Enabling Disconnected Communication

Amy L. Murphy and Xiangchuan Chen
University of Rochester
Computer Studies Building 734
P.O. Box 270226
Rochester, New York, USA 14627
+1 716 275 1420
{murphy,chenxc}@cs.rochester.edu

ABSTRACT
Recent technological trends in the miniaturization of computing devices and the availability of inexpensive wireless communication have led to an expansion of effort in ad hoc mobile computing. In this environment, interactions are transient, computations become highly decoupled, and communication is unpredictable. Much research energy is focused on providing the same models of communication in this environment as exist in £xed networks, focusing on routing protocols for message delivery within connected subsets of hosts. While this work is crucial, it does not address the possibility of communication across disconnected clusters, taking advantage of the movement of hosts from one cluster to another, and their ability to carry messages for hosts in their destination cluster. In this paper, we de£ne the parameters for this model of communication and provide an outline for a delivery protocol in this disconnected environment.

Keywords
Mobile computing, message passing

1 INTRODUCTION

The development of compact computing devices such as notebook computers and personal digital assistants allow people to carry computational power with them as they change their physical location in space. The number of such components is steadily increasing. One goal, referred to as ubiquitous computing, is for these devices to become seamlessly integrated into the environment until we are no longer explicitly aware of their presence, much the way that the electric motor exists in the world today. Part of enabling this vision is coordinating the actions of these devices, most likely through wireless mediums such as radio or infrared.

Ad hoc mobility is an extreme model of mobile environments in which no £xed infrastructure exists to support communication. In other words, the distance between hosts determines connectivity and as components move, the network is continuously reshaped into multiple clusters, with connectivity available within each partition but not across partitions.

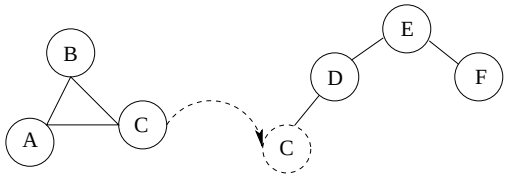

Figure 1: A sample network with six mobile components in two clusters. Dark lines indicate direct connectivity between devices. Dashed lines show the con£guration after node C moves from one cluster to the other.

Freeing mobile users from a £xed infrastructure makes the ad hoc network model ideal for many scenarios including systems of small components such as sensors with limited resources to spend on communication, disaster situations in which the infrastructure has been destroyed, and for settings in which establishing an infrastructure is impossible as in a battle£eld environment or economically impractical as in a short duration meeting or conference.

Much effort has been invested in developing protocols for point to point and multicast communication among hosts in ad hoc networks [1, 4]. In many ways, this work mimics routing algorithms in the £xed network, with the primary difference being the discovery of the path from the source to the destination only taking place on demand, as opposed to £xed routing strategies which build routing paths even if there is no traf£c in the network. This approach in ad hoc environments comes from the observation that the network of connected hosts is constantly changing and pre-computing routes which may never be used unnecessarily consumes costly wireless resources. The focus in these message delivery protocols is on allowing transitive communication of a packet through nodes which are not the destination. For example, Figure 1 shows an ad hoc network. Even though hosts D and F are not directly connected, any messages sent between them can go through host E, with E effectively playing the role of a router for the ad hoc setting.

All of these protocols work at the packet level, and deliv-

ery is only possible if a path exists from the source to the destination for a period of time long enough to discover the route and send the packet. We propose a new communication model, *disconnected communication*, which removes the assumption of connectivity from message delivery, allowing a message to be passed from one host to another even if the source and destination are never connected either directly or transitively. It is designed to act as a middleware, exploiting existing wireless protocols while providing a useful abstraction for message delivery to applications. Our strategy is most applicable to applications where a high degree of asynchrony in message delivery is tolerable.

The remainder of this paper provides a more precise de£nition of the problem of disconnected communication and our proposed solution (Section 2), explores some of the technical issues which must be addressed as well as an implementation path (Section 3) and provides a brief discussion and conclusions (Section 4).

2 DISCONNECTED COMMUNICATION

The communication model, termed disconnected communication, which we describe in this paper is intended to provide a new level of asynchronous communication not available in lower-level mechanisms currently being explored. In protocols such as Dynamic Source Routing (DSR) [2], the default procedure to send a packet initially attempts to £nd a route to the destination. If no such path can be found within a short timeout of the request for transmission (500ms), the packet is delayed for twice that timeout. This process repeats for at most thirty seconds, after which the delivery fails.

Our idea is simple: Rather than simply fail when immediate delivery is not possible, move the entire message (not a single packet) to another host as close to the destination as possible, where *closer* is de£ned to be a host which is likely to be in contact with the destination earlier than the source. For example, in Figure 1, if host A wants to send a message to host F, a delivery scheme such as DSR will fail. If, however, A can determine that C will soon be in contact with F, A can pass the message to C (using standard DSR), and C can pass the message to F after it migrates to the new cluster. We do not limit ourselves to a single intermediate hop as this example suggests, but rather provide a general mechanism to move a message through the network, constantly getting *closer* to its destination. Each time the message is transferred, the new host effectively becomes the sender, taking on the responsibility of propagating the message closer to the destination.

A conference provides a meaningful environment for applying disconnected communication. Often, attendees wish to exchange messages with one another and although email is available, it is centralized and not frequently accessed. Disconnected communication is immediately applicable to move messages through a system of PDAs, enabling attendees to arrange meetings and share comments about the presentations without requiring them to meet or check central locations in order to plan or exchange information.

One challenge to making the protocol function is determining which host, if any, is closer to the destination. For this, we turn to application-level knowledge about host movement and connectivity patterns as well as some general characteristics about hosts in ad hoc networks.

One way to determine if a host is *closer* to the destination is to allow the source to de£ne a sequence of hosts which can serve as the intermediate hops. In the example, A may speci£cally list the sequence $\{C, E, F\}$, meaning that as the message moves through the system, if a node which is closer to the destination in the sequence is in the cluster, the message should be transferred to that node. Hosts can be skipped, and while a message is at an intermediate host, it is not visible to the applications on that host. In other words, the fact that a host is acting as a router for messages is transparent to the applications running on that host. In the example of Figure 1, the message will not be transferred to node E because the destination, F, is immediately accessible when C migrates (it will pass through E as a consequence of the ad hoc routing protocol, but at not point will E be responsible for £nding the next hop at the level of our protocol). Note that a message should never move to a host which is earlier in the sequence from its current location.

While such an explicit list of intermediate hosts is simple to use, it is not always reasonable to expect the source to provide such a list. Instead, the sender can rely on other properties common to hosts in the ad hoc setting. For example, the personal digital assistant is a common device in ad hoc networks, and one typical use of a PDA is for maintaining a user's calendar. Our routing protocol can extract information from the calendar to determine a probability that a speci£c destination host will be reachable within a given time frame. This can be based on a known shared meeting between the hosts, or the probability of meeting a host which is closer to the destination in a future scheduled appointment.

Alternately, a mobile host can track its history of connectivity patterns from which the future likelihood of meeting an individual can be extracted. This approach is appealing because it does not require any application-intervention to determine the next hop, but it does require that a log be maintained locally and that queries be relatively simple.

Other viable options for determining the next hop for a message include analysis of a mobile unit's attributes. For example, a next-hop calculation may take into account physical location in space, direction of travel (e.g., always trying to move a packet north and west), remaining power (e.g., assuming that a mobile unit with limited power is not going to be active for much longer and is an unwise next hop), available storage (e.g., seeking to balance the message dissemination load among multiple mobile units), or any other physical attribute.

While it is possible to design a default combination of these attributes for calculating the next hop for a packet, we believe that knowledge provided by the application is critical in de£ning the correct balance of properties. Therefore, we allow the application programmer to specify a function which, given the attributes of the system (ranging from a calendar to power level) and a host computes a value utility of the host as the next-hop for a message with a given destination. Putting the actual de£nition of the function into the hands of the programmer not only makes the system more ¤exible, but allows the user to manage the complexity of one of the key components, namely the next-hop calculation. After the utility of each host is calculated, the source selects the host with the highest utility and hands off the message.

The next question is how to actually compute these values for every host in the cluster. If the sender knows all of the members in its cluster, as well as all attributes about these host, the analysis can be done centrally. However, knowing all of the group members has already been shown to be a dif£cult problem [5] with moderate overhead, and to add a requirement that all host attributes be passed with the group membership adds unreasonable message overhead.

Our solution is to de£ne a next-hop discovery protocol which distributes the application-de£ned function to the hosts in the cluster utilizing a standard ad hoc broadcast protocol to reach all connected hosts in a cluster. The function is evaluated at each host and if the computed value is above a source-de£ned threshold, a message is sent to the source, indicating the host and its value. The source collects the responses, determines the best, next-hop, and hands off the message. Passing around a function is appealing for a number of reasons. First, the computation is spread among multiple processors, increasing parallelism and fault tolerance. Additionally, the use of mobile code technologies to distribute a user-de£ned function greatly increases the ¤exibility, but we must weigh the cost of the overhead of the message carrying the code with the savings gained by not needing to propagate the host attributes throughout the system. After the function is evaluated and the value is calculated, if it is above a certain threshold, it is sent back to the source in a unicast message.

To increase the reliability that a message will reach its target destination, the source can specify that the message is to be sent to multiple *next hops* simultaneously. In other words, instead of choosing only the most optimal next-hop based on the utility, a host can choose the top n hosts. While this replication may increase the chances of successful delivery, it also introduces complexity to the protocol to deal with duplicate messages moving through the system. There is a need to control both the level of fan-out of the message as well as provide a mechanism to clean up old copies.

3 TECHNICAL CHALLENGES AND IMPLEMENTATION PATH

While this new form of message delivery provides interesting opportunities for communication, the ad hoc network presents certain challenges that must be addressed during the development of the protocol. As we have mentioned, one of the typical concerns in wireless communication is the energy spent on communication as the number of packets sent and received consumes battery power on resource limited mobile devices. Therefore, it is essential to consider the overhead imposed by our algorithm and the effect it can have on the performance of the devices supporting it.

One of the key parameters to ensuring that our algorithm functions correctly is determining the frequency at which messages held at each node are reexamined and the network queried for a new next-hop. If this value is too large, we waste valuable network resources with redundant queries. If this value is too small, we may miss the arrival of an *optimal* next-hop for a message. One way to balance these con¤icting concerns is to analyze the variance in the ad hoc network. If the network is changing rapidly, with new nodes arriving and departing quickly, then the extra bandwidth to test for a new next-hop is more justi£ed. In a more stable network, it is not necessary to check as frequently for a new next-hop.

This brings up the question of how to calculate the degree of change in the network without introducing additional message overhead for this computation. For this, we can take advantage of a low-level protocol present in wireless hardware devices which periodically broadcasts the identity of the host on a well-known broadcast channel. This message, commonly referred to as a HELLO packet, is never forwarded beyond the £rst hop, and serves to identify all neighbors with whom direct communication is available. One possibility for determining the change in the network is to collect the direct neighbors whose HELLO messages are received during an interval of time and compare the list with the hosts detected in the previous interval. While this accurately measures the degree of change in the neighbors, it may not accurately re¤ect the dynamicity of the remainder of the cluster.

For example, in Figure 1, if D, E and F are relatively stable, then D will perceive little change in its neighbor list and conclude that the cluster is stable. If, however, node F was physically located near a door with a stream of hosts passing by, the cluster membership would be highly dynamic, F's neighbor list would be constantly changing, and D's interpretation of cluster stability would be incorrect.

To counter this, we propose adding a single variable to the HELLO message indicating the current perception of the activity of the network. This value is calculated based on the direct information collected from the change in neighbor lists, as well as the perception value of the hosts in the neighbor list. For example, if F has a high value, this will be passed along to E through F's HELLO message, causing E to increase its perception of dynamicity. When this value increases, it will be passed along to D, similarly increasing its perception of the dynamicity of the cluster.

Several questions are immediately apparent, and still under investigation. First, what is the optimal period for detecting changes in the neighbor list? Should this value somehow be related to the perception of the dynamicity of the cluster? How quickly should nodes adapt to larger values observed in the HELLO packets of their neighbors? Put another way: what is the optimal function for calculating the dynamicity value? Is it reasonable to consider a function which quickly ramps up to higher values, but more gradually decreases? The answers to these questions will be the subject of future study and will most likely involve careful de£nition and evaluation of target application environments.

One problem which may arise is looping in the path that a message takes from the source to destination. In some cases, a loop may actually facilitate more timely delivery of the message, but it is possible for a message to enter a tight loop where each time the next hop is determined, it bounces between two hosts for whom the utility is changing quickly. This situation unnecessarily wastes bandwidth and should be taken into consideration.

Another practical consideration is the amount of buffer space available on the mobile hosts for storing messages in transit. We cannot assume in£nite storage capabilities, and therefore must de£ne a buffer management scheme which discards messages with some policy. The de£nition of this policy remains an open issue. Should an old packet be discarded simply because it is old, or should an old packet be kept because it is closer to its destination? Packets which have been replicated as part of their delivery can be deleted as they are likely to reach the destination through some other path, but we still need a mechanism to discriminate among multiple such messages. It is likely that a meaningful strategy can only be de£ned with respect to the speci£c application using this protocol, making it important that our implementation describe a clean API to guide the programmer in the de£nition of this function.

We must also consider that requiring the user to provide a function may be too complex, or may put too much overhead on the network. For this reason, we will investigate de£ning a default next-hop protocol calculated at a host based only on the identity of the destination host. We will also evaluate sending the entire message as part of the next-hop discovery protocol in order to eliminate the overhead of sending the utility responses back to the sender. Any host with a utility above the utility of the source will keep the message.

Our immediate plans include a prototype implementation to enable experimentation with the practicality of our approach as well as the variety of next-hop calculation routines for multiple ad hoc applications. Initially, we will leverage off of the LIME middleware [3]. LIME enables the coordination of mobile units by associating a tuple space with each unit and transiently sharing these tuple spaces when connectivity exists. Currently LIME includes a mechanism to migrate a tuple exactly one hop from its source to the destination, requiring connectivity between the source and destination to deliver the message. Our prototype is designed as a communication wrapper, allowing the programmer to specify the tuple (message), destination, and a next-hop function for describing the migration of a tuple to its destination. This extension allows the message tuple to take multiple hops through the network to reach the destination, and does not require that the source and destination ever be connected.

Once we have built this prototype and demonstrated the usefulness of the protocol, we will extract the functionality into a stand-alone message delivery sub-system and work to optimize the lower-level communications, using multicast messages where possible to distribute the request for a next-hop calculation, and exploring different ad hoc routing protocols and the support they can lend to our protocol.

4 CONCLUSIONS

In this paper we introduced a new communication model which removes the assumption of connectivity between the source and the destination. We also provided an initial protocol description which describes how multiple aspects of the ad hoc network can be combined to enable such communication. This protocol is not intended to replace existing work on ad hoc routing, but in fact is built on top of these mechanisms and extends them, enabling the new model of disconnected communication. This work is in its initial stages, yet our initial £ndings indicate that it has great potential for extending the communication patterns available in ad hoc networks.

REFERENCES

[1] S. Das, C. Perkins, and E. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *Proc. of the IEEE Conf. on Computer Communications (INFOCOM)*, pages 3–12, Tel Aviv, Israel, March 2000.

[2] D. Johnson, D. Maltz, Y. Hu, and J. Jetcheva. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks . Internet Draft, March 2001. IETF Mobile Ad Hoc Networking Working Group.

[3] A. Murphy, G. Picco, and G.-C. Roman. LIME: A middleware for physical and logical mobility. In *Proc. of the 21st Int. Conf. on Distributed Computing Systems (ICDCS)*, Phoenix, AZ, USA, April 2001.

[4] C. Perkins and E. Royer. Ad-hoc on-demand distance vector routing. In *Proc. of the 2nd IEEE Wkshp. on Mobile Computing Systems and Applications*, pages 90–100, New Orleans, LA, USA, February 1999.

[5] G.-C. Roman, Q. Huang, and A. Hazemi. Consistent group membership in ad hoc networks. In *Proc. of the 23rd Int. Conf. in Software Engineering (ISCE)*, Toronto, Canada, May 2001.