On the Need for System-Level Support for Ad hoc and Sensor Networks

Rimon Barr John C. Bicket Daniel S. Dantas
Bowei Du T.W. Danny Kim Bing Zhou Emin Gün Sirer
Department of Computer Science*
Cornell University, Ithaca, NY 14853
magnetos@cs.cornell.edu

Ad hoc and sensor networks are an important, emerging niche that is poorly supported by existing operating systems. In this paper, we argue that network-wide energy management is a primary concern in ad hoc networks, and that this functionality is best provided by a systems layer. We are currently designing and implementing a distributed, power-aware, adaptive operating system, called MagnetOS, specifically targeting ad hoc and sensor networks. MagnetOS provides a single system image of a unified Java virtual machine across the nodes that comprise an ad hoc network. By automatically and transparently partitioning applications into components and dynamically placing these components on nodes within the ad hoc network, our system reduces energy consumption, avoids hotspots and increases system longevity. We show that a systems approach to automatic object placement in an ad hoc network can increase system longevity by a factor of four to five.

1. Motivation

With the recent proliferation of cheap and increasingly powerful mobile devices and sensors, ad hoc networking has emerged as a significant application domain. Ad hoc applications appear naturally in mobile environments and when fixed networking infrastructure is either unavailable or impractical. Examples of such ad hoc applications include large-scale environmental data-collection using sensor networks, coordinated battlefield or disaster-relief operations involving mobile computers, and ubiquitous computing in interactive, smart environments. Despite the importance of these emerging application domains, developing applications for ad hoc and sensor networks remains difficult and poorly supported by existing operating systems.

Two inherent characteristics of the ad hoc computing environment make developing applications for ad hoc networks particularly difficult: ad hoc networks are limited in resources such as battery capacity, and they exhibit frequent and drastic variation in key system metrics, such as bandwidth and connectivity. Form factor limitations in miniaturized devices place tight constraints on the available energy per node [Hill et al. 00]. In addition, the network topology, available power and bandwidth can vary rapidly and through several orders of magnitude [Satyanarayanan 96]. Applications need to adapt, not only to external changes in the resource constrained, frequently varying ad hoc environment, but also to internal changes initiated by the applications themselves. For instance, a sensor application tracking an object that moves over time may need to relocate its event-filtering component closer to the object to reduce network communication. In addition, an application may change its behavior, as in the transition from defensive to offensive mode in a battlefront application, which may modify its communication pattern and necessitate a reorganization of components deployed within the network for optimal performance.

In this paper, we argue that network-wide energy management is best provided by a distributed, power-aware operating system. We describe MagnetOS, an operating system designed specifically to support the adaptation needs of ad hoc applications. The motivation for a new system stems from the lack of support for ad hoc networking applications. Specifically, current state-of-the-art operating systems do not provide the network-wide adaptation mechanisms or policies that applications need in order to address these resource limitations and variations. They force ad hoc applications to treat the network as a "system of systems;" that is, a collection of disparate, autonomous

* Supported in part by ONR Grant N00014-01-1-0968. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of these organizations or the U.S. Government.

computers, because they do not provide a unifying system abstraction. In the absence of a unifying system, applications need to provide their own mechanisms and policies for adaptation separately and independently. Consequently, existing applications rely either on a static assignment of components to nodes or use manual techniques to migrate objects in response to change. Both of these approaches ultimately lead to inefficient energy usage, or premature system failure, or both. Statically assigning node functionality results in non-adaptive, fragile and energy-inefficient systems. The overall application will fail as soon as critical nodes on the dataflow path stall, run out of power or move out of communication range. In turn, locally optimal, manual policies for object mobility pursued by individual applications can lead to globally energy-inefficient behavior when multiple applications interact on the same ad hoc network. For example, local misuse of the battery on a central hub node that is relaying messages may disconnect large parts of the network with adverse global effects.

2. Operating System Support for Ad hoc Networks

We are currently designing and implementing MagnetOS, a distributed, adaptive, power-aware operating system for ad hoc networks. Our goal is to provide a unifying, stand-alone system for ad hoc networks with the following properties:

- **Adaptive:** The system should adapt to the resource constraints and changes in the underlying network in a rapid, yet temporally stable manner. In particular, it should adapt to changes in the network topology, application behavior, and available resources such as power on each node.
- **Efficient:** Policies and mechanisms used for adaptation in the systems layer should yield good power utilization and increase system longevity, without excessive communication or power consumption.
- **General-purpose:** The system should execute applications over networks of nodes with heterogeneous resources and capabilities, as well as over both ad hoc and fixed nodes. Changes in underlying hardware or software choices; network infrastructure, protocols or topology and physical environmental characteristics should not require programmer involvement.
- **Scalable:** The system and its underlying algorithms for optimization and adaptation should work both on a single node and across a large ad hoc network.

MagnetOS achieves these goals by providing a single system image (SSI). The abstraction we provide to applications is that the entire network is a single unified Java virtual machine[1]. The system is comprised of a static and a dynamic component, following the Distributed Virtual Machine architecture [Sirer et al. 99]. A static partitioning service partitions regular Java applications, intended for a single JVM, into objects that can be distributed across the network. It rewrites the application at the byte-code level and injects the necessary instructions to retain original application semantics, even though the rewritten applications are distributed across many nodes. A dynamic runtime component on each node then provides services for application monitoring and object creation, invocation and migration.

The single-system image abstraction is well suited for ad hoc networks because it provides two advantages. First, it presents the operating system with great freedom in object placement, which is essential for efficient, adaptive and scalable execution of applications. The high-level SSI interface separates object location from program execution semantics and enables MagnetOS to move objects to the best-suited nodes within the network. For example, MagnetOS can transparently migrate a filter component to the node closest to its data sources to reduce network communication. Second, a single system image operating system greatly simplifies application development. Manual techniques for adapting applications to changes in their environment are hard to develop, error-prone and platform-specific. Each application needs to re-implement the same migration, monitoring and communication mechanisms, correctly, on every hardware platform. In contrast, MagnetOS enables the distributed execution of ordinary monolithic Java applications. In order to ensure that MagnetOS achieves performance that is as good as or better than handwritten code, an auxiliary interface provided by the MagnetOS runtime allows programmers to explicitly direct object placement, overriding the automatic object placement decisions.

3. Automatic Object Migration and System Longevity

The core of a transparent network-wide operating system consists of algorithms for deciding when and where to move application components. MagnetOS uses two practical, online, power-aware algorithms, called *NetPull* and *NetCenter*, to reduce application energy consumption and increase system longevity. These

[1] Consequently, application components correspond to Java objects in our system, and we use the two terms interchangeably throughout this paper.

algorithms share the same basic approach: they shorten the mean path length of data packets sent between components of an application by automatically moving communicating objects onto nodes that are topologically closer together (Figure 1). The process works by profiling the communication pattern of applications in discrete, asynchronous epochs. Each node decides locally and independently where to move the application components at the end of every epoch. *NetPull* profiles communication at the physical link level, and migrates components over physical links one hop at a time in the direction of greatest communication. *NetCenter* operates at the network level, and migrates components multiple hops at a time directly to the host with which a given object communicates most. *NetPull* and *NetCenter* require knowledge regarding the last hop and sender of each packet, respectively.

Figure 2 shows that automatic and transparent object placement techniques improve system longevity by reducing energy utilization. In this simulation, we examine the behavior of *NetPull* and *NetCenter* in an ad hoc network consisting of 3600 sensors randomly scattered over a 300-by-300 m^2 field, using a fast, packet-level, large-scale ad hoc network simulator we developed. The graph shows the number of nodes that run out of power as a function of time, under four different migration policies. *Static* corresponds to a static assignment of objects to nodes, characteristic of non-adaptive applications. *Random* is an adaptive algorithm that balances the load on the network and avoids early failures due to communication hotspots by moving components to randomly chosen nodes at each epoch. *NetPull* and *NetCenter* perform the automatic object migration scheme outlined above. Figure 2 shows that *NetPull* and *NetCenter* improve system longevity four-fold compared *Random* and five-fold compared to *Static*. The last data points on the curves indicate when the ad hoc sensor network application fails under each migration policy. Failure is defined to occur when more than 50% of the area on the field can no longer be sensed due to lost network connectivity. Under the *Static* placement strategy, the failure of a few critical nodes results in system failure, as *Static* does not migrate components in response to loss. *Random* fares better, because it avoids hotspots and effectively distributes the network load. *NetPull* and *NetCenter* significantly outperform both, because they take application communication behavior and network resource constraints into account [Sirer at al. 01]. Power-aware adaptation through automatic, transparent object placement can lead to significant gains in system longevity.

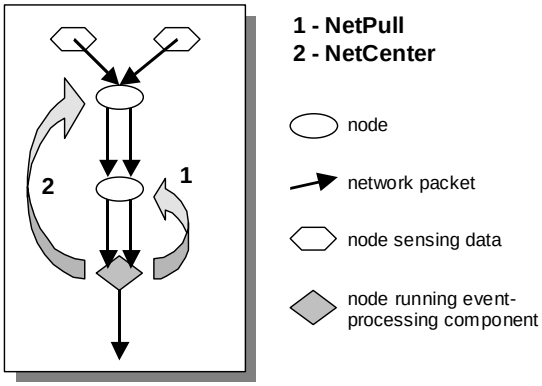

Figure 1: Automatically migrating components closer to their data sources in a sensor network increases system longevity and decreases power consumption by reducing total network communication cost. NetPull (1) moves one hop in the direction of greatest communication, whereas NetCenter (2) moves directly to the host sending the most packets.

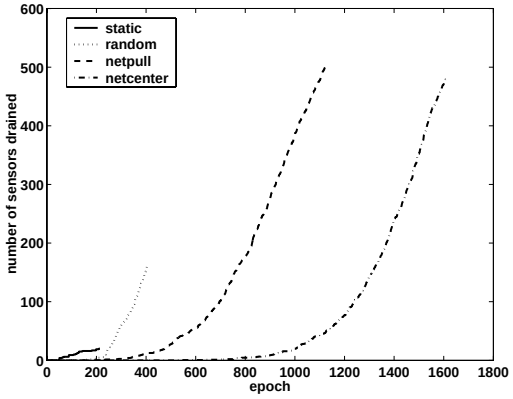

Figure 2: *NetCenter* and *NetPull* migration policies improve system longevity, energy utilization, network connectivity and scalability.

4. Status

The current MagnetOS system provides the image of a Java 1.3 virtual machine. An application partitioning service, implemented using the Kimera rewriter [Sirer et al. 99], modifies object creation, invocation and field access instructions to operate over an ad hoc network. We rely on Java RMI mechanisms for invocation, while an in-kernel AODV protocol [Perkins 97] provides packet forwarding in Linux, a user-level protocol implementation provides the same functionality on Windows. In addition, we are developing our own native JVM for resource-limited devices.

5. Related work

MagnetOS builds on recent research on ad hoc routing protocols, node-based power conservation, distributed object systems and mobile computing.

Prior research on ad hoc networks has focused on multi-hop routing algorithms, such as DSDV, DSR, AODV, ZRP, TORA, CEDAR and many others ([Perkins & Bhagwat 94, Broch et al. 96, Perkins 97, Haas & Pearlman 98, Park & Corson 98, Sivakumar et al. 99], see [Royer & Toh 99] for a survey). These algorithms move data from a source to given destinations as efficiently as possible in the presence of underlying network topology changes. They focus on optimizing the route for metrics such as latency, hop count or power, while leaving the source and destination endpoints fixed. MagnetOS complements this work, and extracts additional and significant power gains by moving the communication endpoints.

Prior work has also examined how to minimize power consumption within an independent host through various mechanisms [Grunwald et al. 00, Weiser et al. 94, Farkas et al. 00, Douglis et al. 95, Helmbold et al. 96, Stemm & Katz 96], including low-power processor modes, disk spin-down policies, adapting wireless transmission strength and selectively turning off unused devices. These mechanisms for saving node power are complementary to our network-wide adaptation. Like [Vahdat et al. 00], we treat energy as a first-class system resource, and extend its management across an entire ad hoc network.

Mobile computing projects have examined mobility toolkits, frameworks and code libraries that facilitate construction of mobile applications. They provide services for disconnected and mobile operation [Joseph et al. 95], quality of service tradeoffs [Campbell 98], synchronization and data management [Mascolo 01]. MagnetOS shares the goals of these projects, but differs from them by offering a complete operating system with a unifying system abstraction. Systems such as Emerald [Jul et al. 88], Thor [Liskov et al. 96] and Globe [van Steen et al. 99], among others, have pioneered efficient and practical mechanisms for object distribution. Our work on object distribution extends this work with mechanisms and policies for power-aware adaptation, targeted for the ad hoc network model.

Finally, directed diffusion [Heidemann et al. 01] proposes a framework for application-specific labeling of data, gradient-based routing and filtering packets in the network using stateless components specified in a constrained language. MagnetOS extends this work by supporting components that carry state, and enabling components to be written in a general-purpose language. Consequently, MagnetOS can perform state-requiring operations, such as aggregation, in the network.

6. Summary

In this paper we advocate for operating system support of ad hoc and sensor networks. We contend that only a network-wide, power-aware operating system can efficiently and fairly manage the limited and time-varying resources inherent in ad hoc application environments. We describe MagnetOS, an ad hoc operating system that provides a single system abstraction of an ad hoc network to applications. MagnetOS works by automatically partitioning applications into networked components and transparently migrating those components to the best-suited nodes in the network. We have shown that automatic object placement policies, such as *NetPull* and *NetCenter*, can reduce energy consumption and increase system longevity. Our large-scale simulation results demonstrate that the wins can be as much as a factor of four to five compared to energy-oblivious techniques for adaptation. Compared to the "system of systems" model that characterizes the current state of the art, a single system image provides the necessary unifying abstraction for adaptive, power-efficient, platform-independent applications.

References

[Broch et al. 96] J. Broch, D. B. Johnson, and D. A. Maltz, The Dynamic Source Routing Protocol for Mobile Ad hoc Networks. *Mobile Computing*, 1996.

[Campbell 98] O. Angin, A. T. Campbell, M. E. Kounavis and R. R.-F. Liao. The Mobiware Toolkit: Programmable Support for Adaptive Mobile Networking. *IEEE Personal Commununication Magazine*, 1998.

[Douglis et al. 95] Fred Douglis, P. Krishnan and Brian Bershad. Adaptive Disk Spin-down Policies for Mobile Computers. In *2nd USENIX Symposium on Mobile and Location-Independent Computing*, April 1995.

[Ellis 99] Carla S. Ellis. The Case for Higher-Level Power Management. In *Proceedings of the 7th Workshop on Hot Topics in Oper-ating Systems*, Rio Rico, AZ, March 1999.

[Farkas et al. 00] Keith I. Farkas, Jason Flinn, Godmar Back, Dirk Grunwald and Jennifer-Ann M. Anderson. Quantifying the energy consumption of a pocket computer and a Java virtual machine. In *Measurement and Modeling of Computer Systems*, pp 252-263, 2000.

[Grunwald et al. 00] D. Grunwald, P. Levis, K. Farkas, C. Morrey and M. Neufeld. Policies for dynamic clock scheduling. In *Proceedings of 4th OSDI*, San Diego, CA, October 2000.

[Haas & Pearlman 98] Z. J. Haas and M. R. Pearlman, The zone routing protocol (ZRP) for ad hoc networks (Internet-Draft). Mobile Ad hoc Network (MANET) Working Group, IETF, Aug. 1998.

[Heidemann et al. 01] John Heidemann, Fabio Silva, Chalmarek Intanagonwiwat, Ramesh Govindan, Deborah Estrin and Deepak Ganesan. Building efficient wireless sensor networks with low-level naming. In *Proceedings of the 18th Symposium on Operating Systems Principles*, Lake Louise, Alberta, October 2001.

[Helmbold et al. 96] D. Helmbold, D. Long and B. Sherrod. A Dynamic Disk Spin-Down Technique for Mobile Computing. In *Proceedings of the ACM International Conference on Mobile Computing*, 130-142, Nov. 1996.

[Hill et al. 00] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, Kristofer Pister. System architecture directions for network sensors. In *Proceedings of the Conference of Architecture Support for Programming Languages and Operating Systems*, 2000.

[Joseph et al. 95] Anthony D. Joseph, Alan F. De Lespinasse, Joshua A. Tauber, David K. Gifford, and M. Frans Kaashoek., Rover: A Toolkit for Mobile Information Access. In *Proceedings of the Fifteenth Symposium on Operating System Principles*, December 1995.

[Jul et al. 88] Eric Jul, Henry Levy, Norman Hutchinson, Andrew Black. Fine-Grained Mobility in the Emerald System. *ACM Transactions on Computer Systems*, 6(1), Feb. 1988, 109-133.

[Liskov et al. 96] B. Liskov, A. Adya, M. Castro, M. Day, R. Gruber, U. Maheshwari, A. Myers, L. Shrira. Safe and Efficient Sharing of Persistent Objects in Thor. In *Proceedings of SIGMOD*, Montreal, Canada, June 1996.

[Mascolo 01] Cecilia Mascolo, Licia Capra and Wolfgang Emmerich. XMIDDLE - A Middleware of Ad hoc Networks. *UCL-CS Research Note 00/54*, 2001.

[Park & Corson 98] Vincent D. Park and M. Scott Corson. Temporally-Ordered Routing Algorithm (TORA) version 1: Functional Specification. Internet-Draft, draft-ietf-manet-tora-spec01. txt, August 1998.

[Perkins 97] Perkins, C.E. Ad hoc On-Demand Distance Vector (AODV) Routing. IETF Internet Draft, Dec.1997.

[Perkins & Bhagwat 94] C. Perkins and P. Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *Proceedings of the ACM SIGCOMM*, October 1994.

[Royer & Toh 99] Elizabeth Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications Magazine*, April 1999, 46-55.

[Satyanarayanan 96] M. Satyanarayanan. Fundamental Challenges in Mobile Computing. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*. Philadelphia, PA, May 1996.

[Sivakumar et. al. 99] Raghupathy Sivakumar, Prasun Sinha and Vaduvur Bharghavan. CEDAR: a core-extraction distributed ad hoc routing algorithm. *IEEE Journal on Selected Areas in Communication*, 17(8), August 1999.

[Sirer et al. 99] Emin Gün Sirer, Robert Grimm, Arthur J. Gregory and Brian N. Bershad. Design and Implementation of a Distributed Virtual Machine for Networked Computers. In *Proceedings of the Symposium on Operating Systems Principles*, Kiawah Island, South Carolina, pp. 202-216, December 1999.

[Sirer at al. 01] Emin Gün Sirer, Rimon Barr, T.W. Danny Kim, Ian Yeen Yan Fung. Automatic Code Placement Alternatives for Ad hoc and Sensor Networks. *Computer Science Technical Report 2001-1853*, Cornell University, October 2001.

[van Steen et al. 99] M. van Steen, P. Homburg and A.S. Tanenbaum. Globe: A Wide-Area Distributed System. *IEEE Concurrency*, January-March 1999, pp. 70-78.

[Stemm & Katz 96] Mark Stemm and Randy Katz. Measuring and Reducing energy consumption of network interfaces in hand-held devices. In *Proceedings of 3rd International Workshop on Mobile Multimedia Communications*, Sept. 1996.

[Vahdat et al. 00] Amin Vahdat, Alvin Lebeck and Carla S. Ellis. Every Joule is Precious: The Case for Revisiting Operating System Design for Energy Efficiency. *9th ACM SIGOPS European Workshop*, September 2000.

[Weiser et al. 94] Mark Weiser, Brent Welch, Alan Demers and Scott Shenker. Scheduling for Reduced CPU Energy. In *Proceedings of First Symposium on Operating Systems Design and Implementation*, Monterey CA, Nov. 94.