Service-Based Computing on Manets:

Enabling Dynamic Interoperability of First Responders

Joseph Kopena, Evan Sultanik, Gaurav Naik, Iris Howley, Maxim Peysakhov, Vincent A. Cicirello, Moshe Kam, and William Regli, *Drexel University*

Mobile ad hoc networks will form a critical part of the first-responder communications infrastructure. Empirical data shows how network-aware, autonomous, mobile agents can manage information services on live manet environments.

A multidisciplinary team in Drexel University's College of Engineering has been working with local law enforcement and transportation officials to identify problems in enabling police, fire, security, and other public protectors to effectively communicate and collaborate in first-response situations. Development of the Philadelphia Area Urban Wireless Network Testbed (PA-UWNT) is part of this effort.[1–3] The PA-UWNT is a mobile ad hoc network (manet) comprising mobile computers (PDAs, tablets, and laptops) and Web Service-based applications (situation awareness, sensors, tracking, data feeds, and so on). Such systems offer a solution for communications and situation management in infrastructure-free environments where power, network, and other computing-related infrastructure are likely nonexistent or inoperable (for example, coordinated police presence at large public events, medical personnel at an accident scene, and emergency responders to a natural disaster, as well as homeland security and military scenarios).

Our experience in the PA-UWNT project indicates that constructing such systems will require new research developments in computer networking, agent and service-based computing, and security that integrate each of these disciplines at several fundamental levels. At the application level, properties of mobile computing and the urban setting make mobile-agent software and service-based computing particularly attractive. The complex, dynamic, and disconnected nature of mobile networks quickly overwhelms centralized controls and data distribution. Sophisticated reasoning might be required to maximize resource use, even during network transit. Flexible matching, choreography, and composition might be required to use the possibly heterogeneous mix of available resources. Here, we outline part of this approach through experiments demonstrating the utility of integrating the network and agent layers and enabling agents to reason about the current operating context. We posit that autonomous agents that can reason about the network's state and services offer an effective means of meeting the manet environment's challenges.

Service-based computing for first responders

We've used the PA-UWNT to conduct live experiments in urban settings and evaluate the real-world efficacy of service-based computing systems in operationally relevant environments. Working with university and local government officials, the team has performed system metrology in areas of Philadelphia that present special challenges for first-responder coordination. These urban regions include widely varied terrain with a significant effect on wireless networking and system communications: subterranean platforms, corridors, and tunnels, consisting of many metal columns and dense interior architecture; large and small buildings that provide communication signal problems due to multipath, reflections, and interference; and buildings and trees combined over an extended area, creating special radio frequency and modulation needs. Effective systems in these environments must be able to operate

1541-1672/05/$20.00 © 2005 IEEE
Published by the IEEE Computer Society

Figure 1. Varied urban terrain poses challenges to the design of a communication and coordination system for first-responder teams. The large views show areas of interest including sites on or near the Drexel University campus, underground subway corridors, and Center City locations.

in and adapt to changing conditions as users move through the city or are redeployed. Figure 1 shows an aerial map of this challenging urban environment.

Benefits

Agent-based, service-oriented computing on a platform such as the PA-UWNT can benefit first-responder teams in many areas.

Interoperability. In first-response scenarios, dynamic teams come together, requiring an effective means of communication and coordination. These dynamic teams likely are made up of a heterogeneous mix of communications equipment and other resources. For example, in a multijurisdictional situation, officers from multiple police departments might need to coordinate and interact with each other as well as fire department personnel and other public protectors. A service-based-computing paradigm can provide an effective means for seamless interoperation for these dynamic teams.

Situation awareness. Dynamic real-time access to sensors and tracking information can enhance responders' *situation awareness*. For example, GPS-equipped mobile devices can let public protectors locate their teammates. Also, video feeds from cameras often located throughout office buildings can give tactical teams a better view of a situation before moving in on a suspect.

Connecting to back-end databases. Different police and fire departments use different computer-aided dispatch systems, personnel databases, and public databases (vehicle registrations, state systems, and so on). Up-to-date information can be critical to dynamic decision making. The ability to directly access dispatch systems and other live data sources accelerates critical information flow to personnel on the scene. This information flow can be two-way as well, providing incident commanders with a bird's-eye view of the situation.

A scenario

Consider a hostage situation in a building. Initially, responders know very little about the hostage takers (for example, how many there are, how heavily armed they are, and who they are) or about the hostages (for example, how many there are, how old they are, and whether they're injured). If the situation is in a major metropolitan area, a nearby police unit might arrive at the scene and access video feeds from security cameras in the building. We can wrap these video feeds as services provided wirelessly to the tactical team the moment the teams arrive at the scene.

Further into the scenario, the first responders have set up a command post at ground level. A second police unit arrives from another nearby jurisdiction to offer assistance. Simultaneously, a helicopter drops a small tactical team on the building's roof. Ideally, these arriving teams can interact seamlessly with the first team. For example, the information the first team gathered can provide a shared situational awareness for the composite team. A service-based architecture can enable multijurisdictional interoperability by providing users the ability to publish-subscribe to services on other networks through service registries and common service description languages and ontologies. The helicopter that dropped the team on the roof can provide a video feed of balconies and windows, providing a service that the first team couldn't access previously. The rooftop team has worked its way down a stairwell. One member of this team glimpsed one of the suspects, transmitting an image down to the ground floor command center. Via a satellite feed (another service) in the truck outside, this image is matched with high probability to a known terrorist in an FBI database (another service). The service forwards this information back to the ground floor and distributes it to the three teams wirelessly, further enhancing their understanding of the seriousness and danger of the situation.

Challenges

A manet is an infrastructureless wireless network in which each host can act as a router for network traffic, allowing message traffic meant for one host to pass through others on a potentially multihop route to its destination. The mobility of hosts in a manet can lead to hosts leaving communications range and frequent network topology changes (see figure 2). The manet environment's dynamics pose several challenges to distributed computing to meet the communication and collaboration needs in first-response scenarios. Such challenges can include mobility and service dynamics.

Mobility

Instead of being stationary and dependent upon a fixed infrastructure, the hosts within a manet can change their physical topology and rely on each other to maintain communication with the rest of the network. Manets cause sig-

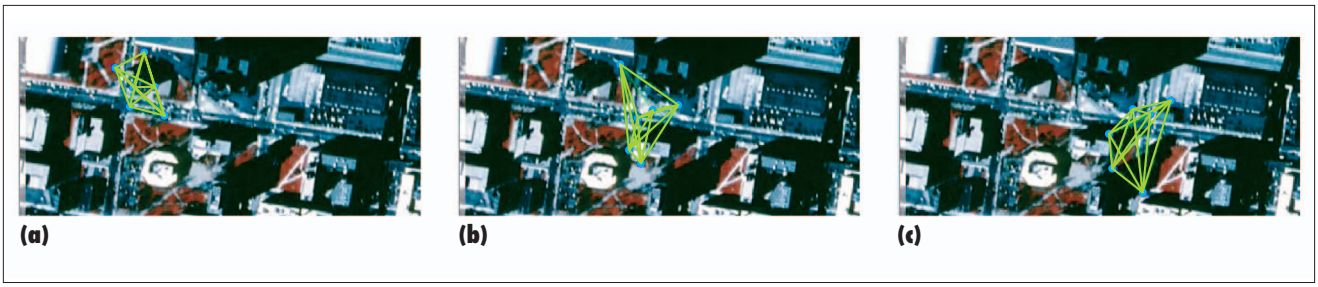


Figure 2. Three field-test snapshots showing examples of dynamic network topology as a team moves through the urban environment on Drexel University's campus. The blue dots indicate the GPS locations of six hosts of the PA-UWNT. The green arcs indicate pairs of hosts that are directly connected. As the users of the hosts move on foot through the campus, the PA-UWNT's topology changes. This network state data is also available live to PA-UWNT users, collected and delivered to the users by mobile agents.

nificant problems with a service registry's implementation in the network. Hosts often use registries as a central directory to access information about all available services, but the unpredictability of the network topology in a manet prevents many hosts from having access to the registry. A host can be connected to the registry one moment and become disconnected the next, which means the registry can no longer host useful service advertisements that querying agents might use to locate a given service. While the registry caters to one portion of the network, some hosts might never connect to the registry. This leaves their services undiscovered and might even prevent interaction between providers and requesters that are directly connected.

Service dynamics

Because sharing a common registry among all hosts is impractical in a manet, consider a case in which each host maintains its own service registry that functions as a traditional Semantic Web registry. This would facilitate communications among service requesters and service providers either directly connected or connected across multihop network routes. The challenge here: How do we deal with services that disconnect from the network? This question is especially important when services leave the network suddenly, without warning (for example, when a user of the mobile computing node that contains the service moves out of communications range).

A host can start or stop instances of a service to accommodate the changing needs of a dynamic network—Maxim Peysakhov and William Regli's work on service management, for example.[4] Other examples include a loss of power or device malfunctions. In such cases, the host might not have the foresight to inform the local service registries. Therefore, the service registry in this environment must be able to handle service providers' sudden disappearance and their frequent reregistration and deregistration.

Technical approach

Our team has been addressing three key problems:

- service registration,
- service discovery, and
- service choreography.

In terms of service registration, we must first overcome significant challenges related to the manet environment that typical Semantic Web systems don't face. A single service registry is insufficient. In a manet, it's quite common for the network topology to become disconnected as users move in an environment that's characterized by obstacles to wireless communications. As we mentioned, any approach to service registration must also contend with services frequently disappearing and reappearing as their hosts move out of communications range, shut off a device, or, in some environments, become destroyed (for example, by an adversary). We'll describe our suggested registry architecture in the next section.

Disappearing hosts in manet environments raise several issues regarding service discovery. How do the users, hosts, software components, and other services know when a host, or a service on a given host, has become unavailable or whether it will return? Likewise, new hosts can join the network as users move into communications range. How can users and applications on one network use services that might be available to them on another if these manets have the ability to merge? We employ a population of random-walk service discovery agents whose task is to report service locations and capabilities to the manet's local service registries.

The manet environment's dynamics also pose challenges for service choreography, introducing uncertainty and quality-of-service issues. If an agent or user is interacting with a service when that service unexpectedly disappears, how should the agent respond? Given the manet's dynamism, it's possible that this host along with the desired service can reappear. The agent can wait. But, in some circumstances, the service might not return, so the agent might need an alternative service.

An OWL-S Web Services registry for manets

The Semantic Web aims to produce machine-interpretable Web pages, so that agents and other software can use the information on the Internet more efficiently than parsing the human-interpretable languages currently in use.[5] The *Web Ontology Language* (OWL) is the markup language for defining ontologies for the Semantic Web, and *OWL-S* (www.daml.org/services/owl-s/1.1) is a group of ontologies designed specifically for Semantic Web Services. The OWL-S service comprises three major parts: a profile, a process model, and a grounding. The profile summarizes the inputs, outputs, preconditions, and effects that a service can possess. The process model describes the details of those parameters. The service grounding defines how to piece the parameters together to affect the desired results. When combined, the OWL-S ontologies give hosts and agents a way to interact autonomously with the services they require.

The OWL-S ontology was designed to allow significantly automated service discovery and composition. One of our goals in designing our service registry was to keep this functionality, while making it more appropriate for a manet environment. For this proof-of-concept model, we've designed and

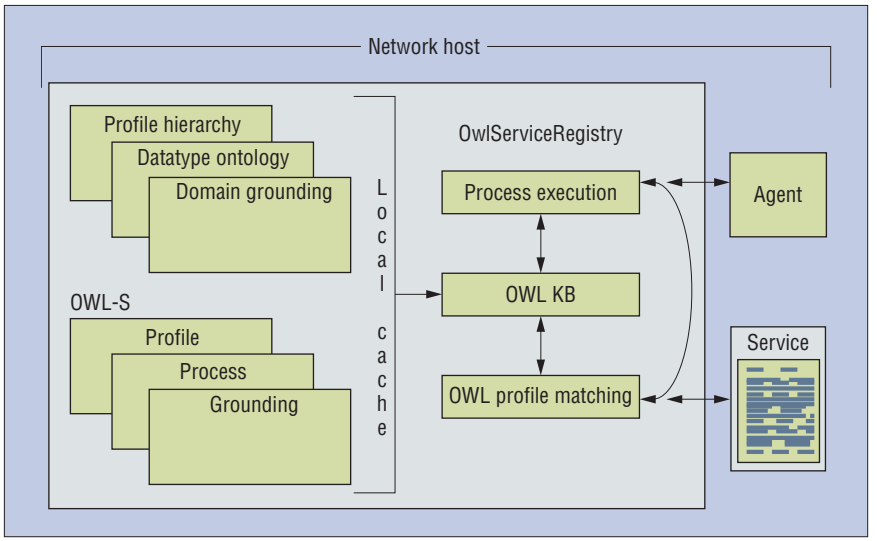


Figure 3. The local OwlServiceRegistry architecture.

used a grounding that uses Java methods as the main form of communication. The Java grounding ensures that each service has a method name and an optional parameter name and index. With this information, services can receive input, and agents can retrieve their output, thereby fitting smoothly into the PA-UWNT. We can easily replace these services with a *Web Service Definition Language* grounding for other systems. A manet's hosts can get disconnected from the Internet, rendering them unable to retrieve the ontologies from any other common source. Therefore, we've placed copies of all of OWL-S's components on each individual host, so they can be loaded from a local directory. Any other knowledge that must be universal, such as a datatype definition or another OWL class, must also be retrieved from local cache.

Figure 3 shows our OwlServiceRegistry architecture. For our network, we placed a service registry on each host. Aided by OWL-S and other ontologies, the registry handles service advertisements, requests, and a mapping between service names and descriptions. To invoke a service, a host uses the execution module to perform the Java grounding of the desired service.

Registration. When a service arrives at a host that has an available registry, the service gives the registry its OWL-S description. The host then loads the description into the registry's knowledge base, retrieves new service names, and maps the names to the service object and OWL-S description so that it can find the service at a later time.

Querying. An autonomous agent knows what's required to complete a particular task step. Whether it's to retrieve or give a certain type of information, the agent's actions will be facilitated by an OWL-S profile-matching mechanism. Agents make queries based on inputs, outputs, and the profile hierarchy. If the agent is sensitive to the system's state, it can look for services with particular preconditions or, if need be, certain postconditions. A query can also involve a service's effects. Essentially, an agent could base its query on any detail outlined in the service profile. When an agent submits a query, the registry returns best matches as service names. Using the service registry, this service literally gives the agent access to the corresponding object and OWL-S description.

Performing the process model. The agent uses the service symbol from the profile query to obtain the service object and the service's OWL-S description. It then gives these two items to the execution module, along with any input values the agent would like to pass to the service. The execution module performs the process model by retrieving the Java method name and correctly ordering the inputs. At this point, the module is no longer needed, and the agent can move to the next step in its task.

Failure. Because of the network's uncertainty, failure is a frequent issue—one that we pass on to the agent. If a query returns an empty match set, the agent decides what to do next: query with looser terms, wait and then query again, or migrate to another host. If the query returns multiple matches, the agent must specify its preferences. If failure occurs while executing the process model, the agent again has similar choices. The agent can pass its preferences for handling failure to the execution module to minimize interaction between the two components.

Service discovery on manets using mobile agents

To enable agents to reason about availability of alternative services in a manet environment, we must first develop an effective approach to service discovery in this challenging environment. In earlier work, we began to consider how software agents can achieve global state awareness in peer-to-peer networks such as manets[6]—for example, agents that can gather information about the location and capabilities of services on such dynamic networks. These services can include Web Services as well as services that mobile agents provide.

Our approach is based on the deployment of a set of service monitoring agents, $\mathcal{A}$, that randomly walk the set of hosts, $\mathcal{H}$, of the manet to provide an accurate, online means of service discovery. The agents act like bees working to "pollinate" the network's service registries with knowledge of locations of services that they encounter. The underlying network topology dictates each agent's walk (see figure 4). Furthermore, there's no guarantee that all agents visiting a host have encountered a service thus far on their walks.

This approach has important advantages over alternatives, such as naive message passing and broadcast. First, it uses minimal network bandwidth, and bandwidth usage scales linearly. This is an important issue for resource-constrained mobile devices and large-scale, peer-to-peer networks. Second, mobile code provides for synergy between networks of nonhomogeneous service discovery architectures. Third, services need not register themselves. Finally, properties of random walks are relatively easy to mathematically model and likewise make inferences upon. For example, in other work,[6] we've presented an adaptation of the PageRank[7] algorithm as a means of estimating the probability that an agent randomly walking a network will visit a given host. This estimate can then be used to predict service availability—for example, to estimate the probability that a service discovery agent with knowledge of a particular service will visit our host within some length of time.

The agents' task environment, a manet, is stochastic, dynamic, and continuous. A delay

exists between actual topology changes and propagating knowledge of these changes throughout the network. The agents don't have a goal, per se; their sole purpose is to randomly walk the network that's gathering information. Agents' percepts comprise only the set of services available at the current host, S_h, and the set of hosts neighboring the current host, $\{x \in \mathcal{H} \mid E_{h,x} > 0\}$, where $\mathcal{E} \subseteq \mathcal{H} \times \mathcal{H}$ is the set of edges in the topology, and the notation $E_{x,y}$ denotes the weight of the edge from node x to node y. Edge weights in the network graph represent transition probabilities between hosts in the network. For most networks, these will be uniform. However, ad hoc wireless networks might correlate edge weights to link quality between hosts to avoid agent migration over unreliable links.

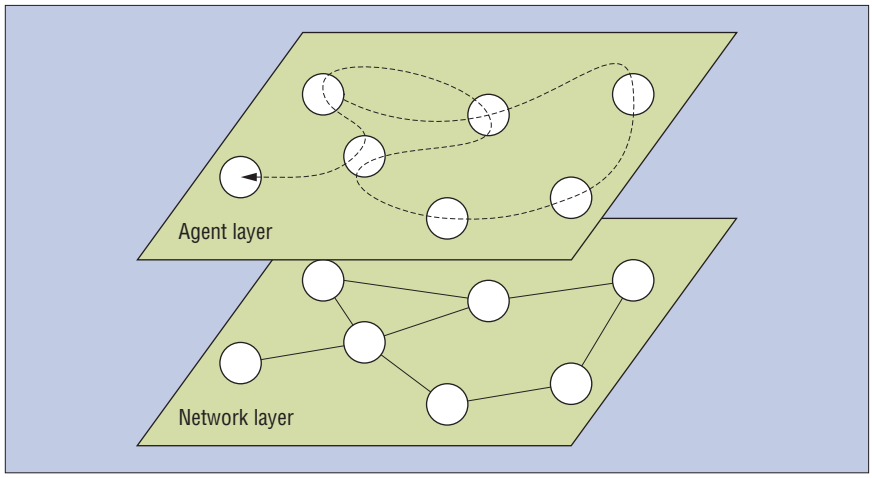


Figure 4. An agent randomly walking a peer-to-peer network as a means of service discovery.

Agents' actions consist only of hopping to a neighbor host from their current host. At each host, agents query for services, storing these data in memory (along with a time stamp). The network dictates agents' itineraries. The agent randomly selects successor hosts for migration from the set of available neighbor hosts in the network.

Service choreography and the effects of early and late binding on manets

A basic property of manets is their highly dynamic and uncertain nature, which can result in inefficiency and complete system failure if ignored.[8] To operate effectively, agents at the application layer might need to reason about network state.[2,9] Consider a mobile agent that must visit several hosts sequentially—for example, as part of a data collection task such as checking battery levels. For any given network topology, several possible migration itineraries might exist. An agent that considers the manet's properties, such as topology and signal strength, in selecting its itinerary is likely to require less time and resources to complete its task. However, the network's dynamic nature further complicates this problem—any static itinerary might quickly become obsolete. An agent might improve its performance by considering such dynamics and updating its itinerary through reasoning about the manet's uncertainty—for example, predicting future topological change. By reasoning on the current and predicted network state, an agent is more likely to return timely results, potentially preventing costly system failures.

Figure 5 illustrates this ability. An agent

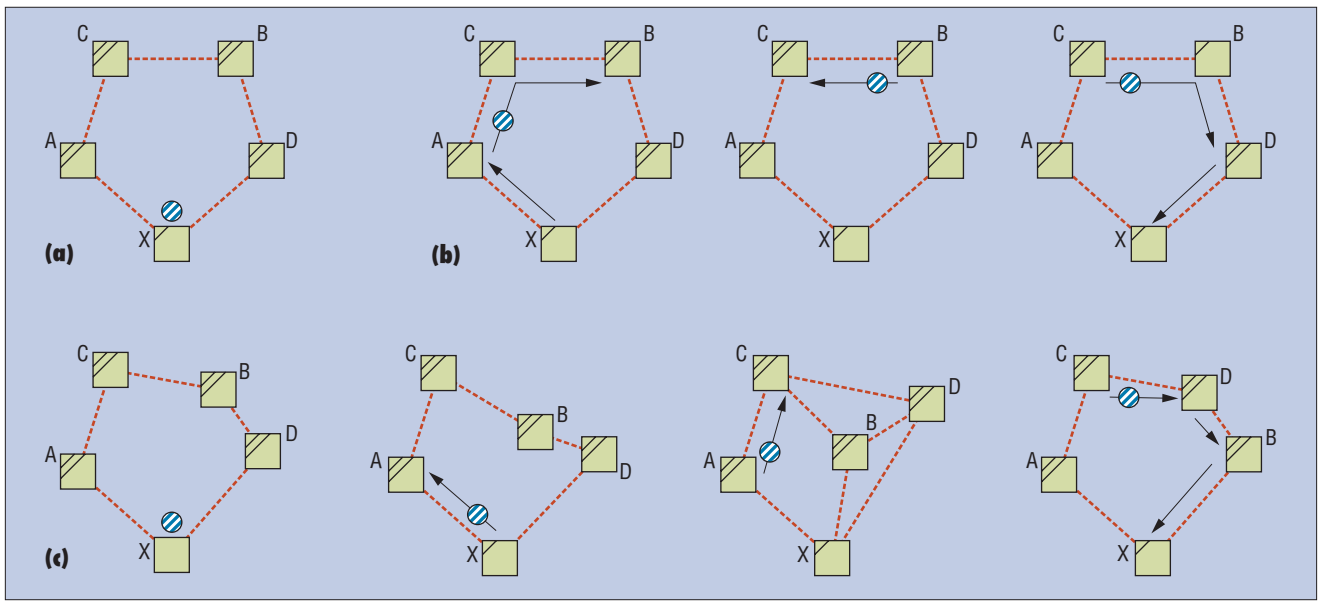


Figure 5. Agents that perceive and reason about the network's properties can adapt to changing conditions and use available resources such as time, bandwidth, processor cycles, and power more efficiently. (a) An agent from X is to collect data from hosts A–D. (b) Itinerary A, B, C, D produces redundant network hops. (c) By sensing network topology and developing a more efficient tour, the agent can avoid redundant hops.

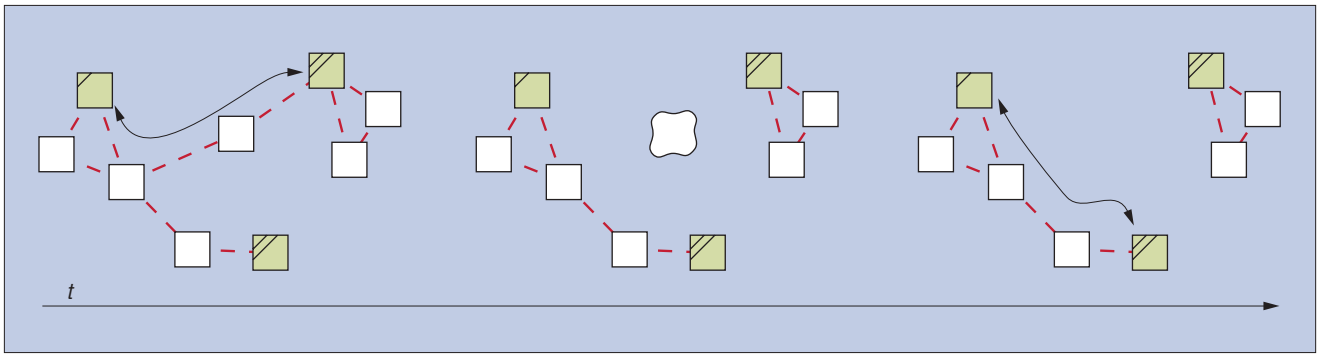


Figure 6. Agents are sent from a host to use a service. If a network failure disconnects that provider, knowledge of available alternative services lets the agents replan and recover from the failure.

must collect data from hosts {A, B, C, D} in figure 5a, starting from host X. Visiting the hosts in that order creates redundant network hops (see figure 5b). To migrate from host A to B, the agent must be relayed through host C. It immediately returns there in following the itinerary, and then is relayed through B again in continuing on to D. This extraneous network traffic wastes time, bandwidth, power, and other resources on each of those hosts. To avoid these extra hops, the agent must sense and plan on the network topology, developing the itinerary {A, C, B, D}.

In addition, to avoid similar situations as conditions change, the agent must continually poll the network and replan (see figure 5c). The agent must perceive and react to the network state, as well as reason about available services and resources. Agents traveling to specific targets are subject to required services becoming unavailable through events such as host outages, transmission delays, and link disruptions. Disseminating knowledge of available services lets agents search for alternatives and respond to such failures.

Figure 6 illustrates an example in which agents recover from a network failure by switching to an alternative service provider. By regularly consulting knowledge of available services, agents can direct migration through the network and continually move toward the closest provider. In this way, service-based agents are afforded more flexibility and an increased ability to react to changing conditions.

Figure 7 illustrates a typical manet scenario consisting of a set of wireless, mobile hosts. Agents on host A need to use some service, available on each host S_i. In figure 7a, host A is connected to S_0 and less directly to S_1. However, these nodes are in motion and the closest provider rapidly changes, as in figure 7b and figure 7c. S_0 eventually disconnects completely from A in figure 7d. Agents in such a dynamic world must be able to reason about available services and network state. Service knowledge lets an agent tolerate outages and discover options. Network knowledge lets an agent evaluate these options on the basis of factors such as topology and link quality—for example, switching from S_0 to S_1 in figure 7c.

Experiments

The experiments we conducted included one in simulation and another in the live PA-UWNT.

Simulation results

We conducted a small simulation using the *M*acro *A*gent *T*ransport *E*vent-Based *S*imulator (MATES).[10] We created 20 hosts inside a 120-sq.-meter area with a random walk mobility model that didn't preserve connectivity. At each iteration, every host had an equal probability of turning left or right 40 degrees, or maintaining direction. Hosts also moved forward one meter per iteration but never left the simulation area. Link quality is inversely proportional to the distance between hosts, each having a radio range of 30 meters. Migration times degrade with distance, requiring a single iteration at link qualities close to one.

Host A periodically spawned agents of eight types, each using a different migration logic to search for the service:

- *Packet:* Always travel to host S_0, failing if at any point no path exists from the current host to S_0.
- *Bundle:* Always travel to host S_0, waiting for it to reconnect when no path exists to it.
- *Inertia:* Always travel to host S_0, following the last known path and waiting if it's disconnected.
- *List:* Travel to host S_0. If at any point there's no path, travel to S_1. Fail if that host is ever disconnected.
- *Random walk:* At each iteration, migrate to a randomly chosen neighboring host. Wait if none are available. Succeed if that host provides the service.
- *P-early binding:* Consult the service registry before leaving host A, and choose the closest host offering the service. Proceed to that target as a packet agent.
- *B-early binding:* Consult the service registry before leaving host A, and choose the closest host offering the service. Proceed to that target as a bundle agent.
- *Late binding:* Consult the service registry at each step, and migrate toward the closest host offering the service. Fail if no hosts offer the service.

Agents expired after 750 iterations and were resurrected upon migration failure, such as disconnection in transit. This simulation also makes three large assumptions: correct network topology is instantly globally available, correct service knowledge is instantly globally available, and service matching is instantaneous.

Table 1 presents results for 36 trials, each consisting of 1,000 agents per class over 15,000 iterations. Most notably, the late binding agent is the most likely to succeed because it uses all available services, lowering the probability of expiring after never finding a service connected to its network component. Hop count and iterations are mostly informative, as the numbers are skewed against successful agents by long periods with no connected services. On any given run, the late binding agent will perform as efficiently as any agent, given the assumptions we mentioned.

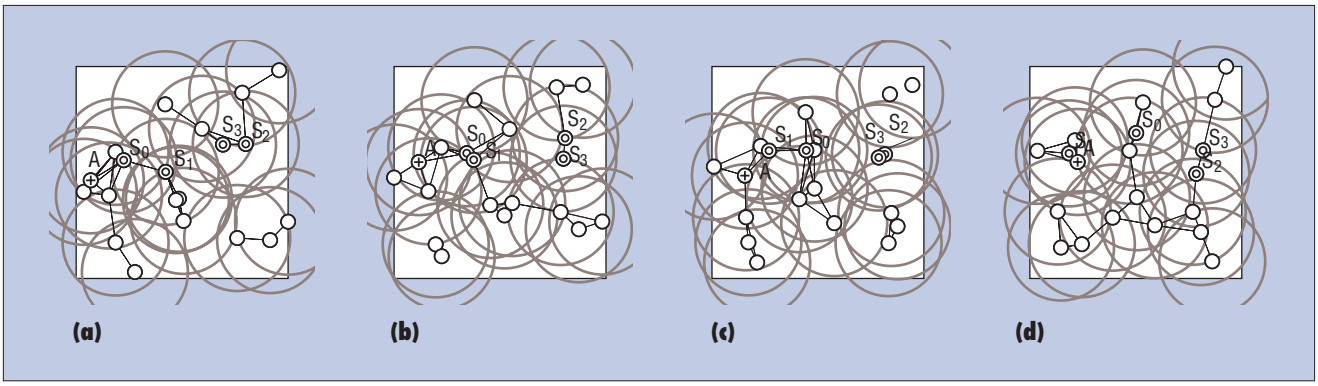


Figure 7. A representative manet scenario of randomly walking hosts. Agents on host A require a service available on each host S_i. The hosts' movements affect the set of connected providers and their distance. (a) Host A begins connected to hosts S_0 and S_1. (b) S_0 is moving to the right and S_1 to the left. (c) S_1 becomes the closest service provider to host A. (d) S_0 becomes disconnected from host A.

Live experiment results in the PA-UWNT

We conducted a second set of experiments in the live PA-UWNT. Our field test team carried on foot a total of nine mobile hosts (Tablet PCs) over an area approximately two city blocks by two city blocks. This area primarily included two large parking lots separated by several small trees, picnic benches, and so forth. The parking lots weren't full but did have several cars, including a few that were mobile, such as our facility's routine security patrol. None of these cars were deliberately part of our scenario; they were simply "natural" parts of the environment. At times during the experiment, one or more hosts were located at stationary locations, while the others were carried at typical walking speeds. At the start of the experiment, a single host contained a single instance of a service. This host's user walked in a constrained area at one end of the field test location. At various times throughout the experiment, two other hosts became (and ceased to be) service providers (other instances of the same service). One of these hosts had free rein of the field test location, while the other walked in an area at the opposite end of the field test arena. Each host maintained a service registry. Random walk service discovery agents were periodically spawned to update the service registries of the manet's hosts.

A single host (not one of the service providers) produced five agents every 10 seconds—one each of five types of agent: packet, bundle, p-early binding, b-early binding, and late binding (as previously described). Each of these agents had the task of obtaining the service and returning to its base host with the result. We released the different agent types simultaneously to help ensure that all agent types experienced as close to the same network conditions as all others. Only five of the original eight types were used, to minimize the possibility that network congestion would skew the results—for example, flooding the network would reduce the performance of all of the agent types.

In the live field test (results shown in table 2), the service-aware agents (in the case of p-early and b-early binding) performed at least as well as their unaware counterparts. The largest performance gain is when the b-early binding agents were 89.6 percent successful and the bundle agents were 72 percent successful. Curiously, the late binding agents were the best performing in the simulation but not in the live experiment. This is likely related to the fact that one of the key assumptions of the simulation was broken in the live experiment—the assumption that correct service knowledge is instantly, glob-

Table 1. Results of simulating several classes of agents searching for a service on a manet.

Agent type	Success (%)	Hops	Iterations
Packet	38	1.197	44.488
Bundle	88	4.064	342.880
Inertia	88	4.168	350.712
List	70	2.008	71.297
Random walk	74	17.999	648.556
P-early binding	65	1.881	66.842
B-early binding	72	2.202	93.339
Late binding	97	2.308	131.444
Summary	73	4.210	202.818

Table 2. Results from a live field test in the PA-UWNT.

Agent type	Number of agents	Success (%)	Time (seconds)
Packet	125	81.6	380.1
Bundle	125	72.0	375.5
P-early binding	125	81.6	355.7
B-early binding	125	89.6	32.8
Late binding	125	75.2	385.9
Summary	625	80.0	297.6

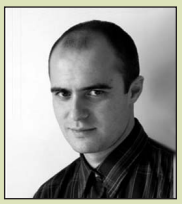
The Authors

Joseph Kopena is an undergraduate student in computer science at Drexel University. His research interests include service-based computing in disruption-prone networking environments, knowledge representation for engineering design, and low-cost mobile robotics for education. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; tjkopena@cs.drexel.edu.

Evan Sultanik is a student in Drexel University's dual BS/MS program in computer science. His research interests include mobile agents, service-based computing, and dynamic peer-to-peer networks. He's a charter member of Drexel University's student chapter of the IEEE Computer Society. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; evan@sultanik.com.

Gaurav Naik is a graduate student in the Department of Electrical and Computer Engineering at Drexel University. His research interests include distributed systems, discrete event systems, and network security. He received his BS in electrical and computer engineering from Drexel University. He's a student member of the IEEE and the ACM. Contact him at the Dept. of Electrical and Computer Eng., Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; gnaik@minerva.ece.drexel.edu.

Iris Howley is an undergraduate student in the Department of Computer Science at Drexel University. Her research interests include Semantic Web services and mobile ad hoc networks. Contact her at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; ikh23@drexel.edu.



Maxim Peysakhov is a PhD candidate in the Department of Computer Science at Drexel University. His research interests include nature-inspired algorithms, evolutionary computation, automatic design generation, and software agents. He received his MS in software engineering from Drexel University. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; peysakhov@cs.drexel.edu.

Vincent A. Cicirello was formerly a research scientist in the College of Engineering at Drexel University and is now an assistant professor of computer science at the Richard Stockton College of New Jersey. His research interests include applied artificial intelligence, multiagent systems, machine learning, planning and scheduling, biologically inspired computing, and anytime algorithms. He received his PhD in robotics from Carnegie Mellon University. He is a member of the AAAI, the ACM, the IEEE, and the Society for Industrial and Applied Mathematics. Contact him at the Richard Stockton College of New Jersey, Pomona, NJ 08240; vincent.cicirello@stockton.edu.

Moshe Kam is the Robert Quinn Professor of Electrical and Computer Engineering at Drexel University and the director of Drexel University National Security Agency Center of Excellence in Information Assurance Education and Drexel's Data Fusion Laboratory. His research interests include decision theory, detection and estimation, sensor fusion, dynamic systems, and network security. He received his PhD in electrical and computer engineering from Drexel. He's a fellow of the IEEE. Contact him at the Dept. of Electrical and Computer Eng., Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; kam@minerva.ece.drexel.edu.

William Regli is an associate professor in the Department of Computer Science in the College of Engineering at Drexel University. His interests include interdisciplinary research spanning engineering design, graphics, and modeling and intelligent systems. He received his PhD in computer science from the University of Maryland at College Park. He's a member of the ACM, the IEEE, the AAAI, and Sigma Xi. Contact him at the Dept. of Computer Science, Drexel Univ., 3141 Chestnut St., Philadelphia, PA 19104; regli@drexel.edu.

ally available. With the frequency that we started and stopped instances of the service on two of the experiment's hosts coupled with a less-than-instantaneous updating of the service registries, the late binding agents were more prone to the effects of inaccuracies in the service registries. Since the late binding agents (the most service-aware of the five agent types we considered) relied on this information the most, inaccuracies had the largest effect on them. The b-early binding agents are the performers in this live experiment likely because they chose what appeared to be the closest service provider, stuck with that choice, and used network awareness[1,2,9] to efficiently find their way to that provider.

Manets will form a significant part of the communications infrastructures first responders and other emergency personnel use in the very near future. The software infrastructure for these distributed and dynamic environments will need to interoperate with emerging Web Services standards. The convergence of these technologies, while promising to enable new capabilities for communication, collaboration, and situation awareness, introduces profound multidisciplinary system-engineering challenges. While service-based computing and multiagent systems are certainly going to be key elements in these systems, the manet environment's dynamics and complexity require new approaches and tools for service registration, service discovery, and service choreography. The approach we advocate uses a cross-layer

design that enables intelligent software agents to reason about network and service dynamics so that practical, effective systems can eventually be deployed in support of homeland security personnel. Our empirical studies indicate that this approach might have significant practical benefits in helping intelligent software systems adapt to the wireless environments first responders face. ■

References

1. V.A. Cicirello et al., "Designing Dependable Agent Systems for Mobile Wireless Networks," *IEEE Intelligent Systems*, vol. 19, no. 5, 2004, pp. 39–45.
2. J. Kopena et al., "Network Awareness and the Philadelphia Area Urban Wireless Network Testbed," *AI for Homeland Security: Papers from the 2005 AAAI Spring Symp.*, AAAI Press, 2005, pp. 70–75.
3. E. Sultanik et al., "Secure Mobile Agents on Ad Hoc Wireless Networks," *Proc. 15th Innovative Applications of Artificial Intelligence Conf.* (IAAI 03), AAAI Press, 2003, pp. 129–136.
4. M. Peysakhov and W. Regli, "Ant Inspired Server Population Management in a Service Based Computing Environment," *Proc. IEEE Swarm Intelligence Symp.*, IEEE Press, 2005, pp. 357–364.
5. J. Hendler, "Agents and the Semantic Web," *IEEE Intelligent Systems*, vol. 16, no. 2, 2001, pp. 30–37.
6. E. Sultanik and W.C. Regli, "Service Discovery on Dynamic Peer-to-Peer Networks Using Mobile Agents," to be published in *Agents and Peer-to-Peer Computing, 2nd Int'l Workshop (AP2PC 04), Revised and Invited Papers*, LNCS, Springer, 2005.
7. S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks and ISDN Systems*, vol. 30, nos. 1–7, 1998, pp. 107–117.
8. J. Wu and I. Stojmenovic, "Guest Editors' Introduction: Ad Hoc Networks," *Computer*, vol. 37, no. 2, 2004, pp. 29–31.
9. M. Peysakhov et al., "Network Awareness for Mobile Agents on Ad Hoc Networks," *3rd Int'l Conf. Autonomous-agents and Multi-agent Systems* (AAMAS 04), IEEE CS Press, 2004, pp. 368–376.
10. E.A. Sultanik, M.D. Peysakhov, and W.C. Regli, *Agent Transport Simulation for Dynamic Peer-to-Peer Networks*, tech. report DU-CS-04-02, Drexel Univ., 2004.

For more information on this or any other computing topic, please visit our digital library at http://computer.org/publications/dlib.