily many nodes between A and B and between D and C. After completing a join, the node that had to change its ring address (*not* the control node) sends an update message to its original ring.

#### 2.3 Maintaining Ring Overlays

When the coordination task ends, group members will start to depart, and the ring will begin to dissolve. Nodes intending to disconnect from the group will announce this intention before simply disconnecting. When this occurs, the departing node handles its own departure by setting up a connection between its predecessor and its successor. Such an announced departure is the best case, but since ROCC must also handle unpredictable cases that result from device mobility and the unreliability of the wireless links.

**Unannounced Disconnection.** When a node departs from the ring unexpectedly, the network must restore the ring connectivity and gracefully discard information related to the departed node's participation. In ROCC, a departed node's predecessor will eventually attempt to forward the token to its successor (the departed node). When the successor does not acknowledge receipt of the token, the sender assumes the node has departed. Because the sending node has complete information about the group members and their order (given the information stored in the token), the sender can simply select its successor's successor and attempt to send the packet directly there. If this node acknowledges receipt of the token, it sets its predecessor to the sender, and the sender sets the node as its new successor. Because ROCC relies on an underlying routing protocol, it does not matter if this new successor is directly connected to the sending node or not; the underlying communication substrate will handle the end-to-end connections. If the successor's successor is unavailable (i.e., it has also been disconnected) the node holding the token simply continues down the list of the ring's participants (in order) until it finds a node that is responsive.

Lost Ring Address. ROCC is almost completely decentralized. However, to ensure that ring addresses are unique, the address assigned to the ring needs to be the address of one of the nodes in the ring. When this node departs unannounced, this ring address needs to be updated. In the process described above for handling unannounced disconnections, if a node discovers that it has lost its successor *and* that successor's address is also the ring address, the node sends a notification around the ring to update ring address before passing the token along to the next neighbor.

Lost Token. The last and most difficult case to handle occurs when a node disappears while it is holding the token. In handling this case, we must balance the overhead of reorganizing the group with the increased latency incurred by the increase in the amount of time the group does not communicate at all. If a node does not receive the token in twice its estimated token rotation period (based on the average time it took the token to rotate for the past rotations), it pings its predecessor. The node continues to query successive predecessors until it either finds one alive or finds itself. If this process does not locate the token, it is assumed that the token disappeared with one of the departing nodes, and the remaining nodes coordinate to generate a new token.

## 3 Comparison of ROCC and Multicast

To gauge ROCC's usefulness as a coordination middleware service, we analytically compared it to another coordination service, wireless multicast. Variations on wireless multicast (such as MAODV [6]) are commonly used to support

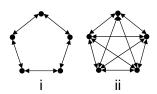


Figure 3. The two network topologies examined analytically

coordination middleware. In our analysis, we compared the latency and overhead of the two approaches. To bound our analysis, we compare the approaches' performance for two types of networks (as shown in Figure 3): a network in which each node is connected only to two other nodes, and a fully connected network in which every node is directly connected to every other node. The picture shows five nodes in each network; our analysis varies the number of nodes in the network from 2 to 20. These represent two fairly extreme cases; we expect that results for other types of topologies lie in between. We expect that ROCC can take full advantage of overhearing optimizations when there are several overlapping connections, so we expect the second type of setup to fully demonstrate ROCC's potential. In environments where collaborative applications are likely, we expect networks with several redundant links to be common.

Both ROCC and basic multicast run with the support of media access control (MAC) protocols. We assume that both ROCC and multicast use the 802.11 MAC protocol without the RTS/CTS exchanges that are commonly used to reduce collisions. We omit RTS/CTS exchanges because they have been shown to provide no benefit to throughput in wireless ad hoc networks (and may in fact be detrimental) [7] due to the fact that a node's interference range is much larger than the transmission range over which the

RTS/CTS functions. We assume a bandwidth of 2Mbps, and we evaluate the latency and overhead for each approach for an entire "round" of collaboration. We assume an active collaboration activity in which every participant has (a 1KB piece of) data to send, and we determine the amount of time and overhead involved in ensuring that every participant has every other participant's piece of data. For now, we evaluate only the costs of participating in the ring; future work will extend this analysis to include situations in which nodes are added and removed from the group.

Figure 4 shows the total data sent in the two topologies. ROCC generates far fewer bytes than a multicast protocol; this can be explained by two factors: ROCC eavesdrops to help reduce the amount of coordination necessary, and ROCC transmissions do not experience collisions. As described in the previous section, upon receiving the token a node using ROCC will request information that it has not already heard. In the case of full connectivity, a node does not have to request anything because it has already overheard all of the other participant's transmissions. When the network is less than fully connected, ROCC follows the same procedure; however, each node will request the missing information before its own data and token to the successor. Multicast protocols require each node to send one message to every member of the group along a constructed multicast tree. As a result, multicast protocols generate more mes-

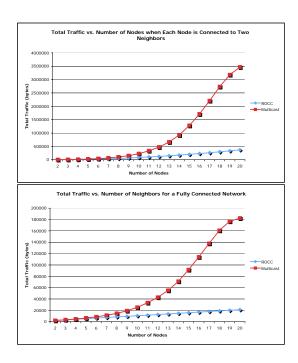


Figure 4. The total amount of traffic generated by ROCC and multicast for each network configuration.

sages as the number of connections decreases, though the messages ROCC sends are likely to be larger (since they contain an aggregation of several nodes' data items). Multicast encounters collisions when neighboring nodes attempt to transmit at the same time. ROCC forces nodes to take turns, drastically reducing the possibility of collisions. We model only the interference caused by direct neighbors in the multicast case; in truth, neighbors multiple hops away can also interfere [7], further degrading multicast performance.

Figure 5 shows the delay between when a node sends its data until every other node in the network receives that data. To send a single node's data to the entire group in a sparsely connected network (e.g., one shaped as a ring), ROCC must pass the token to all members in order. Multicast transmits data directly to all members—the only slow down incurred is due to collisions. However, in a fully connected network, all nodes overhear a node's original data transmission. On the other hand, to send one node's data to everyone in a fully connected network using multicast will generate collisions proportional to the number of nodes within communication range. As the number of nodes increases, the amount of collisions forces many retransmissions that greatly slow the network. This demonstrates the benefit ROCC achieves due to overhearing. In collaborative application domains, it is likely that group participants are nearby (e.g., in the same classroom), so the likelihood for redundant connec-

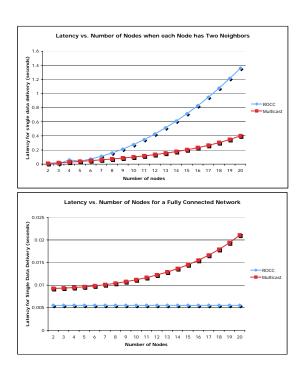


Figure 5. The latency for one node's data to reach the entire network.

tions is high. ROCC takes advantage of these redundant connections, while traditional approaches such as multicast are crippled by them.

#### 4 Related Work

ROCC provides high-level coordination constructs built on underlying communication protocols that collaborative applications can use for timely, reliable group communication. ROCC is inspired by token ring protocols whose goals revolved around mediating access to a shared medium. The Wireless Token Ring Protocol (WTRP) [2] brought token rings into the wireless domain. WTRP builds on 802.11, and it aimed to mediate access to the wireless medium. While WTRP provides inspiration for ROCC, the fundamental goal is different. ROCC relies on underlying protocols for medium access control and instead aims to provide coordination constructs tailored for collaborative applications. As such, ROCC aims to ensure reliable delivery to every group member while existing token ring approaches are tailored to supporting unicast communication.

As a middleware service for coordination, ROCC has the potential to aid existing coordination middleware. For example, the LIME middleware [5] uses basic multicast communication to ensure delivery among the members of a LIME coordination group. Replacing multicast in LIME has the potential to offer decreased overhead and increased timeliness for group-wide communication. In the TOTA middleware [4] propagation rules require a distributed overlay data structure to ensure that tuples are received at every member of a group of TOTA nodes. Employing ROCC in this situation may be able to provide an abstraction on which this distributed data structure can be implemented. Decentralized publish-subscribe systems [1] connect groups of publishers and subscribers to ensure that a publisher's event is delivered to every registered subscriber. When the rate of publication is high, ROCC can offer significant benefits to such applications by ensuring this delivery with minimal overhead. Finally, middleware for collaborative applications [3] often entail group definitions in which groups of collaborators should maintain a consistent overall picture. Using ROCC as the group support protocol in such middleware provides a clean abstraction for this grouping mechanism, while also offering the benefits of reliability, timeliness, and decreased overhead.

#### 5 Conclusions

We have presented a middleware service, the ring overlay for collaborative coordination (ROCC), which we designed to support the kind of information sharing found in collaborative applications executing in modern wireless environments. ROCC combines knowledge about applications and the communication channel to create a decentralized, efficient coordination service for wireless environments. Since ROCC builds on token rings, we benefit from the fairness inherent in turn-taking schemes and at the same time reduce message collisions. An analysis of ROCC shows that the protocol outperforms existing coordination services with lower overhead and decreased latency. These results demonstrate that, as applications' demands for coordinated activity continue to grow, we must adopt efficient coordination abstractions that can ease the development task by raising the level of abstraction and can inherently lead to efficient implementations. Given the feasibility of the ROCC approach demonstrated in this paper, future work will provide further analysis of dynamics associated with ROCC, comparisons through simulation, and the encapsulation of ROCC as a standalone coordination middleware service that can be incorporated into both middleware and application solutions.

## Acknowledgments

The authors would like to thank the Center for Excellence in Distributed Global Environments for providing research facilities and the collaborative environment. This work was funded, in part, by the National Science Foundation (NSF), Grant # CNS-0620245. The views and conclusions herein are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

## References

- [1] A. Carzaniga, D. Rosenblum, and A. Wolf. Achieving scalability and expressiveness in an internet-scale event notification service. In *Proc. of PODC*, pages 219–277, 2000.
- [2] M. Ergen, D. Lee, R. Sengupta, and P. Varaiya. WTRP: Wireless token ring protocol. *IEEE Trans. on Vehicular Technology*, 53(6):1863–1881, November 2004.
- [3] S. Holloway and C. Julien. Developing collaborative applications using sliverware. In *Proc. of CoopIS*, pages 587–604, 2006.
- [4] M. Mamei and F. Zambonelli. Programming pervasive and mobile computing applications with the TOTA middleware. In *Proc. of PerCom*, pages 263–273, March 2004.
- [5] A. Murphy, G. Picco, and G.-C. Roman. LIME: A middleware for physical and logical mobility. In *Proc. of ICDCS*, pages 524–533, April 2001.
- [6] E. Royer and C. Perkins. Multicast ad hoc on-demand distance vector (MAODV) routing. In *Proc. of WMCSA*, 1999.
- [7] C. M. Wu and T. C. Hou. The impact of RTS/CTS on performance of wireless multihop ad hoc networks using ieee 802.11 protocol. In 2005 IEEE Int'l. Conf. on Systems, Man and Cybernetics, volume 4, pages 3558–3562, October 2005.