Just-In-Time Sensor Networks

Gary V. Yee, Brian Shucker, Joe Dunn, Anmol Sheth, Richard Han
Computer Science Department
University of Colorado, Boulder
Boulder, Colorado 80303
{Gary.Yee, Brian.Shucker, Joseph.Dunn, Anmol.Sheth, Richard.Han}@Colorado.edu

Abstract

Much of the sensor networking research over the last six years depicts a similar picture of deployment. Specifically, a sensor network is deployed (either randomly or placed in a specific location), sits statically for several months collecting data, and adapts itself through various protocols. Yet this research often overlooks potential optimizations gained by adding motes to the network on-demand and within seconds. This paper introduces a shift in the traditional outdoor, static sensor network paradigm by considering the possibilities and limitations of a rapid, just-in-time (JIT) deployment. We look at what sensor networks would be like if we could add new nodes in real-time and how existing protocols would change. We also observe that the technology to utilize this paradigm for outdoor deployments exists as an extension of a low-cost, commercially accessible solution; i.e., a ball launcher or a cannon that tosses sensor nodes.

1. Introduction

Many areas of research in sensor networks deal directly with the ability to adapt to changing conditions. This has resulted in the ability to dynamically change attributes such as routing paths, MAC protocols, program images, and duty cycling. Yet there are several sensor network optimizations and adaptations that cannot be accomplished through software changes alone. The lack of hardware capabilities or a poor geographic layout of nodes are characteristics that create upper bounds on the ability of software protocols to optimize communication and coverage capabilities. For example, a linear topology of homogeneous, static nodes with only one route to the base station has a "hot spot" in it, but software protocols have no recourse for load balancing onto another route. Similarly, there is no software solution to make a network of static nodes cover an area with n redundancy if only $n-1$ nodes are in the area, even if a node detects that an important event has occurred and the user would like more detailed data. This would be analogous to the limitations a C programmer would face if all memory allocation had to be done statically.

This has resulted in recent research striving for more adaptability at the physical level of sensor networks by using automated deployment throughout the lifetime of a network instead of just in the initial phases. For example, technology such as UAVs or ground robots have been used to automatically manage networks for the purpose of guaranteeing levels of connectivity [3]. We categorize these works as *run-time deployment* solutions.

While we believe this is a promising direction of research, we argue that run-time deployment research only scratches the surface of what is possible. To the best of our knowledge, few people have considered the possible optimizations and adaptability of a sensor network if run-time deployments were done in real-time - i.e., within a few seconds of a request. We define a system that utilizes "just-in-time (JIT) deployment" as one where a heterogeneous set of sensor nodes may be deployed: (1) inexpensively in terms of total cost of hardware, (2) within a communication radius of a targeted point, and (3) in the time-span of one order of magnitude slower than data moves across the network via multi-hop routing. We focus primarily on how JIT deployment impacts outdoor applications for the purposes of incremental searches, target tracking, redundancy optimizations, congestion and topology control, bulk data transfers, load balancing of routes, and rapid prototype design.

These benefits can lead to practical, potentially mission-saving functionality for a variety of sensor applications. Two real-world scenarios that could have benefited from JIT sensor networks include the search and rescue effort for the Philippine mudslides of February 2006 and the real-time intrusion detection, tracking, and classification application known as Project ExScal [8, 7]. In this paper, we further investigate how the rapid adaptability of JIT deployments could have benefited these scenarios.

This paper's primary objective is to further clarify the new possibilities of a JIT deployment and examine how this impacts our current analysis of sensor networks. We conduct our analysis under the practical limitations of using a mote-launching cannon for deployment. In section 2 we discuss various optimizations and applications that become feasible in the JIT deployment scenario. In section 3 we investigate application domain considerations of this approach. In section 4 we briefly describe the requirements and limitations for a mote-launcher that could carry out a JIT deployment.

2. JIT deployment

In this section we present some novel solutions that become available when working in the JIT deployment paradigm. We also explain how these solutions impact our analysis of classical sensor network problems.

2.1. Changing sensing capabilities

We consider the cases of two real-world, time-sensitive situations where JIT deployments help optimize sensor resource allocation and thus improve the quality of data detected by the network. We then discuss how JIT adaptability affects algorithms that rely upon random redundancy to extend the network lifetime.

2.1.1. Incremental search and target tracking. One of the advantages of JIT deployment is that resources do not have to be statically allocated. This is especially important when considering heterogeneous deployments where more sophisticated sensors may cost more and have shorter lifespans. With JIT deployments, we are able to implement incremental search strategies to efficiently allocate our different types of nodes with the goal of determining the source of a phenomenon.

To illustrate the incremental search, recall the Philippine mudslide scenario, where seismic sensors and sound detection gear was brought in to find survivors trapped beneath dozens of feet of mud. This was a hazardous scenario where rescuers also had to be careful of sinking into the "very moist, very soft" soil [1]. Clearly it would not have been adequate to do a single, uniform deployment of sensor nodes throughout the entire region.

A more complete solution would instead be to first deploy a wave of abundant sensors that could give broad coverage. This would then be followed by a deployment mechanism instantly deploying a concentrated set of nodes in locations determined from real-time data. After sufficient agreement among nodes in a suspected area, a final wave of rare, high-powered seismic sensors could be deployed to provide high-detailed data and call over rescue workers. In this way, we could rapidly place scarce resources in areas with some initial evidence of survivors.

Similarly, target tracking applications can make use of being able to deploy a heterogeneous set of nodes as a suspected target is moving. Project ExScal [8, 7], for example, attempts to create a "tripwire" system that can detect, track, and classify multiple intruders in real-time. While it incorporates a host of sensors such as a magnetometer, infrared, and acoustics sensors, one that did not make the final design is a video enabled sensor, which could at least help classify the intruder. This is presumably due to the power consumption and high cost that would be associated with statically allocating these uniformly across the region. Instead of eliminating this possibility altogether, however, we again propose that the network should wait until an intrusion is detected before allocating high-powered sensors to the relevant area.

2.1.2. Optimizing redundancy. Another area of sensor network research has concentrated on maximizing the network's lifetime in order to meet some guarantee of service. Many of these solutions attempt to take advantage of the amount of pre-existing sensor node redundancy in the network [11, 17]. For the task of preserving a specified level of coverage, the general approach is to formulate a subset of sensor nodes that should be turned on while the rest stay in a deep sleep state and periodically check if they are needed. These works also assume that motes are aware of their two-dimensional position.

We argue, however, that in a JIT sensor network the requirement of pre-existing redundancy in the network is unnecessary and potentially wasteful. Moreover, we propose the more cost and energy efficient method of filling gaps in sensing coverage rapidly and on demand. The savings in cost comes from the fact that redundancy becomes targeted instead of random, while the energy savings can be attributed to the lack of sensors in the network wasting energy by polling for calls to duty and sleeping. Furthermore, the network's recovery time is on par with current solutions, since they must wait for redundant motes to wake up from their deep sleep cycle before they can be activated.

There are several implications for current works on redundancy elimination with coverage guarantees: (1) the main responsibility of the problem's resolution is now pushed on to the deployer (i.e. cannon) instead of the resource constrained motes, and (2) the problem no longer becomes which set of sensor nodes to turn off or on, but instead:

- *Where?* What is the optimal position for a new mote to not only cover the current gaps, but also add redundancy to already covered regions?
- *When?* What is the optimal time to deploy an energy-constrained node given its finite lifespan and deployment errors?

- *What?* Given a heterogeneous set of motes, what is the optimal set of motes to deploy in terms of cost / performance?

2.2. Short-lived, virtual sinks

At SenSys 2005, Wan et al. [16] proposed the idea of using a sparse number of wireless, multi-radio virtual sinks that could be used to offload traffic from routes to the physical sink. We believe that this idea is enhanced in the JIT deployment paradigm.

For example, Wan states: "due to the relatively sparse required concentration of VSs. . . there is no assurance that a VS is adjacent to a congested region." While there is also no guarantee of this in the JIT deployment paradigm, it is at least possible to deploy virtual sinks in the general location of a congested area at the first signs of congestion.

This kind of topology control introduces a different kind of networking concept which we call *data-dependent topology*. In other words, instead of the network topology being based upon some arbitrary formation, additional routes are placed in areas where and when robustness of routes is needed. Virtual sinks make this even more useful since any node is allowed to request and receive a two-hop link back to the base station (i.e. one to the virtual sink, which reports directly to the base station).

Clearly this is something that can not be optimized via software-based protocols. Software protocols can do no better than to optimize routes within the existing fabric. This is significantly different with JIT sensor networks, where protocols can now call for the additional resources required to break through performance ceilings.

We believe there is even more room to explore the ideas of reforming network topologies in the JIT deployment paradigm. For example, what are the tradeoffs between the cost of deploying n motes to produce additional routes versus re-routing traffic through a pre-existing path? This is a common thread in the analysis of JIT deployment algorithms, since the low cost of sensor network hardware is what makes this feature economically feasible.

2.3. Targeted, bulk data transfers

Though the majority of modern sensor network data transmissions are relatively small, occasionally it may be appropriate to send a large amount of data to arbitrary subsets of motes in a network. One common usage of bulk data transfers comes when using an over-the-air reprogramming protocol such as Deluge [13]. We contend, however, that this solution is wasteful when used to reprogram a network that does not have a uniform application set. As a result of the flooding scheme utilized, motes that are not interested in the proposed code update end up using their scarce resources for the routing and storage of each program page. Alternative solutions such as Aqueduct [14] route code images using a minimum spanning tree to ensure that a much smaller number of uninterested motes are involved in forwarding data, but this still disturbs portions of the network.

Our proposal is a brute-force, over-the-air method that creates the absolute minimum interference to the wireless medium. Much like in years past when users received software updates through disks in the mail, we believe that this solution is also very appealing and feasible in the JIT deployment context. The main idea is to send a cheap *data ferry* equipped with only a radio, a cheap power source, and its data payload. When this node arrives at it destination it can begin broadcasting its payload via some pre-established protocol such as Aqueduct. At the same time, the rest of the network can remain unaware of the entire transaction. While this is not feasible today due to the high cost of hardware, we believe that the cost of a cheap data ferry will eventually reach the point where it offsets the cost of energy and bandwidth used to multi-hop the data.

This type of communication introduces a host of new questions which we leave as future work. Specifically, how much data must be transferred before sending a node physically becomes more efficient (both in terms of cost and data rate) than using the multi-hop network? Given the failure rate of communication between n motes relaying data, at what point does sending a node become more reliable?

2.4. Load balancing of routes

Given a typical tree topology sensor network, one of the general problems associated with the many-to-one communication scheme is that various routing techniques have the side effect of creating "hot spots" in the network. This tends to adversely affect the deployment in two ways: (1) areas around hot spots experience increased congestion and (2) hot spots tend to wear out faster than the rest of the network, eventually leading to premature disconnects in the network topology. While the prior is a significant concern in some sensor networks, there are many applications with such low throughput requirements that the latter issue, which affects all sensor networks, is more consequential.

As a result, many routing techniques have investigated the issue of load balancing through data aggregation and varying route selection [10, 15, 5]. We focus on the latter approach to load balancing. In this method, if a node determines that it is spending too much energy forwarding packets, data may be forced to take a path less traveled in order to ensure fairness.

While in general this appears to follow the theme of energy constrained sensor networks, we argue that these solutions are potentially suboptimal in the JIT deployment context. Consider the fact that the end results of fairness and optimal performance algorithms often conflict. Optimal solutions tend to use the shortest, most efficient routes, while fair solutions try to utilize a set of routes equally. In the case of load balancing, it is not uncommon to see algorithms pick routes that are *not* the shortest path. Instead, more hops are potentially taken, and for some routing algorithms the size of the packet is increased [10]. Even if we ignore the cost of energy used to calculate routes fairly, it is clear that fairness may lead to an increase in network power usage.

This leads us to an important observation: in a resource constrained environment, there are two ways to view optimal energy efficiency. In a scenario where energy resources are irreplaceable and finite, it is critical that gaps in coverage or disconnections in communication do not occur. Thus the main goal is to minimize the maximum power usage of any node in the network. The JIT deployment scenario, however, allows for low-cost motes to be easily replenished before or immediately after gaps occur from the unbalanced use of hot spots. Hence the energy consumption of the network as a *whole* becomes more critical than that of an individual hot spot.

At the same time we point out that deploying low-cost motes to replace various hot-spots in the network is not free. Thus, in addition to considering the global energy consumption of routing protocols in a network, we note that deployment costs must also be factored into future analysis.

2.5. Rapid prototype setup and maintenance

Real-world implementations and experiments are important to sensor network research, as pointed out by Jason Redi during his welcome speech at SenSys 2005. While several indoor testbeds and management tools [18, 12] are available for static sensor networks, we have yet to see a solution for researchers to easily test on an outdoor testbed.

By lowering the human effort and time requirements of deployment and the cost of the deployer, we envision that a JIT deployment kit could be used to deploy large testbeds and automatically maintain them. This would prove a great benefit to the community by not only allowing the prototyping of large outdoor deployments, but also encouraging users to test their communication protocols under outdoor RF-propagation conditions. In this type of setup, there may even be possibilities for deploying tethered sensor nodes for easy retrieval and clean-up.

3. Application domain considerations

Each sensor network has its own set of requirements that are important to consider when deliberating over the use of a JIT deployment. For example, this approach would be well suited for scenarios where conditions are difficult to predict a priori, but could be determined quickly after an initial deployment. Conversely, it would not be appropriate to use a JIT deployment when precise node placement or orientation is necessary. With respect to using a cannon to launch motes, users must be aware that there is no easy inverse for the deployment operation. Thus the retreival of motes must not be a critical requirement. Furthermore, there are many aspects of packaging that need to be analyzed. Should motes be sent in a package that lands on the ground, for example, users must be aware of the Fresnel zone effect that will lead to a decrease in a mote's effective communication range.

Throughout this work we have also mentioned that the costs and benefits of a JIT deployment need to be considered. While the number of motes used, the cost of hardware, and the effort of placing a node are important factors they do not form a complete model. A more thorough evaluation also needs to focus on the cost and value of a sensor network's primary contribution: data. In other words, users need to determine whether the value of additional data availability justifies its cost in deploying the necessary motes.

Several factors in determining the cost of data in a JIT deployment include the added packaging complexity C_p, the cost of sensors placed C_s, the cost of the deployer C_d, the time to deploy T_{deploy}, the time to retreive data T_{data} (includes time to join the network), and the network's total power consumption to send data Pwr. We can model the cost of data C_{data} by

$$C_{data} = (C_p + C_s) + C_d + f(T_{deploy}, T_{data}, Pwr)$$

The value of data, on the other hand, is a function of data latency L_{data}, the amount of data A_{data}, and the position of the data source Pos_{src}. We can model the value of data V_{data} by

$$V_{data} = f(L_{data}, A_{data}, Pos_{src})$$

Further analysis of these functions is dependant upon the application domain and is outside the scope of this paper. Nevertheless, these functions provide the basis of a cost/benefit analysis for a JIT deployment.

4. A JIT deployer: the cannon

An ideal node launching mechanism would have several key capabilities, most of which deal directly with making deployment rapid and automatic. These include:

- *Programmability*: The cannon should be able to execute deployment applications.
- *Rapid response*: The cannon should be able to respond to network commands and deploy at arbitrary locations in a few seconds.
- *Best-effort precision*: The user should have the expectation that most nodes will land within the communication radius of a mote's radio.

- *On-demand activation*: The motes should only be turned on just before deployment.
- *Multiple ammunition inputs*: The cannon should have a quickly changeable set of mote caches.

5. Related work

While there have been several publications over the last few years that have directly investigated the problem of optimizing sensor placement, none of them have addressed the impact of rapid deployment on sensor networking protocols. In this section we briefly summarize several selected works. Bredin et. al [3] provide an algorithm that guarantees k-connectivity by adding motes to an existing network. Clouqueur et. al [4] talk about optimizing a sensor network deployment for tracking a target traversing a region. Wang et. al [17] analyze novel protocols that dynamically configure a network to achieve guaranteed levels of coverage and connectivity. Dhillon et. al [6] propose an algorithm for attaining a certain level of coverage under imprecise detections given terrain attributes.

Prior work has also been done that discusses the possibilities of using UAVs and ground robots in sensor network deployment. Again, our work differs from these since they do not look at the possibility of rapid placement. Corke et. al [9] analyze an algorithm and provide initial results for a sensor network deployed with a manually piloted AVATAR helicopter. Batalin et. al [2] present an algorithm for deploying mobile sensor networks in order to learn more about an unknown environment.

6. Summary

We have proposed and considered the implications of the "just-in-time" sensor network. We drive this discussion by focusing on outdoor deployments that could use a mote launching cannon as its JIT deployer. The main goal of this paper is to encourage the discussion and re-examination of solutions for classical sensor network problems in the context that deployment is low-cost, on-demand, *and* rapid.

References

[1] In philippine mudslide, sensors pick up tapping, scratching. Dallas Morning News, February 2006.

[2] Maxim A. Batalin and Gaurav S. Sukhatme. Coverage, exploration and deployment by a mobile robot and communication network. In *Information Processing in Sensor Networks: Second International Workshop, IPSN*, 2003.

[3] Jonathan L. Bredin, Erik D. Demaine, MohammadTaghi Hajiaghayi, and Daniela Rus. Deploying sensor networks with guaranteed capacity and fault tolerance. In *The Sixth ACM International Symposium on Mobile Ad Hoc Networking and Computing (MobiHoc'05)*, 2005.

[4] Thomas Clouqueur, Veradej Phipatanasuphorn, Parameswaran Ramanathan, and Kewal K. Saluja. Sensor deployment strategy for detection of targets traversing a region. In *Mobile Networks and Applications*, 2003.

[5] Hui Dai and Richard Han. A node-centric load balancing algorithm for wireless sensor networks. GLOBECOM 2003 - IEEE Global Telecommunications Conference, 2003.

[6] Santpal S. Dhillon and Krishnendu Chakrabarty. Sensor placement for grid coverage under imprecise detections. In *Proceedings of International Conference on Information Fusion*, 2002.

[7] Prabal Dutta, Mike Grimmer, Anish Arora, Steven Bibyk, and David Culer. Design of a wireless sensor network platform for detecting rare, random, and ephemeral events. In *The Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)*, 2005.

[8] Anish Arora et al. Exscal: Elements of an extreme scale wireless sensor network. In *Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*. IEEE, 2005.

[9] Peter Corke et. al. Autonomous deployment and repair of a sensor network using an unmanned aerial vehicle. In *Proceedings of the IEEE 2004 International Conference on Robotics and Automation*, Volume 4, pages 3602–3608. IEEE Computer Society Press, May 2004.

[10] Jie Gao and Li Zhang. Load balanced short path routing in wireless networks. IEEE INFOCOM, 2004.

[11] Y. Gao, K. Wu, and F. Li. Analysis on the redundancy of wireless sensor networks. In *Proc. 2nd ACM Intl. Workshop on Wireless Sensor Networks and Applications (WSNA)*, 2003.

[12] L. Girod, J. Elson, A. Cerpa, T. Stathopoulos, N. Ramanathan, and D. Estrin. Emstar: A software environment for developing and deploying wireless sensor networks. In *Proceedings of the 2004 USENIX Technical Conference*, 2004.

[13] Jonathan W. Hui and David Culler. The dynamic behavior of a data dissemination protocol for network programming at scale. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 81–94. ACM Press, 2004.

[14] Lane Phillips. Aqueduct: Robust and efficient code propagation in heterogeneous wireless sensor networks. Master's thesis, University of Colorado, Boulder, 2005.

[15] Curt Schurgers and Mani B. Srivastava. Energy efficient routing in wireless sensor networks. MILCOM, 2001.

[16] Chieh-Yih Wan, Andrew T. Cambell, Shane B. Eisenman, and Jon Crowcroft. Siphon: Overload traffic management using multi-radio virtual sinks in sensor networks. In *Proceedings of ACM SenSys*, 2005.

[17] Xiaorui Wang, Guoliang Xing, Yuanfang Zhang, Chenyang Lu, Robert Pless, and Christopher Gill. Integrated coverage and connectivity configuration in wireless sensor networks. In *Proceedings of ACM SenSys*, 2003.

[18] Geoff Werner-Allen, Pat Swieskowski, and Matt Welsh. Motelab: A wireless sensor network testbed. IPSN, SPOTS Track, 2005.