Lei Fang, Panos J. Antsaklis, Luis Montestruque, Brett McMickell, Michael Lemmon, Yashan Sun, Hui Fang, Ioannis Koutroulis, Martin Haenggi, Min Xie, and Xiaojuan Xie, "A Wireless Dead Reckoning Pedestrian Navigation System," ACM Workshop on Applications of Mobile Embedded Systems, Boston, MA, June 6, 2004. .

A Wireless Dead Reckoning Pedestrian Tracking System

Lei Fang, Panos J. Antsaklis, Luis Montestruque, Brett McMickell, Michael Lemmon, Yashan Sun, Hui Fang, Ioannis Koutroulis, Martin Haenggi, Min Xie, and Xiaojuan Xie
Department of Electrical Engineering
Univ. of Notre Dame, Notre Dame, IN 46556, USA
{lfang,antsaklis.1}@nd.edu

1. INTRODUCTION

The ability to track the position of the user is an essential part of many applications [7, 4, 2]. It is well known that Global Positioning System (GPS) is limited as a navigation aid by its inability to provide static heading and its lack of availability when used around obstructions (terrain or man-made) or in the presence of jamming. Therefore, it is necessary to develop a positioning system which can complement GPS in GPS-compromised areas, which is not a trivial task.

With advances in computation, communication and sensing capabilities, large scale sensor-based distributed environments are emerging as a predominant mobile computing infrastructure [1]. Hundreds or thousands of small, inexpensive and low-power sensors, such as Berkeley Motes [5], can be quickly deployed to monitor a vast field. In this work we make use of this mobile infrastructure and new product innovations to build a pedestrian tracking system (PTS) that is capable of working either indoors or outdoors. Fig. 1 illustrates the system architecture. The system involves a Dead Reckoning Module (DRM), self-organizing wireless networks (sensor mote network) and a map database. Dead reckoning is used by DRMs to determine their positions and trajectories, and the sensor mote network is used to collect trajectory data and make them available at a base station for further processing. The DRM consists of the Leica Geosystems DMC-SX three axes accelerometer and magnetic compass (Leica Vectronix AG [6]) combined with a generic wireless controller board, with the radio, the processing and the power storage all integrated. The controller board for DRM and network nodes both use the open-source Berkeley Motes with the *TinyOS*. In the following, we call the DRM unit with the generic board, *NavMote*, and a network node, *NetMote*. The NetMotes play a critical role in NavMote calibration, trajectory data collection and exfiltration, all via wireless communication links. The trajectory data collected by NetMotes can be further exfiltrated to an information center for displaying, map matching, and other purposes.

WAMES '04, June 6, 2004, Boston, Massachusetts, USA.

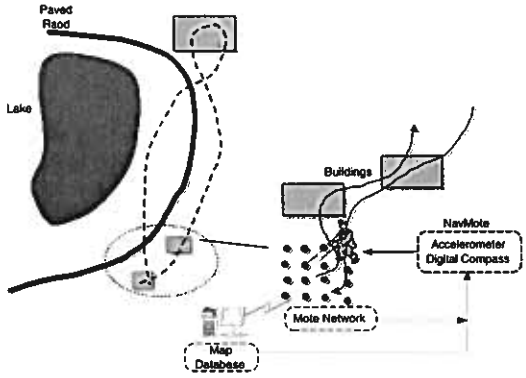

Figure 1: The pedestrian tracking system architecture

There are two challenges for the PTS: On the NavMote side, the resource-poor mote prevents using any computationally expensive algorithms, while size, weight and cost constraints need to be met. On the network side, a large amount of data need to be downloaded from the NavMote to one or more of (unreliable) NetMotes and forwarded to RelayMote with error-prone wireless channels. In this paper, we focus on the design challenges of the NetMote network. (For additional information regarding hardware, sensors, and DR algorithm, see [3].)

The present application differentiates itself from the traditional sensor network applications in several ways. First, large and sudden bursts of data must be delivered to the base station with a low latency. The transport of event impulses is likely to lead to varying degree of channel contention and network congestion. Second, these trajectory data are not redundant, requiring a near zero loss-rate. Third, but not last, the wireless links are highly unpredictable and unreliable, and bidirectional connectivity cannot generally be achieved. Commonly observed in the experiments are isolated nodes/clusters, leading to extended latency and even network deadlocks. To address these problems, we implement an IEEE 802.11 DCF (RTS/CTS/ACK) scheme to avoid channel contention, while guaranteeing a low-loss rate delivery with aggregated message ACKs. In our self-healing adaptive routing scheme, every node maintains a set of next-hop candidates. The "best" candidate is selected based on throughput estimate. Various deadlock cases due to missed

Lei Fang, Panos J. Antsaklis, Luis Montestruque, Brett McMickell, Michael Lemmon, Yashan Sun, Hui Fang, Ioannis Koutroulis, Martin Haenggi, Min Xie, and Xiaojuan Xie, "A Wireless Dead Reckoning Pedestrian Navigation System," ACM Workshop on Applications of Mobile Embedded Systems, Boston, MA, June 6, 2004. .

messages are identified and countermeasures are taken to eliminates these cases.

2. THE SYSTEM DESCRIPTION

2.1 Overview

A typical scenario involves the reconstruction of the space/time path a subject has taken from an initial to a final location. While the NavMote carried by the subject is out of range of the network, it uses the flash memory to store the accelerometer and compass data in real-time. Custom algorithms provide compensation for adverse sensor orientation and calibration, and for collection with possible coding of data. When the subject enters again into radio range of the network, the data captured by NavMote are extracted automatically via wireless communication and deposited in the NetMotes's EEPROM. These data that are distributed within the network are then via multi-hop transmissions sent to a designated special "relay" or "destination" mote (*RelayMote*) for further processing. At this stage, we assume the initial location is known and localization service is available to provide NetMote positions.

In what follows, we discuss the technical consideration that underpin the design of the NetMote network while the detailed design is presented in the next subsection.

The media access control plays a significant role in the performance of managing bursts of data in a wireless shared medium. A number of sensor networks use CSMA or variants for medium access. For example, the TinyOS platform uses a simple CSMA MAC. CSMA suffers from the well-known hidden terminal problem in multihop environment. IEEE 802.11 utilizes an RTS/CTS exchange to eliminate hidden terminals. Although the signaling cost of the RTS/CTS exchange is high for sensor networks where the packet size is quite small, the scheme is justified for the PTS application due to the burst traffic characteristics; for a three-minute walk, NavMote has approximately 6 KBytes data, which are often downloaded to several NetMotes. Every NetMote then tries to transmit the data in its buffer of size 128×16 Bytes in one RTS/CTS exchange. Hence the number of RTS/CTS exchanges is quite small compared to the data size. Random delays are also introduced in addition to backoff as suggested in [9]. To further reduce energy cost, ACK is used in an aggregated manner, not for every single packet.

In the course of network operation, NetMote and link failures may lead to arbitrary and unsupport topology. The RelayMote is required to use the active topology for extraction of information from the network and distribution of control packets to the NetMotes. The algorithm which we have used to satisfy the requirements of reliable routing is a distributed leader election algorithm. The algorithm is used to construct a minimum-hop spanning tree, such that each NetMote knows a "parent node" closer to the RelayMote. At each hop, the NetMote receives a packet and retransmits it to upper level. All the NetMotes keep the number of transmission retries as the throughput estimate. The best parent node, which has least number of retries, is selected. Max retried times is also set to prevent from forwarding data to inactive or dead NetMotes. If there is no "good" parent available, then the NetMote holds data till the next spanning tree update. The watchdog function is another essential part of the scheme to ensure a robust routing protocol. For instance, silent motes caused by CTS can be unlocked automatically even in case of missing messages or software task deadlines.

2.2 NetMote Network Middleware Services

The software supporting the PTS application is written using TinyOS/NesC [8]. The software is organized into a stack consisting of three layers; the application, middleware and operating system or O/S layers. There are three primary *application services*: *Coordinator*: This service manages the interaction of the other application services; *Dump*: This service downloads trajectory data from the NavMote to the NetMote's EEPROM; *Exfile*: This service streams data in a NetMote's EEPROM to the network's RelayMote (base station).

There are five primary *middleware services*: *Ping*: NavMote uses this service to determine if it is in the vicinity of a NetMote; *Backbone*: This service builds and maintains a robust minimum hop spanning tree from all NetMotes to the RelayMote; *Clock Synchronization*: This service maintains a global clock variable across all nodes in the network; *Localization*: This service initializes and maintains a variable representing the NetMote's physical position; *Telemetry*: This service periodically sends packets down the network's backbone to the RelayMote. The telemetry packets contain information about the network's current configuration. This service is used by the RelayMote to build a picture of the entire network.

The *O/S services* are software components interfacing directly to the mote's physical resources such as the UART, radio, sensors, clock, random number generator, and EEPROM (logger).

The services in the middleware layer are responsible for setting up and maintaining the network infrastructure required to recover data from the NavMote in a flexible and reliable fashion. As soon as a NetMote is reset, it starts up these middleware services in a specific order. The *Clock Synchronization* and *Localization* services are started first in order to initialize the mote's clock and location variables. Once these services have stabilized, a signal is issued which starts the *Backbone* service. This service automatically builds a minimum hop spanning tree from all NetMotes to the RelayMote. The service is designed to detect changes in link quality that would adversely effect the network connectivity. Upon detecting such changes, the backbone service reconfigures it's spanning tree to restore network connectivity. Finally, once the backbone service has stabilized, it issues a signal which starts the telemetry service. The telemetry service simply sends packets to the RelayMote that provide the user with a global view of the network's connectivity.

The NavMote uses the *Ping* service to find the network. Upon finding the network, it uses the application layer's *Dump* service to download the trajectory data to the network. The dump service works in an opportunistic manner, dumping as much data as possible to the nearest available NetMote's EEPROM. The download then switches to another download should the original dump stream be interrupted. Having multiple NetMotes store the NavMotes data is desirable since a single NetMote may not have enough EEPROM to store all of the trajectory data and the subject may be moving from being close to one NetMote to being closer to a different NetMote.

Lei Fang, Panos J. Antsaklis, Luis Montestruque, Brett McMickell, Michael Lemmon, Yashan Sun, Hui Fang, Ioannis Koutroulis, Martin Haenggi, Min Xie, and Xiaojuan Xie, "A Wireless Dead Reckoning Pedestrian Navigation System," ACM Workshop on Applications of Mobile Embedded Systems, Boston, MA, June 6, 2004. .

The *Exfile* service is used to transmit data logged in the NetMote's EEPROM to the RelayMote. The *coordinator* on the NetMote ensures that the *Dump* and *Exfile* service operate in a mutually exclusive manner. In this way, the Exfile service does not begin until the download from the NavMote has stopped. Upon starting, the Exfile service begins reading data stored in EEPROM and storing it in a 128 line (16 Bytes/line) buffer. Once this buffer is full, the service issues a request-to-send (RTS) to the next-hop on the backbone's spanning tree. All nodes (except the sender and next-hop) hearing the RTS are immediately set to a dormant state, thereby establishing a *basic service cell* (BSC) in which the sender and next-hop can communicate with little interference from other NetMotes. Upon receiving a clear-to-send (CTS) from the next-hop, the Exfile service begins transmitting the buffered data, line by line, to the next-hop. Every four transmitted lines is acknowledged (ACK) by the next-hop. Should the sender miss an ACK or the next-hop miss an expected message, then they issue a clear (CLR) message that resets the basic service cell so it is ready for the next RTS message.

The Exfile service essentially streams data from all NetMotes with logged data to the RelayMote. The trajectory data transmitted by the Exfile service follows the same spanning tree used by the *Telemetry* service packets. The final destination for both message streams is the RelayMote. Upon being caught by the RelayMote, the packets are forwarded over the RelayMote's UART to a PC that is running a Java graphical user interface (GUI) called *NetConsole*. The GUI displays telemetry packets that are forwarded to the RelayMote from the network. The data contained in these packets allow the GUI to display the NetMote and NavMote position and neighborhoods. The data also allow the GUI to display the routes and connectivity between NetMotes. Finally, the NetConsole dumps the decoded data packets to a file for subsequent post-processing.

3. CONCLUSIONS

In this paper, we describe the development of a pedestrian tracking system that uses three-axes accelerometer and magnetic compass (Leica Geosystems) and a mote network consisting of Berkeley motes.

The system was examined in both indoor and outdoor environments. One group of experiments took place in swamp terrain. The subjects took 5-minute or 3-minute walks through the test course. The principal performance measures are:

1. Exfiltration Performance Measure (Rec Rate): The number of exfiltrated data lines / number of downloaded data lines.
2. Download Performance Measure (DTime, DTput): Download Time, number of downloaded data bytes/sec.
3. Exfiltration Performance Measure (ETime, ETput): Exfiltration Time, number of exfiltrated data bytes/sec.
4. Trajectory Reconstruction Accuracy: Plot of reconstructed trajectories viewed against aerial map of region.

Table 1 shows the observed performance metrics for the 4 trials conducted during the test. A total of 21 NetMotes are deployed in a 3×7 grid, with a grid size around 7 feet. The table shows that we were successful in recovering 99% of the NavMote's trajectory data, with download times on the order of 3min/3min walk (70 data bytes/sec) and exfiltration times on the order of 1min/3min walk (200 data bytes/sec). Extensive experiments show that the accuracy of walk distance estimate is with $\pm 3\%$ and heading accuracy $\pm 1°$.

	Rec Rate	DTime (sec)	DTput (b/s)	ETime (sec)	ETput (b/s)
1	99.9%	200	84	69	242
2	100.0%	189	72	111	127
3	100.0%	189	79	67	222
4	99.9%	184	63	55	213
Avg.		190.5	74.5	75.5	201

Table 1: System Performance

Our experiences tell us that sensor nodes and wireless links are much unpredictable and unreliable than one may think. Hence a simple yet robust MAC scheme/routing protocol are critical to any sensor network application. We focus more on the high throughput and low loss-rate aspects, without fully addressing the energy-aware network operation, which is in some sense more important for the sensor network applications. Other practical challenges include node localization, network security, privacy, and authentication. We believe however that our development offers a vision of future mobile environments that may emerge once ubiquitous wireless coverage becomes available.

Acknowledgment

The support of the DARPA/IXO-NEST Program (AF-F30602-01-2-0526) and of the National Science Foundation (CCR-02-08537 and ECS-02-25265) is gratefully acknowledged.

4. REFERENCES

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: A survey," *Comput. Networks*, vol. 38, pp. 393-422, 2002.

[2] J. Elwell, "Inertial navigation for the urban warrior," in *Proc. SPIE*, vol. 3709, 1999, pp. 196-204.

[3] L. Fang, P. J. Antsaklis, et. al, "Design of a wireless dead reckoning pedestrian navigation system - The NavMote experience," Technical Report, ISIS Lab, University of Notre Dame. [Online]. http://www.nd.edu/~isis/techreports/isis-2004-001.pdf

[4] H. Hashimoto, K. Magatani, and K. Yanashima, "The development of the navigation system for visually impaired persons," in *Proc. 23rd Annual EMBS Int. Conf.*, 2001, Istanbul, Turkey, pp. 1481-1483.

[5] J. Hill and D. Culler, "Mica: A wireless platform for deeply embedded networks," *IEEE Micro*, vol. 22, no. 6, pp. 12-24, 2002.

[6] "DMC-SX Digital Magnetic Compass - Operator Manual," Leica Vectronix AG, Switzerland.

[7] T. E. Starner, *Wearable Computing and Contextual Awareness*, Ph.D. Thesis, MIT, 1999.

[8] TinyOS - A component-based OS for networked sensor regime. [Online]. http://webs.cs.berkeley.edu/tos/

[9] A. Woo and D. E. Culler, "A transmission control scheme for media access in sensor networks," *MOBICOM 2001*, Rome, Italy, pp. 221-235.