Network Awareness and the Philadelphia Area Urban Wireless Network Testbed

Joseph B. Kopena
Kris Malfettone
Evan Sultanik

Vincent A. Cicirello
Andrew Mroczkowski
Moshe Kam

Maxim Peysakhov
Gaurav Naik
William C. Regli

College of Engineering
Drexel University
3141 Chestnut Street
Philadelphia, PA 19104

Abstract

This paper overviews the Philadelphia Area Urban Wireless Network Testbed (PA-UWNT) project and several applications of artificial intelligence therein. PA-UWNT is a research and development effort in mobile and ubiquitous computing, focusing on communication and collaboration between first responders and other emergency personnel. Support systems in these environments face a number of challenges such as a lack of in-place infrastructure, frequent network disruptions, and limited bandwidth and power. The Testbed, consisting of robust networked computing platforms along with access to public and private locations, enables the project to test, evaluate, and develop new approaches to effectively supporting users in these domains. In this work, agents that reason on network state and available services in conducting information dissemination and collection tasks are proposed and evaluated.

Introduction

Drexel University's College of Engineering has been working with local law enforcement and transportation officials to study research problems in enabling police, fire, security, and other emergency personnel to effectively communicate and collaborate at disaster scenes. This effort has included development of the Philadelphia Area Urban Wireless Network Testbed (PA-UWNT) (Cicirello *et al.* 2004), a mobile ad-hoc network (MANET) consisting of PDAs, tablet computers, and laptops. Such systems provide a "bring your own network" solution to communications and management in infrastructure-free environments where power, network, and other computing-related resources are likely to be non-existant or inoperable. Applications include but are not limited to: coordinated police presence at large public events, medical personnel at an accident scene, and emergency responders to a natural disaster as well as scenarios in homeland security and military domains. The Testbed is an integration of the Extendable Mobile Agent Architecture (EMAA) (Lentini *et al.* 1998), a mobile agent platform developed by Lockheed Martin's Advanced Technology Laboratories; 802.11b wireless networking with multi-cast ad-hoc routing; and lightweight computing platforms such as PDAs and tablets.

Using the Testbed, the project is able to conduct experiments in actual urban settings and evaluate the real-world efficacy of developed approaches. Through dialogue between Drexel University, the Philadelphia Police Department, Amtrak, the Southeastern Pennsylvania Transportation Authority (SEPTA), University City Police, and Drexel University Security, the project has established areas of interest within the City of Philadelphia that present challenges to the eventual deployment of such a system. A map of these areas in Center City Philadelphia is given in Figure 1. Pictures from some of these areas are shown in Figure 2.

These regions include widely varied terrain with significant effect on wireless networking and system communications: subterranean platforms and corridors/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 the combination of buildings and trees over an extended area creating special radio frequency and modulation needs. Effective systems in these environments must be able to operate in and adapt to changing conditions as users move through the city or are redeployed.

A hypothesis of the PA-UWNT project is that constructing such systems will require developments in computer networking, agent and service based computing, and security along with fundamental integration between each of these areas. At the application level, several properties of mobile computing and the real-world settings considered by the project make agent and service based computing particularly attractive. The complex, dynamic, and disconnected nature of mobile networks quickly overwhelm centralized controls and data distribution. Sophisticated and arbitrary reasoning may be required to maximally utilize resources, even during network transit. Flexible matching, choreography, and composition may be required to utilize the possibly heterogenous mix of available resources. These concerns are addressed within the PA-UWNT project by developing applications within an agent and service based paradigm. This paper outlines part of this approach through several experiments demonstrating the utility of integrating the network and agent layers and enabling agents to reason on current operating context.

The following section describes this approach in more detail. Simulations and real-world experimentation evaluating this approach are given in the next three sections. The remaining two sections discuss related work and make some closing remarks.

Network and Service Aware Agents

A basic property of MANETs are their highly dynamic and uncertain nature (Wu & Stojmenovic 2004), which can result in inefficiency and complete system failure if ignored. Agents at the application layer may need to reason on network state (Peysakhov *et al.* 2004; Artz, Peysakhov, & Regli 2003) to operate effectively. Consider a mobile agent that has to visit several hosts sequentially, for example as part of a data collection task such as checking battery levels. For any given network topology, there may exist several possible migration itineraries. An agent that considers properties of the MANET, such as topology and signal strength, in selecting its itinerary is likely to require less time and resources to complete its task. However, this problem is further complicated by the dynamic nature of the network—any static itinerary may quickly become obsolete. An agent may further improve its performance by considering such dynamics and updating its itinerary through reasoning about the

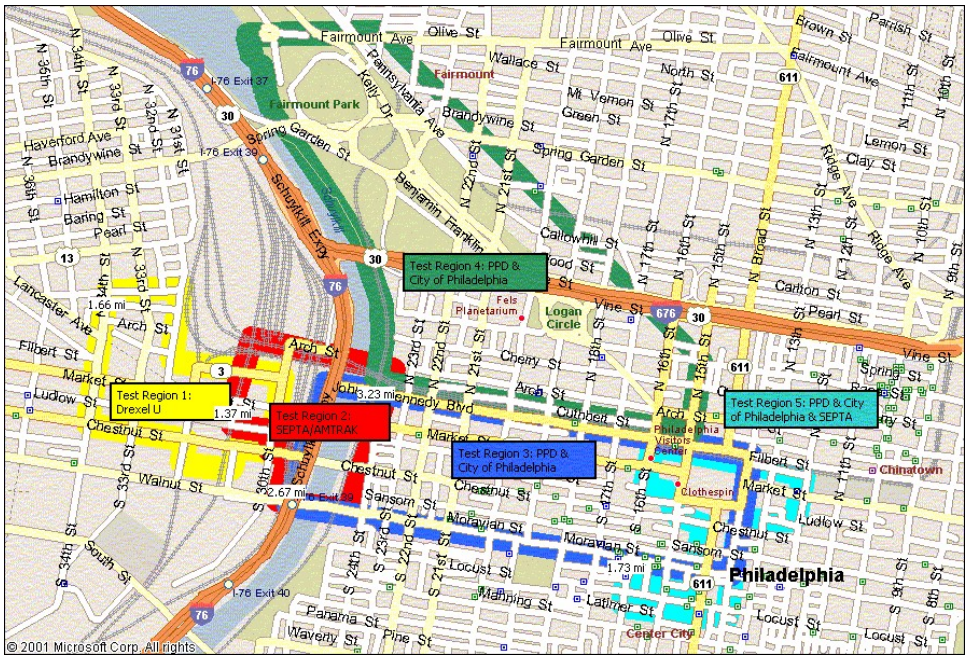


Figure 1: PA-UWNT Testing Areas within the City of Philadelphia.

uncertainty of the MANET, for example predicting future topological change. By reasoning on current and predicted network state, an agent is more likely to return timely results, potentially preventing costly system failures.

Figure 3 illustrates this ability. An agent has to collect data from hosts {A, B, C, D} in Figure 3(a), starting from host X. Visiting the hosts in that order creates redundant network hops, as shown in Figure 3(b). To migrate from host A to B, the agent has to be relayed through host C. It immediately returns there in following the itinerary, 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 it must continually poll the network and replan, as in Figure 3(c).

In addition to perceiving and reacting to network state, agents must also reason on 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 allows agents to search for alternatives and respond to such failures. Figure 4 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 may 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.

Network-Aware Data Synchronization Agents

In this experiment, there is a set of mobile network hosts, each of which require some piece of data about the distributed system state. The state of the hosts is changing over time as they move about in their physical environment. The hosts require a view of the others that is as up-to-date as possible. For example, if the hosts are a team of mobile robots mapping an area, then the task is to synchronize mapping information. If the hosts are tablet PCs on which a shared whiteboard application is functioning, then the task is to synchronize the state of the whiteboard.

Mobile agents are used to synchronize the data across the set of hosts. An agent carries with it the latest copy of the data that it has received about the state of each individual host. When the agent migrates to a host, it merges its data with the host's latest data, updating both the host and the agent. The agent then migrates to another host and continues this process. Timestamps are used in this data merge process to ensure that the agent keeps the most up-to-date information.

The experiment is run in MATES (Sultanik, Peysakhov, & Regli 2004), an open-source, lightweight, discrete-event simulator specifically designed for testing mobile agent control logic in MANET environments. The simulation uses a mobility model that moves a host randomly at each iteration. Hosts can wander out of radio range, but network partitions are prevented. There is always a route between two hosts, although it may consist of several hops. Routing data is made available to the agents. This means that an agent is aware of the neighbors of its current host as well as information on how to reach non-neighbor hosts. The term *destination* means the next host on the agent's itinerary. If hosts are within radio range, a link exists between them. Agents take exactly one simulation iteration to travel across a link. Agents have no knowledge of each other and do not communicate. Each simulation lasts for 100,000 iterations. The number of agents working collaboratively is varied from one to four. Four types of agents are considered as described below. Each type differs by the agent mobility pattern used.

- *FixedAgent.* Tries to visit every host on the network in a pre-specified order. Once the *FixedAgent* reaches the last host on its itinerary, it tries to migrate back to the first host and repeats the process iteratively. Note that the *FixedAgent* may migrate through hosts to reach its next destination if it cannot be reached in a single hop.
- *ReorderableAgent.* Tries to visit every host on its itinerary, migrating to the closest (fewest network hops) unvisited host on its itinerary instead of simply choosing the next host. If more than one host is considered closest, it randomly chooses one of them. Once the *ReorderableAgent* has visited all of the hosts on its itinerary, its itinerary is reset and

Figure 2: Examples of urban testing areas in Philadelphia.

it begins touring the network again. The ReorderableAgent does not update intermediary hosts on its way to its next destination. These intermediary hosts are used only to route the agent to the next host on its itinerary.

- *RandomAgent.* Chooses its destination randomly from its neighbors. This agent does not maintain any record of hosts it has visited, nor does it have a set itinerary. The *RandomAgent* synchronizes on every migration since it always chooses its next destination from its neighbors.
- *GradientAgent.* Does not have an itinerary, but keeps track of visits to each host by incrementing a counter after a successful migration. The next destination of this agent is the neighbor with the least number of prior visits. If more than one host has an equal number of visits, the *GradientAgent* chooses randomly from among them. Like the *RandomAgent*, this agent updates every host to which it migrates.

System performance is measured in terms of the team of homogenous agents present in the system (may be a single agent). The objective that we call out-of-dateness is defined as:

$$\text{OutOfDateness} = \sum_{h \in H} \max_{g \in H} (T - T_h(g)),$$

Where H is the set of hosts in the MANET, T is the current time, and $T_h(g)$ is the time that host h was most recently updated about the state information of host g. Table 1 reports OutOfDateness results for different quantities of agents for each of the agent types on networks consisting of 20 hosts. Results are averaged across 100,000 simulation time units.

Some observations can be made from these results. Notably, the two best migration logics for this problem of those evaluated are the Re-orderable Itinerary and the Gradient-based patterns. Both of these require some degree of network awareness. Further, the ReorderableAgent performs slightly better. This can be attributed to this agent type's usage of an itinerary to ensure it visits each of the hosts. The GradientAgent always chooses a neighbor as its next destination. The next best agent type in this example is the RandomAgent. This leads one to speculate the possible reasons that the FixedAgent performs worse than random. Recall that the FixedAgent uses a Pre-defined Itinerary,

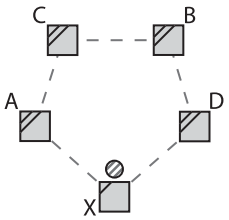
Type	# of Agents	Mean	Std. Dev.
Random	1	12915.547	8605.03
Random	2	5804.275	3584.82
Random	3	3911.574	2745.25
Random	4	2791.396	1859.25
Fixed	1	3877.396	775.72
Fixed	2	4300.934	947.05
Fixed	3	4667.739	1021.92
Fixed	4	4584.596	996.77
Reorderable	1	1593.642	335.38
Reorderable	2	1127.592	271.91
Reorderable	3	909.967	229.27
Reorderable	4	780.212	202.48
Gradient	1	2346.064	722.29
Gradient	2	1474.223	512.70
Gradient	3	1127.283	434.06
Gradient	4	893.248	349.83

Table 1: Performance of several different migration logics for data synchronization agents on a 20-host MANET.

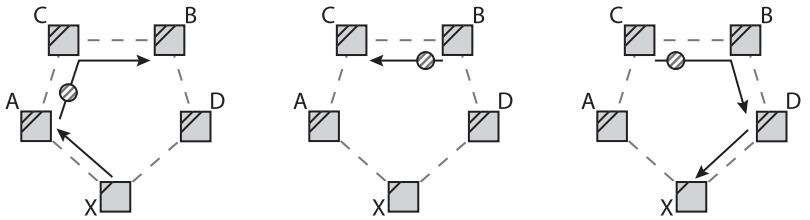
using multi-hop routes if necessary to visit and update the set of hosts in a pre-specified order. If the network topology is static, then one can handcraft an efficient pre-defined itinerary. Or if the network is fully-connected then any pre-defined itinerary would be as good as any other since every host would be a single hop away from any other. But given the dynamic nature of a MANET, a fixed itinerary can lead to migration involving long routes between consecutive stops on the itinerary.

Services and Late-Binding on MANETs

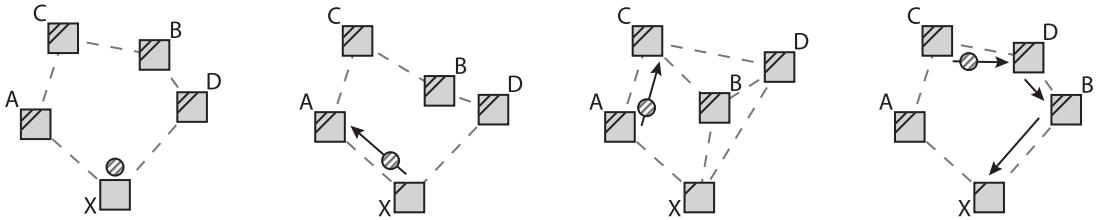
Figure 5 illustrates a typical MANET scenario consisting of a set of wireless, mobile hosts. Agents on host A need to use some service, available on each host S_i. In Figure 5(a), host A is connected to S_0 and less directly to S_1. However, these nodes are in motion and the closest provider rapidly changes, as in Figures 5(b) and 5(c). S_0 eventually disconnects completely from A in Figure 5(d). Agents in such a dynamic world must be able to



(a) An agent from X is to collect data from hosts A, B, C, and D.



(b) The itinerary A, B, C, D produces redundant network hops. In migrating from A to B, the network will relay the agent through C under this topology, only to have it return there later in the itinerary.



(c) By sensing network topology and developing a more efficient tour, the agent can avoid the redundant hops in Figure 3(b). This plan must be continually updated in response to the dynamism of network conditions, e.g. replanning as topology is changed by movement of the hosts.

Figure 3: Agents that perceive and reason on properties of the network may adapt to changing conditions and use available resources such as time, bandwidth, processor cycles, and power more efficiently.

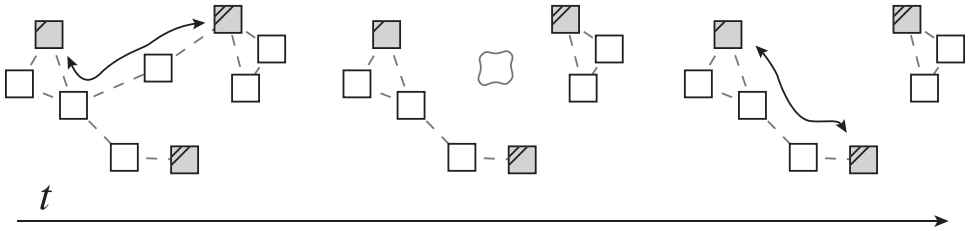


Figure 4: Agents are sent from a host to utilize a service. If a network failure disconnects that provider, knowledge of available alternative services allows the agents to replan and recover from the failure.

reason on available services and network state. Service knowledge allows an agent to tolerate outages and discover options. Network knowledge allows an agent to evaluate these options based on factors such as topology and link quality, e.g. switching from S_0 to S_1 in Figure 5(c).

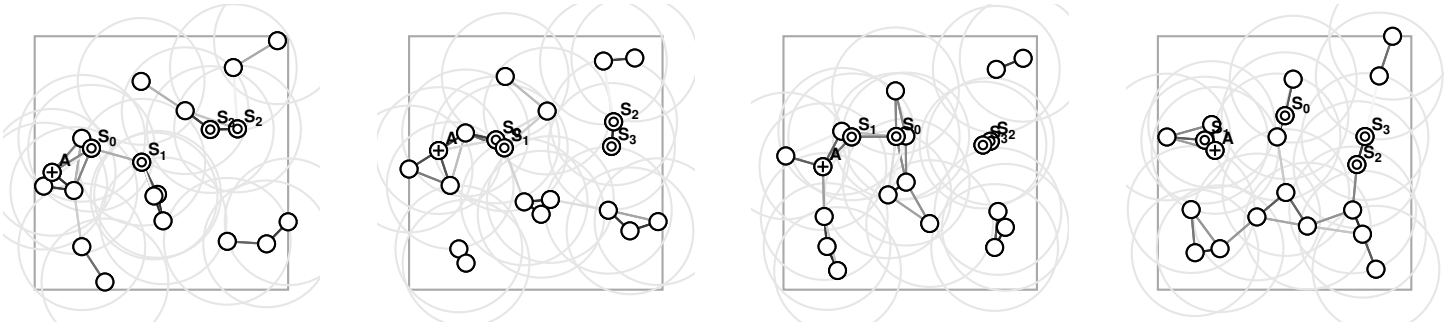
Demonstration. A small simulation, again conducted using MATES, demonstrates this point. Twenty hosts were created inside a 120 sq. meter area with a random walking mobility model that did not preserve connectivity. Each iteration, every host has an equal probability of turning left or right forty degrees, or maintaining direction. Hosts also move forward one meter per iteration but never leave the simulation area. Link quality is inversely proportion to the distance between hosts, each having a radio range of 30 meters. Migration times degrade with distance, requiring one 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 Agent.* Always travel to host S_0, failing if at any point there is no path from the current host to S_0.
- *Bundle Agent.* Always travel to host S_0, waiting for it to reconnect whenever no path exists to that host.
- *Inertia Agent.* Always travel to host S_0, following the last known path and waiting if it is disconnected.
- *List Agent.* Travel to host S_0. If at any point there is no path, travel to S_1. Fail if that host is ever disconnected.
- *Random Walking Agent.* At every iteration, migrate to a randomly chosen neighboring host. Wait if none are available. Succeed if that host provides the service.
- *P-Early Binding Agent.* 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 Agent.* 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 Agent.* Consult the service registry at each step and migrate toward the closest host offering the service. Fail if there are no hosts offering the service.

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

Table 2 presents results for 36 trials, each consisting of 1,000 agents per class over 15,000 iterations. Most notably, the late



(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.

Figure 5: A representative MANET scenario of randomly walking hosts. Agents on host A require a service available on each host S_i. Movement of the hosts affects the set of connected providers and their distance.

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
Average	73%	4.210	202.818

Table 2: Results of simulating several classes of agents searching for a service on a MANET.

binding agent is the most likely to succeed as it uses all available services, lowering the probability of expiring after never finding a service connected to its network component. Shown 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 above assumptions.

Experiment in Center City Philadelphia

The results of the experiment outlined here demonstrate in a live setting the benefit of reasoning on network state. Eight PDAs h_1 through h_8 were carried through Center City Philadelphia by users on foot, as shown in Figure 6. Each user covered a distance of approximately 1.2 miles over a period of roughly 45 minutes, walking in a loosely coordinated pattern to a series of predetermined destinations located along the routes shown in Figure 6(a). An example network topology encountered during the experiment is shown in Figure 6(b). The urban environment in which the experiment was conducted included such obstacles as metal bridges, other elevated structures, concrete and stone buildings, and radio interference from various electrical and radio emitting devices. Each device had direct line of sight to every other device only at the start and end of the experiment.

Twice a minute throughout the experiment, host h_1 produced a "naive agent" *A*, and a "MANET-aware agent" *M*, respectively representing network unaware and aware agents. Regardless of the current state of the network, agent *A* attempted to go to the next host in its itinerary . If a host h_x was unreachable at that point, *A* concluded h_x to be unreachable and moved it to the bottom of the itinerary. Whenever only unreachable hosts were left in the itinerary, agent *A* gave up. Agent *M* used knowledge of the current network state to minimize travel, using a greedy heuristic algorithm to generate improved itineraries. It "knew" which hosts were unreachable and did not attempt to reach them. It gave up when all reachable hosts had been visited. Information on three major characteristics of the agents' performance are given below. Table 3 summarizes the results.

Task survivability: How much of the task was completed? In this experiment, this equates to the number of hosts visited. Agent *M* visited 9.1% more hosts than agent *A*.

Time efficiency: How quickly does the the agent complete its task? This is measured by time per visited host. Although *M* did not visit significantly more hosts, it was notably more efficient. On average agent *M* outperformed *A* in this regard by 55%

Transit overhead: How much time was spent in transmission through the network rather than conducting the task? By avoiding multi-hop routes whenever possible and favoring links with stronger signal strength, *M* was on average 61% faster than *A*.

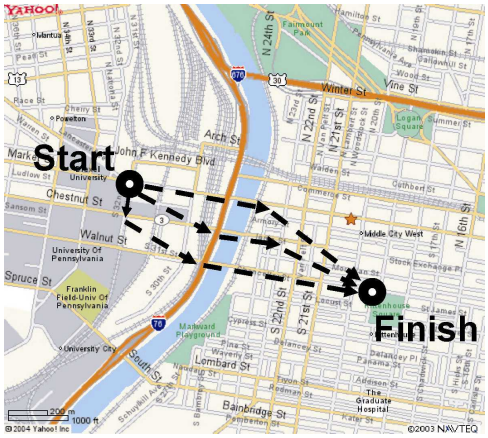
Related Work

Many routing schemes existto address dynamism and disruption in MANET environments (IEEE Computer 2004) . In supporting the application layer these make assumptions which do not necessarily hold, such as end to end connectivity and delay low enough for dialog-style interaction. In addition, requirements such as arbitrary routing priorities are not considered.

Learning, parameterized packet and routing techniques such as (Gelenbe *et al.* 2002) also make many assumptions which do not apply in these domains. Large delays and poor connectivity limit the ability of forward hosts to provide feedback and data to hosts along a reverse route, limiting their ability to update, converge, and adapt to chaging network conditions.

Work also exists on optimizing host mobility to promote connectivity and data delivery (Zhao, Ammar, & Zegura 2004), as well as using agents to probe and manage the network (Minar, Kramer, & Maes 1999). This work complements these efforts—agents that may reason on network state and available services are uniquely able to discover and utilize such capabilities.

Mobile agents reasoning on network state are similar to capsule packets in active networking (Tennenhouse *et al.* 1997). In a sense, this work involves that reasoning and how packets sense and act on their environment. However, such packets are typically network-level entities. Agents in this work are generally possessed with with greater reasoning and expanded roles.



(a) Approximate route taken by the experiment team.

(b) A network topology encountered during the experiment.

Figure 6: Progress of the experiment through Center City Philadelphia.

		Hosts Visited		Time per Host		Transit Time	
Agent Type	Samples	Avg.	Std. Dev.	Avg.	Std. Dev.	Avg.	Std. Dev.
Naive	46	4.261	2.08	31.1	47.7	34.5	118.6
MANET	83	**4.651**	1.77	**13.9**	16.5	**21.6**	57.5
Total	129	4.512	1.89	20.1	32.2	26.2	32.2

Table 3: Results from experiment in Center City Philadelphia

Current dfforts in service-based computing generally fall under the heading of web services (Communications of the ACM 2003), which has roots in formal programming languages and distributed computing. Especially relevant are semantic web services (McIlraith, Son, & Zeng 2001), a development of agent communication languages, process modeling, and knowledge representation. Research in this area is focused on developing expressive languages and reasoning mechanisms for describing and searching on services. The authors are unaware of work applying these results to MANET domains. In particular, dissemination of service descriptions through the network and matchmaking on computationally limited devices are not addressed in the literature. Most existing approaches are inappropriate under the difficult network and processing constraints of these environments. This work has attempted to show that meeting these challenges would greatly benefit applications in these settings.

Conclusion

Mobile ad-hoc networking environments stand to offer first responders and other emergency personnel a great tool for communication and collaboration. However, the dynamics and complexity of a MANET require new approaches to computer networking. To operate well in this setting, application layer agents may be required to be network-aware. The PA-UWNT enables this and other approaches to be tested and developed so that practical, effective systems may eventually be deployed in support of homeland security personnel.

References

Artz, D.; Peysakhov, M.; and Regli, W. C. 2003. Network metareasoning for information assurance in mobile agent systems. In *18th Int. Joint Conf. on Artificial Intelligence*, 1455–1457.

Cicirello, V. A.; Peysakhov, M.; Anderson, G.; Naik, G.; Tsang, K.; Regli, W. C.; and Kam, M. 2004. Designing dependable agent systems for mobile wireless networks. *IEEE Intelligent Sys.* 19(5):39–45. Spec. Issue on Dependable Agent Systems.

2003. *Communications of the ACM*, volume 46(10). Special Issue on Service-Oriented Computing.

Gelenbe, E.; Lent, R.; Montuori, A.; and Xu, Z. 2002. Cognitive packet networks: Qos and performance. In *Proc. of the Tenth Int. Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*, 3–9.

2004. *IEEE Computer*, volume 37(2). Special Issue on Ad-Hoc Networks.

Lentini, R.; Rao, G. P.; Thies, J. N.; and Kay, J. 1998. Emaa: An extendable mobile agent architecture. In *AAAI Workshop on Software Tools for Developing Agents*.

McIlraith, S. A.; Son, T. C.; and Zeng, H. 2001. Semantic web service. *IEEE Intelligent Systems* 16(2):46–53.

Minar, N.; Kramer, K. H.; and Maes, P. 1999. Cooperating mobile agents for dynamic network routing. In Hayzelden, A. L. G., and Bigham, J., eds., *Software Agents for Future Communication Systems*. Springer-Verlag. 287–304.

Peysakhov, M.; Artz, D.; Sultanik, E.; and Regli, W. C. 2004. Network awareness for mobile agents on ad hoc networks. In *3rd Int. Conf. on Autonomous Agents and Multi Agent Systems*.

Sultanik, E. A.; Peysakhov, M. D.; and Regli, W. C. 2004. Agent transport simulation for dynamic peer-to-peer networks. Technical Report DU-CS-04-02, Drexel Univ.

Tennenhouse, D. L.; Smith, J. M.; Sincoskie, W. D.; Wetherall, D. J.; and Minden, G. J. 1997. A survey of active network research. *IEEE Communications* 35(1):80–86.

Wu, J., and Stojmenovic, I. 2004. Special issue on ad hoc networks. *IEEE Computer* 37(2).

Zhao, W.; Ammar, M.; and Zegura, E. 2004. A message ferrying approach for data delivery in sparse mobile ad hoc networks. In *MobiHoc '04: Proc. of the 5th ACM Int. Symp. on Mobile Ad Hoc Networking and Computing*, 187–198.