Truly seamless wireless and mobile host networking

Protocols for Adaptive Wireless and Mobile Networking

DAVID B. JOHNSON AND DAVID A. MALTZ

The goal of the Mobile Networking Architecture (Monarch) Project[1] at Carnegie Mellon University is to develop networking protocols and protocol interfaces to allow truly seamless wireless and mobile host networking. The scope of our efforts includes protocol design, implementation, performance evaluation, and usage-based validation, spanning areas ranging roughly from portions of the International Standards Organization (ISO) data link layer (layer 2) through the presentation layer (layer 6). In this article, we give a status report of our current work in the Monarch Project, placing it in the context of broader efforts by the Internet mobile networking community.

Our work will enable mobile hosts to communicate with each other and with stationary or wired hosts, transparently making the most efficient use of the best network connectivity available to the mobile host at any time. To this end, the networking protocols must support adaptive operation in a number of ways. For example, host mobility means that protocols must be able to adapt packet routing to reach each mobile host in its current location. In addition, different wireless networks, intended, for example, for local-area, metropolitan-area, and wide-area use, make different tradeoffs in factors such as bandwidth, latency, error rate, and usage cost, providing different levels of network connection quality with each wireless networking product or service. Network protocols should be able to adapt in order to optimize use of the best available network connection for each mobile host at any time. Furthermore, in order to allow higher-layer protocols and applications to adapt to these changes in network connection quality, network protocols should be able to provide information to higher layers when such changes take place.

We are experimenting with our protocols in the context of the Wireless Andrew infrastructure currently being installed at Carnegie Mellon University [1]. The Wireless Andrew infrastructure builds on the current wired network infrastructure on campus that consists mostly of 10-Mb/s Ethernet equipment. For high-speed wireless access on campus, we are installing an AT&T WaveLAN network covering most of the campus buildings [2]. WaveLAN uses direct-sequence spread spectrum radio in the 900 MHz ISM band to provide a raw data rate of 2 Mb/s. For wireless access off-campus or otherwise out of range of the WaveLAN network, we are using Cellular Digital Packet Data (CDPD) [3]. The CDPD service uses idle voice channels on the existing Advanced Mobile Phone Service (AMPS) cellular telephone network to transmit data packets at a raw data rate of 19.2 kb/s.

In the next section of this article, we describe our work in routing packets to mobile hosts in a large internetwork, such as the Internet, and give an overview of our implementation work in this area. Next, we discuss the problem of routing in an ad hoc network of wireless mobile hosts, as might be needed in an area without established wireless networking infrastructure; we describe a new protocol we have developed for routing in such a network and summarize the results from a simulation of the protocol. We then describe our recent work in providing support for adaptive operation of higher-layer protocols and applications; we have developed an inexpensive protocol and application programming interface (API) for notifying higher layers when the quality of a mobile host's network connection changes as it moves between different locations, possibly including changes in the type of network in use at each location. Finally, we compare our work to related mobile networking research elsewhere and present conclusions.

Mobile Internetwork Routing

Existing internetworking protocols, including the Internet Protocol (IP), NetWare Internetwork Packet Exchange (IPX), ISO Connectionless-mode Network Protocol (CLNP), and AppleTalk, do not support host mobility. In order to aggregate the routing information and routing decisions at each level of the internetwork topology, internetworking protocols use *hierarchical* addressing and routing schemes. For example, in the Internet, IP addresses are divided into a separate *network number* and *host number*; routers throughout the Internet need be concerned only with routing a packet to the correct network; once there, it becomes the responsibility of that network to route the packet to the correct individual host. This routing aggregation becomes increasingly important

[1] *The Monarch Project is named in reference to the migratory behavior of th emonarch butterfly. Each autumn, millions of monarch butterflies migrate from central and eastern United States and Canada to overwintering roosts in central Mexico; with the coming of spring, the monarch population again migrates northward. The name "Monarch" can also be considered as an acronym for Mobile Networking Architecture."*

as the size of the internetwork grows. The Internet, in particular, currently consists of over 6 million individual hosts, and this number has been doubling approximately every year. Indeed, new levels of hierarchy have been added to the Internet addressing scheme with subnetting [4] and Classless Inter-Domain Routing (CIDR) [5], and additional support for further hierarchy is planned in IPv6, the new version of IP currently being designed for the Internet [6].

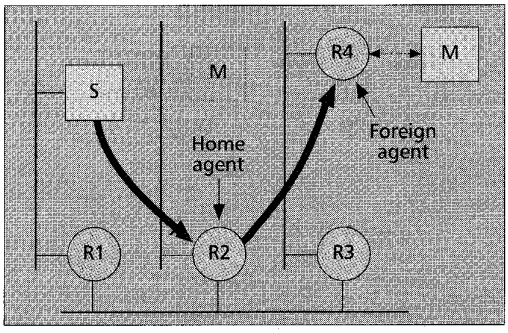


Figure 1. *Basic architecture of the IETF Mobile IP protocol.*

It is this hierarchy, however, that defeats host mobility. With hierarchical addressing and routing, packets sent to a mobile host can only be routed to the mobile host's home network regardless of the host's current location, possibly away from home. A mobile host could perhaps change its address as it moves from one network to another, but such changes can be difficult and error-prone; changing addresses involves modifications to a number of configuration files on the host and on network servers, and often requires that all existing transport-level network connections be restarted or the host rebooted. In addition, a mechanism would be needed to inform other hosts of the mobile host's new address, further complicating the change to a new address. Instead, a solution is needed for correctly routing packets to any mobile host in its current location given the host's (constant) home address.

The IETF Mobile IP Protocol

The Internet Engineering Task Force (IETF) is the principal protocol standards development body for the Internet. Over the past few years, the IETF Mobile IP Working Group has been working to develop a standard for routing IP packets to mobile hosts in the Internet, and we have contributed a number of protocol designs to this effort [7–10]. Working within the IETF provides a direct avenue for transferring the results of our research into the Internet community. In this section, we provide an overview of the basic IETF Mobile IP standard which is currently nearing completion [11].

Figure 1 illustrates the basic architecture of the protocol. In Fig. 1, R1, R2, and R3 are routers, each connecting an IP subnet to a simplified Internet backbone. M is a mobile host whose home network is the network connected by R2 but which is currently connected to a wireless network through router R4. Each mobile host must have a home agent on its home network, which forwards IP packets to the mobile host while it is away from home. Here, router R2 is serving as the *home agent* for mobile host M, although any host or router on this home network could serve that role. When visiting any network away from home, each mobile host must also have a *care-of address*. Normally, the care-of address is the address of a *foreign agent* within the local foreign subnet, which has agreed to provide service for the mobile host; the foreign agent delivers packets forwarded for the mobile host to it on the local network. Here, R4 is serving as the foreign agent for M. Optionally, if a mobile host can acquire a temporary IP address within the local subnet, such as through the Dynamic Host Configuration Protocol (DHCP) [12], it may instead use this temporary address as its care-of address; packets tunneled to the mobile host are tunneled to this temporary address, while the mobile host continues to use its home address for all other functions. In this case, the mobile host in effect operates as its own foreign agent with this temporary address.

To find a foreign agent with which to register, an *agent discovery* protocol is used. Agent discovery also provides a means for a mobile host to detect when it has moved within range of a different wireless network. It can detect when it has moved to a new foreign network when it receives an advertisement from a new foreign agent, and when it has returned to its home network when it receives an advertisement from its home agent. The agent discovery protocol operates as a compatible extension of the existing Internet Control Message Protocol (ICMP) *router discovery* protocol [13].

When moving to a new location, a mobile host must register with its home agent so that the home agent always knows the mobile host's current care-of address. When using the address of a foreign agent as its care-of address, the registration takes place through that foreign agent so the foreign agent can agree to provide service to the mobile host and knows that the mobile host is using this care-of address. The association between a mobile host's home address and its care-of address is called a *mobility binding*, or simply a *binding*. Each binding has associated with it a *lifetime* period, negotiated during the mobile host's registration, after which the registration is deleted; the mobile host must reregister within this period in order to continue service with this care-of address.

When sending a packet to a mobile host, a sending host (called *correspondent host*) simply addresses and sends the packet in the same way as any other IP packet. The packet will thus be routed through the Internet to the mobile host's home network. The correspondent host need not understand the Mobile IP protocol or know that the destination host is mobile. While a mobile host is registered with a care-of address away from home, the mobile host's home agent must intercept any packets on its home network addressed to the mobile host. For each such packet intercepted, the home agent *encapsulates* the packet and *tunnels* it to the mobile

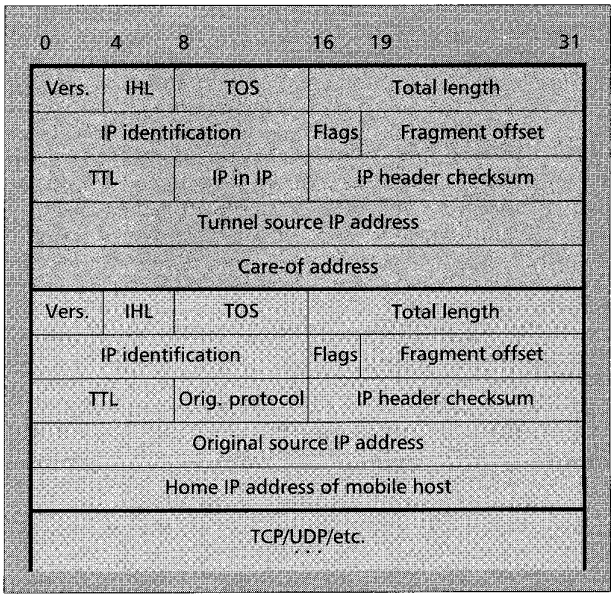
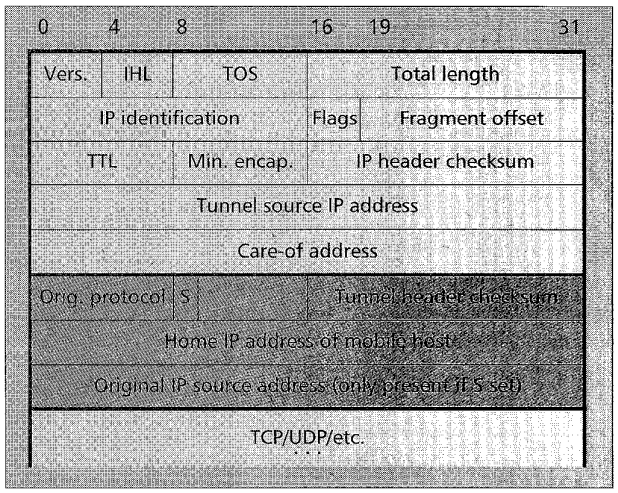


Figure 2. *Mobile IP tunneling using "IP in IP" encapsulation.*



■ **Figure 3.** *Mobile IP tunneling using "minimal" encapsulation.*

host's care-of address.

The default encapsulation protocol, known as "IP in IP" encapsulation, is illustrated in Fig. 2. With this protocol, a new IP header (shaded) is wrapped around the existing packet. The source address in the new IP header is set to the address of the node tunneling the packet (the home agent), and the destination address is set to the mobile host's care-of address. The protocol number, such as Transmission Control Protocol (TCP) or User Datagram Protocol (UDP), in the new IP header is set to the protocol number for "IP in IP" encapsulation. Once encapsulated, the packet is routed through the Internet in the same way as any IP packet addressed to the foreign agent, and only the home agent and foreign agent need know that tunneling is taking place. When the packet arrives at the foreign agent, the packet is processed by the encapsulation protocol at the foreign agent, as indicated by the protocol number in the IP header. The foreign agent removes the added header and transmits the packet to the mobile host over the local network interface on which the mobile host is registered.

The "IP in IP" encapsulation protocol adds 20 bytes (the size of an IP header) to each packet tunneled to a mobile host away from home. An alternative tunneling protocol, known as "minimal" encapsulation, is also defined within the basic Mobile IP protocol, and adds only 8 or 12 bytes to each packet. This protocol is illustrated in Fig. 3. With this protocol, a small tunneling header (shaded) is inserted in the packet *after* the existing IP header, before any existing transport level header such as TCP or UDP. The destination address in the IP header is copied into the tunneling header and is replaced in the IP header by the mobile host's care-of address. Similarly, the protocol number in the IP header is copied into the tunneling header and is replaced in the IP header by the protocol number indicating minimal encapsulation. Finally, if the original sender of the packet is not the node tunneling the packet (the home agent), the source address in the IP header is copied into the tunneling header and is replaced in the IP header by the tunneling node's address, and a bit is set in the tunneling header to indicate that the copied source address is present. When the packet arrives at the foreign agent, the original IP header is reconstructed, the tunneling header is removed, and the packet is transmitted locally to the mobile host. Although more efficient than "IP in IP" encapsulation, the minimal encapsulation protocol cannot be used with IP packets that have been fragmented [14], because the tunneling header does not provide a means to indicate that the original packet was a fragment.

All registrations of a mobile host with its home agent must be authenticated in order to guard against malicious forged registrations. Without authentication, an attacker could register a false care-of address for a mobile host, causing its home agent to arbitrarily redirect future packets destined to the mobile host. Registration authentication must verify that the registration request legitimately originated with the mobile host, that the request has not been altered in transit to the home agent, and that an old registration request is not being replayed (perhaps long after the mobile host was at that care-of address).

The protocol currently uses an extensible authentication mechanism, with the default currently based on the MD5 *secure one-way* hash function [15]. A "keyed MD5" algorithm is used, based on a secret key shared between a mobile host and its home agent, such that the authentication value can only be correctly computed by a node knowing the secret key. Administration of the shared secret key should be fairly simple, since both the mobile host and its home agent are owned by the same organization (both are assigned IP addresses in the home network owned by that organization). Manual configuration of the shared key may be performed, for example, any time the mobile host is at home, while other administration of these hosts is being performed. Replay protection currently may use either *nonces* or *timestamps*.

Route Optimization Extensions

In the basic IETF Mobile IP protocol, while a mobile host is away from its home network, *all* packets for the mobile host must follow the path shown in Fig. 1. Each packet is routed through the Internet to the mobile host's home network and must then be tunneled by the mobile host's home agent to the mobile host's current location. This indirect routing through the home agent in general causes unnecessary overhead on the home network and on the portion of the Internet leading to and from the home network, and causes unnecessary latency in the delivery of each packet to the mobile host.

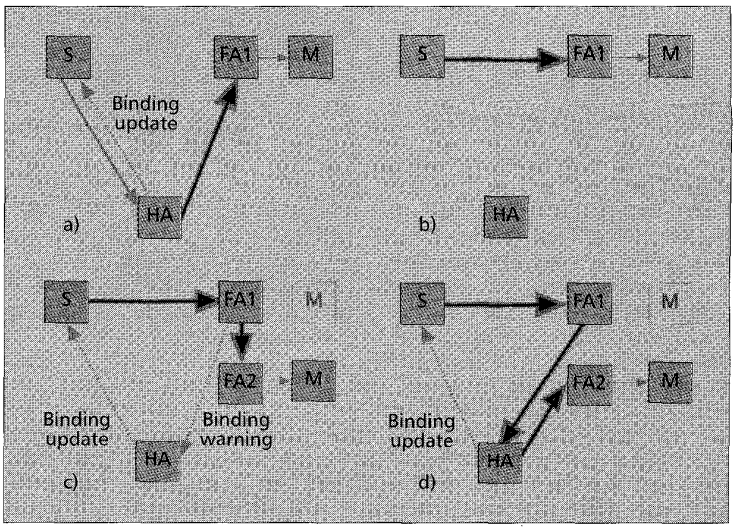
We have developed a compatible set of extensions to the basic IETF Mobile IP protocol to address this problem, and these extensions are now being standardized alongside the basic Mobile IP protocol within the IETF [16]. These extensions, known as "Route Optimization," allow other hosts or routers sending packets to a mobile host to dynamically learn and cache the mobile host's current location; the sending node can then tunnel its own packets directly to the mobile host, bypassing the trip to and from the home agent. This capability has been present in all of our designs submitted to the Mobile IP Working Group [7–10], and we view it as essential for the efficiency and scalability of the protocol.

In the Route Optimization extensions, when a mobile host's home agent intercepts and tunnels a packet to a mobile host away from home, the home agent also returns a *binding update* message to the original sender of the packet (the correspondent host), as shown in Fig. 4a. This allows the sender to cache the current binding of the mobile host and to use the care-of address in the binding in tunneling its own packets to the mobile host in the future, as shown in Fig. 4b. One challenge that must be addressed in the design of this mechanism, though, is that of *cache consistency*; when a mobile host moves to a new location, all cached copies of its binding at correspondent hosts become out of date.

With Route Optimization, when a mobile host moves from one foreign agent to another, it may notify its previous foreign agent of its new care-of address by sending it a *binding update* message. This allows the previous foreign agent to cache the new binding of the mobile host, forming a "forwarding pointer" to its new location. If a correspondent host later tunnels a

packet for the mobile host using an out-of-date cache entry, the previous foreign agent will receive the packet and will re-tunnel it to the new location. The previous foreign agent also sends a *binding warning* message to the mobile host's home agent to request it to send a binding update message to the correspondent host. For example, Fig. 4c shows the operation of the protocol after mobile host M has moved from foreign agent FA1 to foreign agent FA2.

If, instead, the cache entry at the previous foreign agent no longer exists by this time (e.g., because that entry in the cache was replaced with an entry for a different mobile host), the foreign agent instead forwards the packet to the mobile host's home agent by tunneling the packet to the mobile host's own address, as shown in Fig. 4d. The packet will thus reach the home agent in the same way as any other packet addressed to the mobile host; the home agent will also be able to determine from the tunnel encapsulation header that it was tunneled from this foreign agent, allowing recovery in the case in which the home agent believes that this is the current foreign agent serving the mobile host, but perhaps the foreign agent has crashed and lost its knowledge of the mobile host's registration.



■ **Figure 4.** *Operation of the route optimization protocol extensions: a) sending the first packet to a mobile host; b) sending subsequent packets to a mobile host; c) sending the first packet after a mobile host moves; d) tunneling the packet in case the cache entry has been dropped.*

Cache consistency is thus addressed in both cases by dynamically updating any out-of-date cache entry when it is next used. A packet routed based on an out-of-date cache entry will be routed indirectly to the mobile host's new location, and the cache entry will be updated as a side effect.

A further challenge that must be addressed in the design of Route Optimization is that of *authentication*. Unlike the basic IETF Mobile IP protocol, Route Optimization may, in general, require the ability to authenticate a binding update message to any node in the Internet. In the basic Mobile IP protocol, all control over routing packets to a mobile host rests with the mobile host's home agent, which intercepts and tunnels all packets to the mobile host. Authentication of registration messages with the home agent in this way is reasonably easy, since the home agent and the mobile host can share a secret key. However, with Route Optimization, any correspondent host that is to cache a mobile host's binding must be able to authenticate the binding update message in which it learns the mobile host's binding, in order to guard against attacks involving forged binding updates. Authentication in this case is much more difficult, since the correspondent host may belong to a different organization than the mobile host and its home agent, and there is currently no generalized authentication or key management mechanism for the Internet; patent restrictions and export controls on the necessary cryptographic algorithms have slowed development and deployment of such facilities in the Internet.

In the Route Optimization extensions, we are currently using the same style of authentication for binding update messages as is used for registration in the basic IETF Mobile IP protocol. In order for the home agent to send a binding update to a correspondent host, it must share a secret key with the correspondent. Until a key distribution mechanism is defined for the Internet, these keys will be manually configured, and if no shared key exists, the Route Optimization extensions cannot be used with this correspondent. The correspondent host can still communicate with the mobile host using the basic IETF Mobile IP protocol.

We have defined the protocol to minimize the number of pairwise shared secret keys required for operation. By establishing a shared secret key with some home agent, a correspondent host is able to receive authenticated binding updates (and thus to maintain cached bindings) for all mobile hosts served by this home agent. This relationship is fairly natural, since the mobile hosts served by any particular home agent, in general, all belong to a single organization (which also owns the home agent and the home network). If the user of a host often collaborates with any number of people from this organization, manually establishing the shared secret key with this home agent may be worthwhile.

Implementation Status

We have completed an implementation of the mobile internetwork routing protocol under the NetBSD version of the UNIX operating system. This implementation contains all features of the basic IETF Mobile IP protocol, and we are currently completing additions to the implementation for Route Optimization and network connection quality notifications for supporting adaptive higher-layer protocols and applications (described later in this article). Our implementation includes all functions of a mobile host, correspondent host, home agent, and foreign agent, and allows dynamic, transparent switching between the Ethernet, WaveLAN, and CDPD networks of the Wireless Andrew infrastructure. Since NetBSD is based on the 4.4BSD Lite UNIX source, we believe our implementation should be able to be ported easily to other versions of UNIX derived from one of the Berkeley source distributions, but we have not yet attempted this. We intend to make the source for our implementation freely available once it is completed.

The implementation is divided between a portion in the kernel and a daemon process running on the host. In general, operations that must be performed for each packet, such as encapsulation and decapsulation, are performed in the kernel, whereas higher-level functions and policy decisions are performed within the daemon. For example, the exchange of packets necessary for registration and the management of registration lifetimes is the responsibility of the daemon, which sends messages on a `PF_ROUTE` routing socket to the kernel to manipulate the kernel's routing tables. This structure is similar to the implementation of existing routing daemons for UNIX, such as `routed` and `gated` [17].

Routing in Ad Hoc Wireless Networks

At times, no infrastructure such as the Internet may be available for use by a group of wireless mobile hosts, or the use of an available network infrastructure may be undesirable due to reasons such as cost or convenience. Examples of such situations include disaster recovery personnel or military troops in cases in which the normal infrastructure is either unavailable (e.g., in a remote area) or has been destroyed (e.g., after an earthquake); other examples include business associates wishing to share files in an airport terminal, or a class of students needing to interact during a lecture. If each mobile host wishing to communicate is equipped with a wireless local area network (LAN) interface, the group of mobile hosts may form an *ad hoc network*. An ad hoc network is a temporary network, operating without the aid of any established infrastructure or centralized administration.

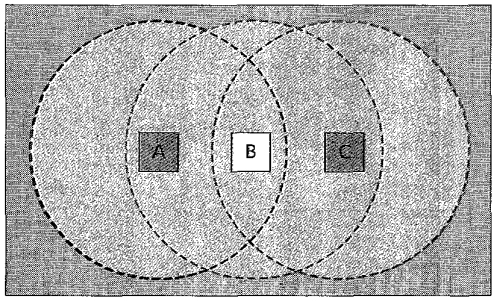


Figure 5. *A simple ad hoc network of three wireless mobile hosts.*

In an ad hoc network, some hosts wishing to communicate may be outside of wireless transmission range of each other, but may be able to communicate if other hosts in the network are willing to forward packets for them. For example, Fig. 5 depicts a simple ad hoc network of three mobile hosts, in which the transmission range of each host's wireless interface is indicated by a circle around the host. Mobile host A cannot directly send a packet that will reach C, since C is outside A's wireless transmitter range. However, mobile host A can send the packet to B if B is willing to forward the packet to C by retransmitting it.

An ad hoc network in general requires some form of routing protocol in order to dynamically find multihop paths through the network and in order to adapt to new routes as the mobile hosts in the network move. Furthermore, the protocol must be able to operate correctly in spite of the varying propagation characteristics of each mobile host's wireless transmissions, for example, due to changes in sources of interference in the vicinity of each mobile host.

Conventional Routing Protocols

Conventional routing protocols for wired networks use either *distance vector* or *link state* algorithms, and the basic distance vector algorithm has also been used successfully in some wireless ad hoc networks [18–20]. In distance vector routing, each router broadcasts to each of its neighbor routers its view of the distance to all hosts, and each router computes the shortest path to each host based on the information advertised by each of its neighbors. For use in ad hoc networking, each mobile host is treated as a router and periodically broadcasts a routing update packet to any neighbor mobile hosts within its transmission range.

However, in an ad hoc network, network bandwidth, battery power, and available central processing unit (CPU) processing time on each host are likely to be limited resources. With distance vector routing, a mobile host must continue to send periodic routing updates, occupying network bandwidth and consuming battery power on the host for the transmissions. Furthermore, each of its neighbor mobile hosts must continue to receive these updates, and thus cannot easily conserve its own battery power by putting itself into "sleep" or "standby" mode when not busy with other tasks. In addition, many of the "links" between routers seen by the routing algorithm may be redundant, since all communication is by broadcast transmissions. These redundant links unnecessarily increase the CPU overhead required to process routing updates and compute new routes.

Finally, conventional routing protocols are not designed for the type of dynamic environment that may be present in ad hoc networks. In conventional networks, links between routers occasionally go down or come up, and sometimes the cost of a link may change due to congestion, but routers do not generally move around dynamically, as may happen in an ad hoc network. Distance vector algorithms, in particular, converge slowly to new stable routes after changes in topology, and may create temporary routing loops and "black holes." Furthermore, in some environments and host configurations, distance vector protocols may compute some routes that do not work, since wireless transmissions between two hosts may not necessarily work equally well in both directions, due to differing propagation or interference patterns around the two hosts. Depending on the wireless network medium access control (MAC) protocol in use, even though a host, such as A in Fig. 5, may receive a routing update from another mobile host, such as B, packets that A might then transmit to B for forwarding may not be able to reach it.

A Dynamic Source Routing Protocol

We have designed a new routing protocol for ad hoc networks based on a different type of routing. Rather than using either distance vector or link state routing, our new protocol uses *dynamic source routing* of packets between hosts in the ad hoc network [21, 22]. In source routing, the sender of a packet determines the complete sequence of nodes through which to forward the packet, and lists this route in the packet's header; when received by each node along this path, the packet is simply retransmitted to the next "hop" indicated in the path. Source routing has been used in a number of contexts for routing in wired networks, using either statically defined or dynamically constructed source routes, and has been used with statically configured routes for routing in a wireless network [23].

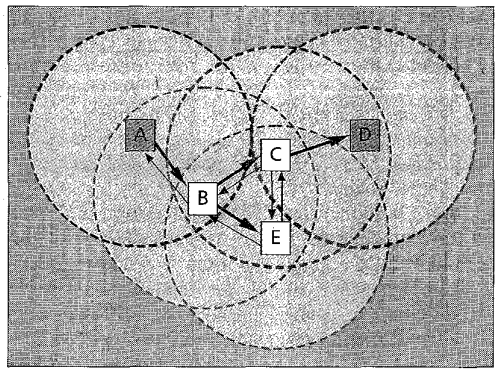


Figure 6. *Operation of the route discovery protocol.*

In our dynamic source routing protocol, there are *no* periodic routing messages of any kind. Each mobile host participating in the ad hoc network maintains a *route cache* in which it caches source routes it has learned. When one host sends a packet to another host, the sender first checks its route cache for a source route to the destination. If a route is found, the sender uses this route to transmit the packet. If no route is found, the sender may attempt to discover

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.