Algorithm Development in the Mobile Environment

Amy L. Murphy
Dept. of Computer Science, Washington University
St. Louis, Missouri, USA
alm@cs.wustl.edu

April 19, 1999

1 Introduction

Mobile computing is emerging as a novel paradigm with its own characteristic problems, models, and algorithms. Much effort is being expended on integrating mobile units with fixed networks, providing bridges to connect wireless to wired. The result is a fixed core of static nodes and a fluid fringe of mobile units, an overall system similar to the cellular telephone network. With this graph as a foundation, our approach to mobility treats the mobile units themselves a persistent messages moving through the network. This model allows algorithms from traditional distributed computing to be directly implemented in the mobile environment. However, it has been noted [2] that because of the unique properties of mobility such as limited bandwidth and disconnection, it is not practical to do such a direct translation. Our direction is fundamentally different, and rather than directly apply the distributed algorithms for their original purpose, we adapt them to solve issues unique to the mobile environment. Specifically, we propose two algorithms for message delivery to mobile units, the first based on distributed snapshots and the second on diffusing computations.

Our interest goes beyond this view of mobility to the more novel model of ad hoc mobility, or groups of transiently associated mobile units which do not depend upon any fixed support infrastructure. Connection and disconnection is controlled by the distance among units and by their willingness to collaborate in the formation of a cohesive, albeit transitory, community. By removing the fixed network, algorithms which rely on it, such as those described in the previous paragraph, are no longer applicable. Our primary goal in this area is to provide for the rapid development of dependable applications. To achieve this goal we take a two phase approach, the first by developing a fully specified shared memory model for coordination of ad hoc components and the second by providing a set of algorithms to work in conjunction with this model.

Our work in both models of mobility complements itself well, with our established success in the first providing a solid stepping stone for our efforts in the second. This abstract briefly describes the previously mentioned message delivery algorithms, then outlines our ongoing and future work in the ad hoc environment.

2 Base station mobility

Within the base station model of mobility, we propose two algorithms for delivery of a data message to a mobile unit. The first is based on the search approach in which the system does not keep track of the mobile unit as it moves, but rather searches for it each time a message is to be delivered. This model works well in environments where movement is rapid in comparison to the number of messages being sent. The second algorithm uses a distributed tracking approach where the nodes of the system keep information relevant to the current location of the mobile unit.

Snapshot and search. Our first algorithm adapts the distributed snapshot algorithm by Chandy and Lamport [4] whose goal is to provide a consistent view of the state of a network of nodes and channels. The state consists of the process variables and any messages in transit among the nodes. This is accomplished by sending a set of control messages to neighboring nodes, informing them that a snapshot is in progress,

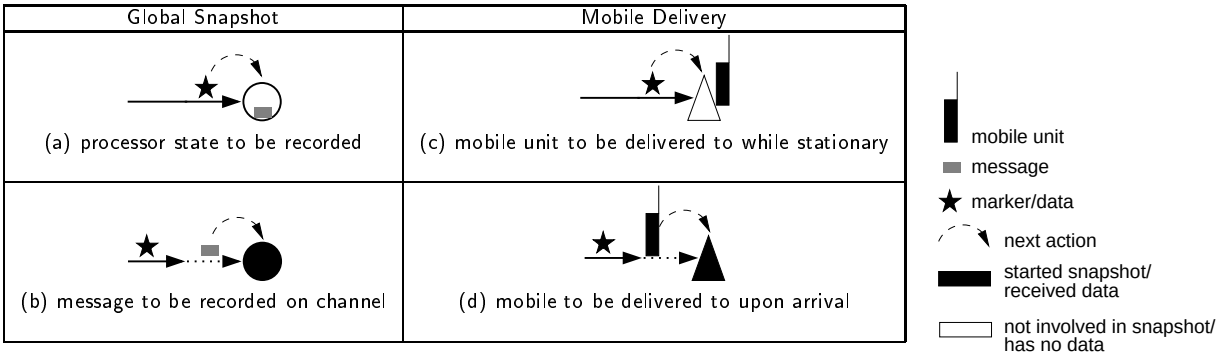

Figure 1: Translation of concepts from global snapshots into mobile delivery. The curved arrow shows the processing of an element from a channel while the text describes the action triggered by such movement.

and initiating local snapshots. The main property of snapshots that we will exploit is that every message will appear exactly once in the recorded snapshot state.

To move into the mobile environment, we refer to the model of nodes and channels described earlier with mobile units roaming as persistent messages. This provides the network in which the snapshot runs, however we must also describe the details to accomplish message delivery. One possible adaptation would be to analyze the recorded snapshot to identify the location of the mobile unit and simply send the data to that location, however it is likely that the mobile unit will move between the time this information is calculated and the time the data is sent. Therefore, we propose augmenting the control messages with the data to be delivered and altering the local snapshot recording to perform data delivery if the mobile unit being recorded matches the destination address in the data (see Figure 1). Because each mobile is recorded exactly once in the snapshot, we are guaranteed that the destination mobile unit will receive exactly one copy of the data. It is worth noting that by simply changing the destination address in the data to a multicast address, without any changes to the algorithm, each mobile unit in the multicast group will receive a copy of the data.

Having successfully adapted the snapshot to message delivery, we approached the restriction of FIFO channels imposed by the Chandy-Lamport algorithm. Because our model considers a mobile unit to be a message that behaves the same as the control messages of the snapshot, it seems unreasonable to believe that the mobile unit can travel through space at the same rate as the message travels through the channel. Therefore we examined the handover algorithm which regulates the movement of the mobile unit and developed a simple scheme to ensure FIFO behavior. This approach is general and can be applied to any setting requiring FIFO behavior of mobile units and messages.

Diffusing computations and tracking. Our second algorithm adapts the diffusing computations model of Dijkstra and Scholten [5] to create a graph representing the region of movement of a single mobile unit [9], rather than a graph tracking the message expansion. The initial node of the distributed algorithm becomes the home node of the mobile setting and we define a node to be *active* only when the mobile unit is present, and *idle* at all other times. Therefore, as the mobile unit moves and as communication resources permit, the pruning rules can reduce the tree to only the path from the root to the mobile unit. To deliver a data message in this setting, the data is sent initially to the home node, and propagated down the children links until the mobile unit is located. A second cleanup phase is necessary to delete the data at each node.

Although this algorithm is simple to understand and easy to prove by mapping the properties of diffusing computations into the mobile environment, the needless propagation of data along paths that have not yet been cleaned up is a source of inefficiency. Therefore we developed a related algorithm where each node in the tree has at most one child (yielding a unique path from root to mobile), and rather than deleting idle nodes that have no children, an idle node is deleted when it has no parent. Not only does the resulting algorithm increase efficiency with respect to stored data during delivery, it is also more amenable to route optimization, a future direction for this research.

Future directions. Our primary contribution is a new methodology for algorithm development, namely the treatment of the mobile unit as a message in the distributed setting. In addition to physical mobility, these algorithms are also amenable to logical mobility where mobile agents roam a fixed network of servers. One specifically identified problem in mobile code is that of orphan detection[3, 12]. The described approaches thus far either have high overhead to maintain connectivity to the parent, or do not provide strong guarantees that orphans will be deleted. Applying our algorithms addresses both of these concerns. We are in the process of incorporating these algorithms into a mobile code toolkit to test their potential to provide orphan detection and other general coordination of mobile agents.

3 Reliable rapid application development in ad hoc mobility

Base station mobility provides an introduction to the issues of mobility such as transient connections, limited bandwidth, and varying rates of mobility, while ad hoc mobility takes an additional step by removing structure and replacing it with a high level of transiency. Additional complexity arises from issues such as the constantly changing connectivity, changing participants, disconnection and partitioning of the network, limited resources such as bandwidth, computation and power, and unannounced disconnection of mobile components. A distinguishing feature of our approach to these issues is the attempt to reduce all interactions to a coordination problem. Whenever possible, abstract coordination constructs will present the individual mobile units with the impression of a continuously changing computing environment thus hiding many of the mechanics of movement and the true nature of other units within range. This strategy, made popular by Linda, facilitated temporal and spatial decoupling of processes in the parallel computing domain and simplified program development. Our work transfigures this basic idea for use with ad hoc networks. Each mobile unit will be working on a single, shared global virtual data structure continuously reshaped by the movement of units in space, the transfer of agents among communicating units, and changes in the quality of service achievable in a particular configuration. In this direction, we present a tool for coordination based on the tuple space model of Linda [6]. This existence of this tool then provides an environment for developing and testing algorithms for the ad hoc environment.

Coordinating ad hoc mobile units using Linda. In general, Linda provides coordination among components through a shared, globally accessible, persistent tuple space. Coordination is achieved as multiple processes access this structure by outputting, taking, or reading tuples. Processes are transient, but the data remains persistent. Such persistence is not possible in the ad hoc mobile environment where no single shared memory exists and connections among independent components vary in both time and space, while constantly requiring coordination. Therefore, LIME (Linda in a Mobile Environment) [10] is a model which demands weaker constraints on the tuple space, allowing reconfiguration of its contents as connectivity changes.

In LIME, an interface tuple space (ITS) is bound to each logical mobile agent component and is accessed using the usual Linda primitives. When the mobile components are colocated, the tuple spaces are transiently shared and all tuple space accesses, including pattern matching for reading and removing data are done on the now shared data space (see Figure 2). Additional primitives with extensions for location are provided to address specific tuple spaces, however the presence of the specified tuple space is dependent on connectivity. In addition to these primitives, LIME provides a mechanism to register for tuple space events, extending Linda to provide a "push" model in addition to the traditional "pull" model. The specification of LIME is given using the Mobile UNITY [11] semantics, allowing the formal verification of applications and algorithms built on top of LIME.

Because LIME provides a minimal set of established coordination primitives to the application programmer, it is anticipated that it will provide a foundation for the rapid development of dependable applications in the mobile environment.

Future algorithm work. Just as distributed computing has developed a rich set of algorithms for solving recurring issues, developing a similar set of algorithms tailored to ad hoc mobility is an important area of study. Although research has been done in fixed base station mobility [8, 1] by placing the majority of the processing at the fixed stations, this schema is not appropriate where each network component is itself mobile. Therefore algorithms analogous to established distributed computing algorithms, such as termination detection, leader election, and stable property detection must be revised with ad hoc mobility in mind. Not

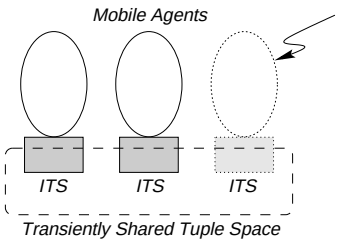

Figure 2: Transiently shared tuple spaces in LIME.

only do the algorithms themselves need to be redefined, but also weaker semantics must be formally defined to allow for reasoning about the performance of these algorithms. As applications are being developed on top of the LIME system, the necessity for these algorithms is becoming apparent. For example, in a simple round based game where all participants are known and each participant independently completes a turn, a termination detection algorithm is needed to allow the game to progress to the next round.

The ad hoc environment also demands that certain classes of algorithms be revisited from the perspective of mobile computing. For example, randomized algorithms may provide a convenient mechanism for describing probabilistic guarantees. Also, as non-determinism proved to be a useful model in defining parallelism, randomness may prove to be a useful tool for modeling arbitrary movement of mobile units. Self-stabilization algorithms in distributed computing provide guarantees conditional on the relative stability of the network. Such guarantees are directly translatable to the mobile environment when considering automatic reconfiguration after mobile unit movement. Finally, epidemic algorithms may provide insight into mechanisms to spread information from one mobile unit to another as connections change.

The LIME system employs the basic data structure of a tuple space and defines the movement of data with respect to the location of a tuple inside a tuple space. This is one possible coordination construct designed to transfer data transparently with respect to a predefined policy. Other such policies and data structures can be developed for other application arenas. For example, data replication policies play an important role in many mobile settings due to the increase in availability, performance, and dependability. Weak consistency models have been discussed in the area of fixed networks and file systems [7], however the ad hoc environment remains unexplored. Although the tuple space provides general data storage, other structures, such as trees, may be more intuitive.

Other basic properties of the ad hoc environment can be exploited when developing standard algorithms. Properties such as position in space and direction of movement lead to optimization problems such as dynamic route minimization in connected systems, boundary detection in dense communities, and connectivity preservation in autonomous systems. The issues of orphan detection also become significant when loss of units is expected as in robot systems with multiple redundant units and a built-in expected loss.

In each of these cases, solutions that favor early decisions by a subset of participants, provide stable or monotonic metrics, and can tolerate extended lack of involvement on the part of many units must be explored, formally defined, proven, and tested.

Discussion. As implementors of LIME we are able to exploit it in two specific ways with respect to algorithm development. First, during the design of the system itself, several algorithms have been incorporated, notably transaction protocols for reconfiguration. Second, our ad hoc algorithms can be implemented on top of LIME, providing us the opportunity to develop a more practical understanding of which conditional properties are meaningful, as well as extending the usability of LIME with higher level functionality.

4 Conclusions

Each form of mobility, base station and ad hoc, presents challenges simply by introducing the notion of movement into an environment where many assumptions are made based on location. By first incorporating mobility into this arena, we were able to understand the issues that distinguish mobility, and develop a paradigm for algorithm development which adapts well established knowledge from distributed computing to the mobile environment. By removing the fixed graph, and moving away from the approaches taken by

traditional distributed computing algorithms, we are forced to rethink many of the previous assumptions when designing new algorithms. Therefore, in the development of these new algorithms, one must pay careful attention to the conditional properties and environmental assumptions, and apply formal proof techniques to verify algorithm correctness. With formally defined algorithms in concert with LIME, a formally defined tool, we are moving toward our goal for rapid development of dependable applications in the ad hoc environment.

5 Acknowledgments

This work would not be possible without the support of my research advisor Dr. Gruia-Catalin Roman and collaboration with my colleague Dr. Gian Pietro Picco. Thank you.

References

[1] A. Acharya, B.R. Badrinath, and T. Imielinski. Checkpointing distributed applications on mobile computers. In *Proceedings of the Third International Conference on Parallel and Distributed Information Systems*, October 1994.

[2] B.R. Badrinath, A. Acharya, and T. Imielinski. Structuring distributed algorithms for mobile hosts. In *Proceedings of the Fourteenth International Conference on Distributed Computing Systems*, pages 21–28, Poznan, Poland, 1994.

[3] J. Baumann and K. Rothermel. The shadow approach: An orphan detection protocol for mobile agents. In *Mobile Agents: Second International Workshop MA '98, LNCS 1477*, pages 2–13, Stuttgart, Germany, September 1998. Springer-Verlag.

[4] K.M Chandy and L. Lamport. Distributed snapshots: Determining global states of distributed systems. *ACM Transactions on Computing Systems*, 3(1):63–75, 1985.

[5] E.W. Dijkstra and C. Scholten. Termination detection for diffusing computations. *Information Processing Letters*, 11(1), 1980.

[6] D. Gelernter. Generative Communication in Linda. *ACM Computing Surveys*, 7(1):80–112, Jan. 1985.

[7] J.J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems*, 10(1):3–25, 1992.

[8] J. Matocha. Distributed termination detection in a mobile wireless network. In *36th Annual ACM Southeast Conference*, Marietta, GA, April 1998.

[9] A.L. Murphy, G.-C. Roman, and G. Varghese. Search and tracking algorithms for rapidly moving mobiles. Technical Report WUCS-98-01, Washington University, Dept. of Computer Science, St. Louis, MO, USA, January 1998.

[10] G.P. Picco, A.L. Murphy, and G.-C. Roman. LIME: Linda meets mobility. Technical Report WUCS-98-21, Washington University, Dept. of Computer Science, St. Louis, MO, USA, July 1998.

[11] G.-C. Roman, P.J. McCann, and J.Y. Plun. Mobile UNITY: Reasoning and specification in mobile computing. *ACM Transactions on Software Engineering and Methodology*, 6(3):250–282, 1997.

[12] F.M. Assis Silva and R. Popescu-Zeletin. An approach for providing mobile agent fault tolerance. In *Mobile Agents: Second International Workshop MA '98, LNCS 1477*, pages 14–25, Stuttgart, Germany, September 1998. Springer-Verlag.