Ad Hoc Mobile Networking and General Mobility Issues

Ramon Lawrence
Department of Computer Science
University of Manitoba
umlawren@cs.umanitoba.ca

May 29, 1998

1 Introduction

With the increasing trends of networking and mobility come many interesting opportunities and difficult problems. Mobile computing allows a user to use a computer as if they were physically attached to a network but are actually freely moving around in the environment. Unfortunately, to support this mobility places several restrictive restraints on the system including weight and size limitations, battery restrictions, and lower bandwidth communications. Algorithms that implement mobile communications must recognize these factors and guarantee a level of usage satisfactory for the users.

The main problems associated with networking including naming and routing are even more difficult in a mobile environment as the mobile units are allowed to move. Ad hoc mobile networks are temporary networks of intercommunicating mobile units which interact without using an established connection infrastructure either because an infrastructure does not exist or it is not practical to use it. Ad hoc mobile networks are even more complex than mobile networks involving fixed stations because mobile units may act as hosts causing problems due to mobile unit migration and lesser reliability.

This paper studies ad hoc mobile networks and presents a few of the general problems associated with mobile computing. The paper is organized as follows. Section 2 provides motivation for the trends of mobile computing, and the reasons for the anticipated growth in mobile networking. Section 3 studies the problem of providing a mobile computing service based on the limitations of communication, mobility, and portability. Ad hoc mobile networks are defined in Section 4, and the problem of routing in these dynamic networks is discussed. Mobility places several requirements on application designers not found in typical fixed systems. These requirements are detailed in Section 5. Section 6 presents future work necessary in the research and industrial environments and attempts to justify that in many areas mobile computing is a unique, and possibly "new", research area. Conclusions are presented in Section 7.

2 The Demand for Mobility

In the early years of computing, users were satisfied with having a large and relatively slow machine improve their lives by performing tedious and repetitive mathematical operations. As the power and capability of the computer grew, so did the users' expectations on what it should accomplish. First, the computer migrated from the large mainframe environment to the desktop, allowing individuals to write documents, create graphs, and use the computer as an everyday household tool. Eventually, computers in isolation became insufficient as the desire for intercommunication and network access increased. With the exponential growth of the Internet, millions of computers are connected to an on-line system spanning the globe, but users continue to demand more.

Intercommunication is no longer sufficient. Paralleling the trend of cellular phones and pagers, users are now demanding the same mobility from their computers. To satisfy this demand, computers are becoming smaller and more portable. The trend is towards ubiquitous computing, where computers are seemlessly integrated into the environment. The computer is evolving into a common tool much like a wristwatch or a cellular phone but with the increased functionality of supporting network access and powerful computational abilities. Not only do these requirements challenge the hardware designers to shrink and evolve the computer into a more usable device, they require the researchers and programmers to ultimately design better algorithms and applications to support this environment. It is unlikely that the trend to smaller computers for personal convenience will reverse. Thus, to provide constant Internet access, powerful computing facilities, and portable ubiquitous computer devices for today's market requires the trends and issues involved in mobility be carefully studied and resolved.

3 The Challenges of Mobile Computing

To support mobile computing, it is necessary to have an understanding of the problems the mobile computing environment presents[2]. Many of these problems are physical relating to the restrictions on the usage and capabilities of the mobile units, but there are other problems including defining interactions with fixed stations, handling disconnections, and designing application software to handle mobility gracefully. The problems fall into three categories: communication, mobility, and portability.

3.1 The Communication Issue

The basic problem of mobile computing is to provide a mechanism for the mobile units (MUs) to communicate with each other or a fixed network. Due to the size and power restrictions of the mobile units, communication is harder than in fixed networks which have high-speed connections and an unlimited power supply.

The current methods for communication are radio, infrared, and cellular telephony which provide about 2 Mbps, 1 Mbps, and 14 Kbps bandwidth respectively. Infrared is the method of choice inside buildings but cannot be used outside because sunlight affects transmission. The bandwidth provided by any of these methods is far less than in fixed networks which can provide bandwidth up to 600 Mbps or more. Due to the low bandwidth, mobile applications and networking protocols must adapt their behavior to conserve this precious commodity.

Mobile unit transmissions are broadcast transmissions heard by anybody with the appropriate transceiver in the broadcast area. A **cell** is the area in which a transceiver can receive transmissions. Any number of mobile units may be in a cell, and the bandwidth of a cell is shared between them. Due to this bandwidth sharing, effective bandwidth in a cell is either measured in units of bandwidth/user or bandwidth/m^3. The former measure is preferred because it is independent of the number of users in a cell.

Both hardware and software techniques are used to increase effective bandwidth. Hardware techniques rely on adding additional transceivers to increase the bandwidth in an area. One way to do this is by having multiple transceivers at different frequencies covering the same area. Each frequency supports a given bandwidth, and the total bandwidth in the area is increased with each frequency. This technique is rarely used because the number of usable frequencies for mobile communication is limited. Instead, the area of each cell is reduced because the bandwidth in a cell is inversely proportional to the area covered. By reducing the cell size, the bandwidth within the cell is increased. With smaller cell size, more transceivers must be installed to cover the same area. This method is more cost effective than the previous method and uses less transmission frequencies.

Software techniques also conserve bandwidth. Compression reduces the amount of data transferred, and buffering delays transmission during peak times for better performance. Prefetching attempts to anticipate the mobile user's data requirements and gets the data before it is used. If the prefetching algorithm is correct, the data is retrieved during non-peak times and is available for the user when needed without delay. Otherwise if the data retrieved is not used, precious transmission resources are wasted on "bad guesses". Finally, delayed write-back reduces transmissions from updates by delaying the send of the update for a while. This delay allows the data to be sent during non-peak times, and it may allow the user to change his mind by either updating the data again or invalidating the previous update. In these cases, the update may not have to be sent back to the server or only need to be sent once instead of multiple times.

There are other communications issues that need to be addressed besides limited bandwidth. A mobile communication link is unreliable resulting in higher signal interference and error rates. Frequent disconnection occurs because of the unreliable links or voluntarily by the user to save power. How a mobile unit continues to operate during these times of disconnection is an open and difficult problem. Ideally, the mobile unit should continue to function but with lesser capabilities, but this is very hard to support especially when considering the issues of data sharing and management. The effect of disconnection is lessened by giving the mobile units greater autonomy. Unfortunately, this autonomy requires more complicated mobile units which contain more data and are more susceptible to data tampering. A lesser autonomous unit is very vulnerable to network disconnection and must transmit more but is less vulnerable to data loss and sharing concerns. There is a fundamental tradeoff of better disconnection handling and processing capabilities in the case of greater autonomy versus added security, simplicity, but increased communication with lesser mobile unit autonomy.

Two other important issues arise in mobile unit communications. Broadcast transmissions are a security issue because anyone within the transmission range is able to hear, and possibly process, the transmission if it is not suitably encrypted. Also, due to the mobile units' movement, they may encounter many different fixed network support stations with a variety of network protocols, capabilities, and bandwidth limitations. A mobile unit must adapt its operations to these changes in a seemless manner.

3.2 The Mobility Issue

The movement of the mobile units creates complicated problems in networking routing and management and handling of location-dependent information. A mobile unit's address changes as it moves because it is connected to the fixed network at different locations. A mobile support station (MSS) has a transceiver to communicate with mobile units and serves as an access point to the fixed network. Any packets sent to the mobile unit must be first sent to its current MSS. As a mobile unit moves, different mobile support stations are responsible for handling its transmission, and the address (of the MSS) through which it can be reached changes. These address changes are called address migration. Routing protocols must handle these changes. Current routing protocols do not handle mobility well because they were designed for a fixed network topology. Although in the proposed Internet standard, IPv6, mobility support is included. In IPv6, a trusted home agent of a mobile unit continually tracks its movement and address. Transmissions to the mobile unit are forwarded first to the home agent if the sender does not know the mobile unit's current location. The home agent forwards the packet to the mobile unit and then informs the sender of the mobile unit's location so that future packets do not have to be tunneled through the home agent.

The second issue in mobile unit movement is that the data and queries used by the mobile unit may depend on its location. If a user wants to know the location of the nearest cab or a route home, these queries can only be answered given the user's current location. Such location-dependent data is subject to degrees of precision. To answer the previous queries exactly, a high degree of messaging may be required even though "rough estimates" may be suitable. Thus, there is a lot of work to be done in query processing to determine how to use location-dependent information and how precise it must be for a given application.

A user location is privileged location-dependent information. If a user does not want to disclose his location but it is required to answer a given query, verification needs to be performed to insure this information is not used maliciously. Since the transmissions are broadcast, unscrupulous agents could intercept location information and act accordingly. For example, convicts could intercept current police car locations, and robbers could detect if a person is not home by listening to his remote communications. The use and security of location-dependent information is an important research area new to mobile computing.

3.3 The Portability Issue

It is difficult to design and implement mobile devices that are small and convenienant to use. The way mobile devices interact with their environment depends on the design and intended use. Mobile devices may be rather bulky laptop computers, PDAs, or devices as small as a wristwatch. There are several hardware issues that must be addressed in the construction of these devices.

The primary problem is that remote devices require batteries to operate. Battery power is limited, and the battery is often the largest source of size and weight of the device. Further compounding the problem is that battery technology is proceeding at a much slower rate than computer technology. It is estimated that battery power will only increase 20% in the next 10 years[3]. Thus, battery limitations are the fundamental stumbling block in creating smaller and more functional mobile units. With slow battery technology improvement, the focus is on preserving the limited available power.

Several steps can be taken to save battery power. Shrinking components and using lower voltage chips reduces usage. A lower processor clock frequency reduces power consumption with the tradeoff of lower performance. Powering down unused components like a disk drive or hard drive conserves power and has a minimal effect on the user. Finally, since transmission requires about 10 times more power than receive[2], algorithms must be designed to reduce mobile unit transmission. For example, a fixed support station can periodically broadcast information instead of having the mobile units explicitly query for it.

Other hardware portability issues include security risks, smaller user interfaces, and limited storage capacity and processing power. Security risks arise because of the mobile unit's small size which lends itself to damage or theft. If critical shared data and access privileges are stored on the mobile unit, encryption or security protocols must be in place to insure the "renegade" mobile unit does not compromise the security of the data on other mobile units or the fixed network. Smaller user interfaces challenge application designers to use different input techniques and adjust their applications to make the most out of limited screen area. Finally, the limited storage capacity and processing power necessitate sharing processing with fixed support stations or adapting applications to require less resources.

4 Ad Hoc Mobile Networking

An **ad hoc mobile network** is a collection of intercommunicating mobile hosts forming a temporary network without using as established network infrastructure. Such temporary networks have applications in remote areas or disaster zones where no infrastructure exists. Also, small temporary networks are useful in situations where an existing infrastructure is too expensive or provides weaker performance than direct connections. These situations include exchanging files between two users, connecting laptops at a conference, or even temporary networks of vehicle-mounted transmitters. The applicability and practicality of ad hoc mobile networks is debatable and will be discussed in Section 6. The rest of this section describes routing in ad hoc mobile networks. The routing algorithm described is dynamic source routing based on work done at Carnegie Mellon University in the Monarch project[5, 4].

Figure 1 shows an ad hoc mobile network with 4 mobile units. Each mobile unit has a reception range, or cell, illustrated by the dotted lines. The arrows between the mobile units represent the current network topology. In this example, the network topology is linear with connections from A to B, B to C, and C to D. Due to asymmetry in transmission and reception, a cell does not have to be circular nor does it hold that there must be a path from node i to node j given that there is a path from node j to node i. In this case, the cells are configured such that the transmission paths are symmetrical. This figure must be considered as a snap-shot in time of the current network configuration. It is highly likely that the mobile units are moving, and the network topology will change as they move in and out of each other's cells.

4.1 Dynamic Source Routing in Ad Hoc Mobile Networks

4.1.1 Conventional Routing

Conventional routing algorithms assume a relatively stable network topology and optimize their performance based on this stability. A routing protocol for an ad hoc mobile network cannot assume such stability as the mobile units are in constant motion. Thus, the routing tables at each mobile host will be frequently changing in a way not experienced in conventional networks.

There are two common methods of routing in fixed networks. They are the **link state method** and the **distance vector method**. Both require the routers to periodically broadcast routing information to other routers in order to update their routing tables. In the distance vector method, each router broadcasts to its

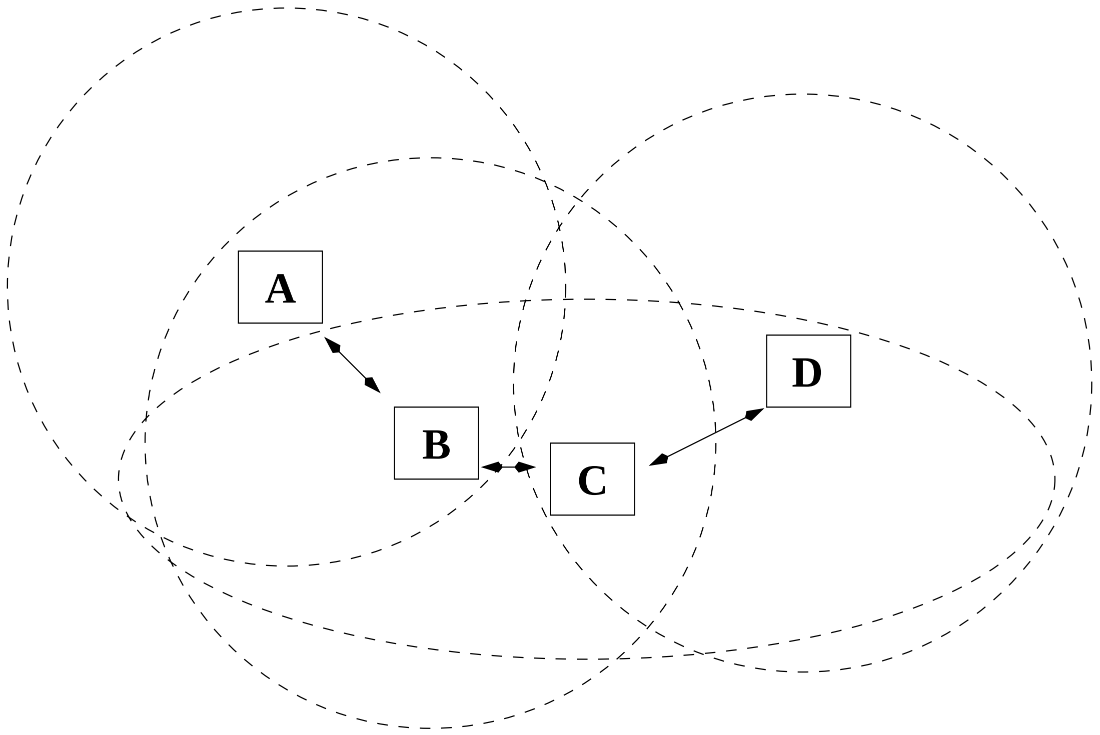

Figure 1: An ad hoc mobile network

neighboring routers its view of the distances to all hosts. Each router then computes the shortest paths based on these distances. A router in the link state method broadcasts the status of its neighboring links to all routers. A given router unions the link information to compute the shortest paths for its routing tables. In either case, routing information is broadcast and network failures or topological changes are slowly detected and propagated to all routers. These methods are unsuitable for mobile networks because broadcasting requires too much power, and they are not very efficient in detecting topology changes. Although the distance vector algorithm can be implemented for mobile networks, it is slow or even fails to detect valid paths.

4.1.2 DSR Assumptions and Motivations

An alternative routing method proposed is dynamic source routing (DSR). In source routing, the sender determines the complete path for the packet. The goal of DSR is to allow for efficient updates of routing tables given mobile unit movement without requiring periodic router broadcasts. It handles transmission path asymmetry by assuming that symmetry does not exist and handling the forward and backward connections separately. DSR has several desirable properties including:

- No periodic router broadcasts.
- Handles transmission path asymmetry.
- Designed explicitly to handle topology changes.

There are several assumptions made for DSR:

- Full mobile unit participation.
- Small network diameter.
- Hosts may move at any time but at a moderate speed.
- Promiscuous receive is allowed.

The most fundamental assumption is that all mobile units must cooperate to make the network function. That is, each MU must be willing to forward packets for others in the network. Mobile units do not want to transmit unless necessary so this assumption implies some sort of cooperation between the units either because they belong to the same organization or some other higher-level agreement between the network participants has been reached.

A small network diameter is assumed because if the network becomes too large it is highly unlikely that a long path in the network will remain stable enough to be used. Moderate mobile unit movement is also assumed to reduce path instability. With too rapid mobile unit movement, the best routing algorithm is to flood the network with packets in the hope that the destination receives them. These assumptions are reasonable because ad hoc mobile networks will tend to be small and the cooperating participants are likely to move in a way that preserves their connections.

Finally, **promiscuous receive** is the ability for a mobile unit to receive and partially process a packet not destined for it. This ability allows the mobile unit to process packet header information which may allow it to update its routing tables.

4.1.3 The Basic DSR Algorithm

The basic DSR algorithm is as follows:

- Each MU maintains a **route cache** storing routes in the network it knows.
- If the sender has a route to the destination in its cache, it is used to transmit the packet.
- If no route is known, **route discovery** is used.
- Once a route is found, the route is put in the packet header.
- Non-destination nodes in the route forward the packet to the next node in the header.
- While using a route, the host monitors it for errors using **route maintenance**.

The DSR algorithm is similar to routing in fixed networks except the route caches replace the more stable routing tables. Routing caches store known routes which are added through route discovery or examining routes in packet headers. Route cache entries are removed after a given time of non-use in the assumption that the network topology will have changed. A route is used to send packets until route maintenance detects an error. When an error is detected, the route is removed from the cache and an alternate route must be found. The route discovery and route maintenance procedures replace the periodic routing broadcasts. Basically, a mobile unit manages a dynamic routing table (the route cache) which is continually updated based on changes in network topology mostly on an **as needed** basis.

4.1.4 DSR Route Discovery

Route discovery allows any host to dynamically discover a route to any other. The route discovery algorithm is as follows:

- A host initiates route discovery by broadcasting a route request (RR) packet identifying the target (destination) host.
- When a host receives a route request packet:
  - If RR packet is found or host's address is already in route record, discard packet.
  - If target address matches host address, return copy of route in route record in route reply packet to initiator.
  - Otherwise, append host address to route record and rebroadcast.

A route request packet consists of a target host name, a route record, the initiator's address, and a unique request id. Each host maintains a list of recently received route request packets and uses the unique request id and initiator address to detect duplicates. This prevents a RR packet that has already been seen from being retransmitted and propagated through the network. The route record is an ordered list of all the nodes that the packet has visited. By examining the route record, a node has the path that the packet traveled from the initiator to itself. When the target node receives the RR packet, it returns this path in the route record back to the initiator as it must be a valid path from source to destination. The route reply can be returned using a known route or reversing the route in the route record. An alternative is to piggyback the route record as data on a route request to the initiator. When the new RR packet gets to the original initiator, it has the route to the original target as data. Then this route can be used to send the route reply packet back to the original target. After this procedure is completed, both nodes know a route to each other which may not necessarily be the same route in reverse.

4.1.5 DSR Route Maintenance

Given that the network topology constantly changes, **route maintenance** is needed to detect errors and update route caches. The idea is to replace periodic broadcasts by actively or passively observing the status of hops in a route and propagating the changes back to the packet sender.

If the network provides hop-by-hop ACK at each hop, then a host can determine if a route works simply because the data link level will return an error when trying to transmit a packet. If the network does not support lower level ACKs, passive acknowledgement may be possible if the sender can "hear" the receiver retransmit the packet. By using promiscuous receive and hearing the retransmission, the sender knows that the link must still be working. Finally, explicit ACKs can be required from the receiver in the worst case.

When an error is detected, the host detecting the error sends a route error packet to the sender of the packet being transmitted. The route error packet contains the addresses of the two end-hosts of the link in error. When a route error packet is received by the sender (or any other mobile unit using promiscuous receive), the hop in error is removed from the route cache and routes containing the hop are updated. Route discovery may have to be initiated to find a new route. As with the route reply packet, the host detecting the error needs a route to the sender to transmit the packet. A similar solution is possible with the data of the route error packet being piggybacked on a route request packet if no route is known.

4.1.6 DSR Optimizations

There are several optimizations that make the DSR algorithm more efficient. The majority of these involve improving the usage of the route cache. The route cache can be structured as a tree representing the currently known network topology. Adding or removing hops then corresponds to adding or removing links in the tree. A second improvement is to add routes at any time even when they are not explicitly asked for by a route request. This can be done when forwarding a packet for another host or using promiscuous receive and then reading the routing information in the header. Also, a route request can be satisfied by hosts other than the target by using their route cache. If a host receives a route request for a target that it has a route to in its route cache, it can pass this route back to the initiator instead of having to propagate the route request packet on to the target.

Other optimizations include piggybacking data on route requests, using promiscuous receive to identify shorter routes, and storing negative information on hops known not to be working. Also, exponential back-off is used on route requests to prevent one host from continually requesting a route that cannot be found, probably because the network is partitioned.

4.1.7 DSR Performance

DSR has been simulated at CMU. For moderate mobile unit movement, DSR transmitted 1% more packets than optimal. That is, 1% of the packets transmitted did not make it through to the destination on a given route and had to be retransmitted. No statistics were given on delay times. It is obvious that the delays may be large if it takes a long time to satisfy a route request because either the network is partitioned or the network topology is changing too frequently. Nevertheless, this is an important statistic that should have be given in order to analyze the algorithm.

In general, simulation is important in evaluating networking algorithms because of the cost and complexity of implementing a working prototype. It is more feasible to simulate an algorithm, especially when testing boundary conditions, then to work on a real system. In the ad hoc mobile network example, simulation allowed rapid testing of different number of mobile units and varying mobile speeds, without implementation of the protocol and without requiring multiple "volunteers" to run around a test area testing the protocol.

In summary, DSR is a simple yet efficient algorithm for routing in ad hoc mobile networks. Simulation has shown that packet retransmission is a minor problem for moderate mobile unit movement, and although it is not statistically verified, delay times should be reasonable for small network sizes.

5 The Effect of Mobility on Applications

There is a general issue of supporting application requirements[3] in a mobile computing environment. Regardless if the mobile computer is interconnected to a fixed network or is a participant in an ad hoc mobile network, the applications running on the mobile computer will need information on the status of the current connection and the network. An application programming interface (API) allows the mobile networking operating system or protocols to inform applications of changes in the network characteristics that could affect their execution. Specifically, many complicated applications such as databases[1] or real-time communications[7] need network information to determine load balancing, data replication and sharing, and possible execution strategies. Since no formal API yet exists, the discussion will focus on the necessary requirements of an API[6] and some of its possible applications.

Due to bandwidth and physical limitations, applications must be able to adapt for mobility. There are three main concerns:

- Handling different network capabilities
- Sharing of resources
- System/user interface limitations

How each of these problems are handled often depends on the application requirements. In the following sections, the requirements of a mobile API and the features common to many applications are discussed. Then two sophisticated applications, a mobile database and the "Everything Application", are presented along with the challenges faced in their implementation.

5.1 A Mobile API

An application programming interface (API) is a set of protocols through which the system communicates with the applications to share information on the current system environment. A mobile API has 3 important requirements[6]:

- A way for applications and the system to talk about salient environment features. (e.g. current bandwidth available, memory usage)
- A mechanism allowing applications to track their environment. (i.e. A call-back feature (system interrupt) that the system can use to inform an application of environment changes.)
- A mechanism allowing applications to request policy changes based on environment.

There are two basic methods of supporting a mobile API:

- application-transparent adaptation
- application-aware adaptation

In application-transparent adaptation, the system is responsible for handling the change by itself. The goal is to isolate the applications from the mobility such that it appears that they are in fact running on a fixed network. An example of this is the distributed file system CODA. CODA controls all file access for the mobile units. It handles buffering and file transmission during times of connection and attempts to cache files before disconnection. Applications running above CODA only see the file system and do not know that it may be remote or inaccessible at a given time. Obviously, application-transparent adaptation is desirable in that it requires no changes to the applications, but the system cannot always determine application data usage correctly. In these cases, application-aware adaptation is more practical.

Application-aware adaptation relies on some form of mobile API so that the system and applications can communicate on their requirements and handle policy changes based on environment changes. The applications explicitly tell the system the degree of the service they require (amount of bandwidth/memory, delay requirements, etc.) and expect the system to inform them when it is unable to meet those requirements. When the system tells the application that its requirements can no longer be met or that new resources are available, it can adapt its execution and possibly change its original requirements. In this method, the applications are controlling the adaptation. Clearly, they cannot receive any resources that the system cannot provide, but they can make decisions on how they what remaining resources handled to better suit their needs. A good example of the benefits of application-aware adaptation involves a video player application and a video editor application. The video player application requires real-time frame display but may be able to tolerate distortion or color-loss in times of low bandwidth. The video editor does not have a strict delay requirement but does need exact representation of each frame including full color for editing. These two applications are accessing the same video data, but their usage is very different. Without the applications informing the system of their requirements, it is highly unlikely that the system will handle the data appropriately in both cases without knowledge on how it is to be used.

5.2 A Mobile Database

Databases are sophisticated applications which manage large volumes of data. Distributed databases are yet to be commercially implemented as there remain many open problems on their implementation. A mobile database is a distributed database that must handle the high disconnection rates, the lower security, and the limited processing and storage power associated with mobile units. A distributed database implementation could be expanded to include mobile units if the disconnection problem could be handled appropriately.

Mobility causes problems in distributed databases because mobile unit disconnection isolates it from the rest of the participants. Many transaction management algorithms (locking, timestamping) require explicit communication before data access is granted. If a mobile unit is unreachable, it cannot be determined if the mobile unit is accessing the data. Thus, these sorts of algorithms will not work well in a mobile environment especially with frequent disconnection and frequent mobile unit updates of data. An alternative is to allow concurrent data access even during times of disconnection and then to resolve conflicts at reconnection. Algorithms of this type will work both in the distributed and mobile environment. Unfortunately, there is currently no general algorithm that will do this.

A mobile database will also have to perform load balancing, conserve bandwidth, and adapt its user interface. Due to the large amount of data, most processing will be done by non-mobile nodes, but the query processing algorithms should be flexible enough to balance bandwidth limitations and processing requirements. Another factor is the limited user interface. Current databases are queried using SQL. Implementing SQL may be difficult on mobile units if the input device is not a keyboard. Thus, a good database interface[1] should be graphically oriented and make good use of limited screen space. Not only is a GUI easier for the user, it better conforms to the mobile computing environment which tends to use pens and simpler interfaces over the customary keyboard/mouse combination.

With the sophistication, complex data requirements, and system-oriented decisions of a database application, it is obvious that application-transparent adaptation is not practical. A database is one of those "privileged" applications which implements many lower-level system features and thus requires cooperation with the system instead of being directed by it.

5.3 The "Everything Application"

The "Everything Application" is the Holy Grail of mobile computing. It is the grand architecture where a single mobile unit can query databases for information on all facets of life. Current work is proceeding at the University of Minnesota[7] defining such an architecture. The goal is to have mobile units able to access fixed information sources which provide information on traffic congestion, transit status, weather reports, and other news. The logistics required to design such a system are complex, yet the architecture itself is not.

Basically, the architecture consists of a centralized database which contains the relevant information. Private broadcasters can access this information and broadcast it to mobile units using the service. The mobile units are relatively simple with less processing power than a laptop computer. They store current user information and act as front end query processor. All queries are sent back to the centralized database, and the results are then returned to the mobile user. There is no data sharing or replication, and the mobile units have little responsibility. Thus, the mobile database component is relatively simple.

The implementation is complex because the data collection and query processing is difficult. Both data collection and query processing must handle large amounts of temporal and locational data. Currently, there is no economic justification to build such an elaborate data collection and processing infrastructure. Until there is economic reason, the "Everything Application" will remain an interesting but unfulfilled dream.

6 Future Work in Mobile Computing

Mobile computing is a generalized, but not radically "new", area of computing research. It may be debatable that the mobile computing environment is substantially more complex than the distributed environment, but it is obvious that it represents a superset of the problems in a distributed environment. Although many distributed algorithms may be adapted to the mobile computing environment, the problems of disconnection, limited communication, and smaller capacity inherent in the mobile environment present, at the very least, an extension to current distributed problems.

The biggest area for improvement specific to the mobile environment is in the hardware technology. Better communication, smaller devices, longer lasting batteries, and more powerful machines are needed to take mobile computing to the next level. In terms of software, routing and lower-level networking protocols can be adapted from previous technologies, but they must be extended and refined to handle and take advantage of mobile unit movement. Many distributed problems and solutions can also be ported to the mobile computing environment. Database issues and data sharing are difficult in both environments, but any solution that gracefully handles disconnection in the distributed environment should also work for mobile computers. Finally, a mobile API can also be extended from a suitable distributed API when developed and adapting applications to the mobile computing environment is an important design issue. In total, research in mobile computing is highly dependent on and interconnected with distributed systems research and adds several interesting extensions to existing distributed problems.

Ad hoc mobile networking may not represent a new research area, but the architecture may be a viable niche commercial market. There are obviously applications in remote areas and disaster zones with no existing infrastructure. Applications like exchanging files and conference internetworking are also practical even with an existing infrastructure because there is lower overhead with direct connections. On the other hand, it is unlikely that ad hoc mobile networks will be a large market because they suffer from physical and computational limitations and increasing competition from the Internet. With the convergence of telecommunications and computing, the trend is for increased network and Internet access. As these forms of access being more available and cost-effective, they will tend to be used instead of direct mobile connections. Also, since mobile communication is easier supported used fixed structures, and many organizations require a centralized database to coordinate mobile activities, the need for the ad hoc mobile network architecture is small. Thus, there will always be applications for ad hoc mobile networks, but it is unlikely that they will play a pivotal role in the "grand internetworking architecture."

Despite the ad hoc mobile network architecture's limited commercial usage, it does have a place in the research environment. With all hosts moving, the ad hoc mobile network architecture represents the most general kind of networking architecture, and algorithms designed to handle its variability may find application in more stable fixed networks. Research to handle this mobility may lead to new discoveries that would not be found by examining stable topologies, so the ad hoc mobile network has a viable place in networking research.

In summary, the mobile computing environment is a superset of the distributed environment. All the same complexities exist with a few new wrinkles. The mobile computing environment is not a radical departure either in terms of technology or research complexity, but it does represent the most general case in networking and data management, and as such is a viable area of research encompassing the most complex problems.

7 Conclusion

Mobile computing creates several problems in communication, mobility, and portability, and applications and users must adapt their behavior to this new mobile environment. When mobile units interact without a fixed infrastructure, they form an ad hoc mobile network which can be considered the most general and difficult network architecture to handle. Dynamic source routing is one solution to routing in this architecture. Finally, research in mobile computing is important not because it represents a "new frontier for exploration", but because it represents the culmination of networking; to solve the mobile computing problem, is to solve the networking problem in general.

References

[1] R. Alonso, E. M. Haber, and H. F. Korth. A database interface for mobile computers. In *Proceedings*

of IEEE GLOBECOM 92 Workshop on Networking of Personnel Communications, December 1992.

[2] George H. Forman and John Zahorjan. The challenges of mobile computing. *Computer*, 27(4):38–47, April 1994.

[3] Tomasz Imielinski and B. R. Badrinath. Wireless mobile computing: Solutions and challenges in data management. Technical Report DCS–TR–296, Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, 1993.

[4] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. Technical report, Carnegie Mellon University, 1996.

[5] David B. Johnson and David A. Maltz. Protocols for adaptive wireless and mobile networking. *IEEE Personal Communications*, Feb 1996.

[6] Brian D. Noble, Morgan Price, and M. Satyanarayanan. A programming interface for application-aware adaptation in mobile computing. Technical Report CMU-CS-95-119, Carnegie Mellon University, February 1995.

[7] S. Shekhar and Duen-Ren Liu. Genesis and advanced traveler information systems (atis): Killer applications for mobile computing? Technical report, U. of Minnesota, 1994.