

MagnetOS: A Single System Image Operating System for Ad Hoc and Sensor Networks

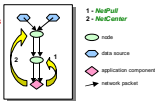
Rimon Barr, Ian Fung, T. W. Danny Kim, Emin Gün Sirer
 {barr, ifung, twkim, egs}@cs.cornell.edu
 Cornell University, SOSP 2001

Programming on ad hoc networks is difficult

- Ad hoc and sensor networks are an emerging, important niche. Ad hoc applications arise naturally. Some futuristic examples include an environmental data collection application over a sensor network, applications for coordinated battlefield or disaster relief operations, and highway traffic control.
- Developing adaptive applications for ad hoc and sensor networks poses significant challenges. Nontrivial applications entail collaboration between components distributed throughout an ad hoc network. Defining these components, optimally placing them on nodes in the ad hoc network and relocating them in response to changes in power, bandwidth and network load is a fundamental problem.
- Current state-of-the-art relies on manual application partitioning. Manual approaches to code and data migration are platform-dependent, error-prone and hard to get right. Locally optimal decisions made by applications that share the same network can lead to globally unstable and energy inefficient behavior.
- The problem is the lack of a unifying system layer. Ad hoc networks appear to applications as a system of independent, autonomous systems. No support from the OS exists to manage resources, seek out the global optimum, or provide mechanisms for code and data migration.

Automatic object placement

- MagnetOS introduces *NetPull* and *NetCenter*, practical, power-aware, online algorithms for automatic object placement within an ad hoc network. These core algorithms can increase system longevity by a factor of four to five over static placement techniques.
- 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* shorten the mean path length of data packets by moving communicating objects closer together.
- They profile the communication pattern of applications in discrete time units and decide, based on local information, where to move the objects. Moving application components from node to node avoids communication hot spots.



MagnetOS: an ad hoc operating system

MagnetOS is an operating system for ad hoc networks, designed to optimally allocate and use the resources available in an ad hoc network of independent nodes. It consists of a thin OS layer that provides a single Java virtual machine image of the network to applications. MagnetOS transparently manages the placement of application components to ensure fair, consistent and globally optimal allocation of resources.

MagnetOS is:

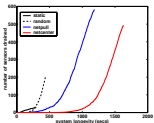
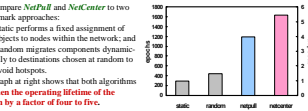
- Efficient.** Policies and mechanisms used for adaptation in the systems layer do not require excessive communication or power consumption. The default policies and mechanisms yield good power utilization and maximize total system lifetime.
- Adaptive.** Responds to changes in network environment, communication pattern of the applications, and the availability of resources in the network.
- General purpose.** Porting an existing monolithic application to execute efficiently on an ad hoc network requires little effort.
- Platform independent.** Applications can be executed on ad hoc networks of heterogeneous nodes, as well as over both ad hoc and fixed networks.

Improved system longevity, energy utilization

We compare *NetPull* and *NetCenter* to two benchmark approaches:

- Static performs a fixed assignment of objects to nodes within the network; and
- Random migrates components dynamically to destinations chosen at random to avoid hotspots.

The graph at right shows that both algorithms lengthen the operating lifetime of the system by a factor of four to five.

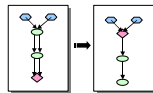


The graph of sensor drainage over time (above) demonstrates that both *NetPull* and *NetCenter* improve the distribution of communication load, and thereby use more of the available field energy. This is because active migration algorithms avoid creating hotspots around critical nodes. Actively migrating components in the network drains fewer nodes than a static placement.

MagnetOS runs ad hoc applications

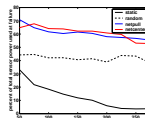
MagnetOS:

- provides a single system image of a unified Java virtual machine to applications over an ad hoc collection of heterogeneous nodes,
- automatically and transparently partitions applications into components,
- dynamically finds a placement of these components on nodes within the ad hoc network
- that reduces energy consumption and increases system longevity.



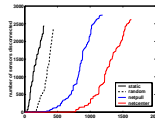
Migrating components closer to their data sources in a sensor network increases system longevity and decreases power consumption by reducing total network communication cost.

Improved connectivity, scalability

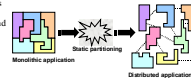


Static placement does not scale, because the number of critical nodes is not proportional to the size of the field. *NetPull* and *NetCenter* scale to large networks, and are unaffected by variations of field size (left). They utilize more of the available energy on the field prior to failure.

The graph of disconnected nodes over time (right) shows that the number of disconnected nodes increases more gradually for *NetPull* and *NetCenter* because they distribute load more evenly across the network. In the case of Static and Random, even though only a small number of nodes are drained, they are all located around the home node and thus quickly disconnect the entire field from the wired network.



Application partitioning



- A static MagnetOS service partitions monolithic Java applications into components that can be migrated and executed over an ad hoc network.
- The division is performed statically along object interfaces at class granularity, which preserves class interfaces, retains type safety, and works at the byte-code level without requiring source access.
- Each node in the network runs a local MagnetOS runtime, which provides services for object creation, invocation, affinity and migration.
- Instructions for object creation are replaced by calls to the local runtime, responsible for selecting an appropriate node and creating a new instance of the object on that node.
- Instructions for object invocation are modified to go through an RPC mechanism to invoke remote objects. MagnetOS uses the Java RMI interface for remote object invocation.
- This automatic partitioning technique provides a convenient, default mechanism for transitioning legacy, monolithic applications to execute over ad hoc networks.

Summary

- Ad hoc and sensor networks are an emerging, important niche. Developing ad hoc applications is difficult due to the lack of a unifying system layer.
- MagnetOS is a distributed operating system for ad hoc networks that provides a single system image, to execute applications in an efficient, adaptive, general purpose and platform independent manner.
- MagnetOS automatically partitions applications and distributes the components in the network to reduce energy consumption and increase system longevity.
- NetPull* and *NetCenter* are online, power-aware algorithms for component distribution that provide improved system longevity, energy utilization, network connectivity and scalability.
- These algorithms provide a factor of four to five improvement in system longevity.