

EXPLOITING THE INTERACTIONS BETWEEN ROBOTIC AUTONOMY AND NETWORKS

Jason Redi and Joshua Bers

*BBN Technologies**

10 Moulton Street, Cambridge, MA, 02138

{redi, jbers}@bbn.com

Abstract For teams of autonomous robots to fully utilize their distributed problem solving and collaboration capabilities, they require a self-organizing, self-healing, multi-hop communication network. Robot teams compose a unique kind of multi-hop network because the adaptive networking algorithms reside in the same architecture as the algorithms for performing motion and mission taskings. Providing timely access to current and predictive information across these subsystems can impart a new “sensory” perception to the robot as well as allow the ad hoc network to more efficiently and effectively perform its role. We present an operational architecture which allows blackboard-style sharing of information between autonomous robot controllers and ad hoc networking protocols.

Keywords: Ad hoc network, architecture, robotics

1. Introduction

An ad hoc network is a (possibly mobile) collection of communications devices (nodes) that wish to communicate, but have no fixed infrastructure available, and have no pre-determined organization of available links [7]. Individual nodes are responsible for dynamically discovering which other nodes they can directly communicate with. A key assumption is that not all nodes can directly communicate with each other, so nodes are required to relay packets on behalf of other nodes in order to deliver data across the network. A significant feature of ad hoc networks is that

*This work was sponsored by DARPA/IPTO under contract number DASG60-02-C-0060. Content of this work does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

rapid changes in connectivity and link characteristics are introduced due to node mobility and power control practices.

A particular advantage of *robots* over other devices that use ad hoc networking protocols is that the algorithms for networking protocols typically reside in the same architecture as the algorithms for performing motion control and overall mission tasks. Robots can therefore easily and quickly exchange information between the robotic autonomy and networking subsystems in order to enhance and improve the ability of the robotic team to perform a task. Although the networking communication layer has traditionally been perceived as a black-box service, we find that providing timely access to the information kept by the ad hoc routing protocols imparts a new “sensory” perception to the robot. Likewise, the ad hoc network benefits from access to information about the environment or future robot motion, since it can then adapt to and optimize for current and future operating conditions. In this paper we introduce our ad hoc networking protocols and then describe our framework for exploiting robotic mission network interactions (ERNI).

2. Related Work

The utility of communications for robot teams has been well established in the literature: [3] was one of the first works to attempt to evaluate the utility of communications capabilities on particular tasks. Robot teams with ad hoc networks have already been deployed such as the ones described in [8], [9] and [6]. [10] showed that a simple store-and-forward wireless network can help a team of robots to collect sensor information faster than allowed by a star-topology. Unfortunately, none of these systems used the feedback between the local networking algorithms and the autonomous behaviors, to improve performance.

3. Approach

The traditional view of the layered network architecture is advantageous for complex systems that need to abstract services in order to simplify designs, as well as allow for future expandability. However, it requires limiting the amount of information that is exchanged between the layers. The guiding principle, to allow only the least common denominator of information that will be pertinent to all types of lower layers, is too restrictive for our purposes.

A potentially more useful way of designing network services is that of a peer process that provides both *basic* and *extended* functionalities. The peer can receive and transmit packets on other module’s behalf, but also can provide extended information or asynchronous alerts to particular

situations. Figure 1 depicts our model for network relationship with the other modules within a robot.

The basic functionality in our system is enabled by the use of the standard Internet Protocol (IP) stack. This allows us to use IP as an “interface” for packet transmission and reception. Application (controller) writers are free to use sockets, datagram manipulation calls, and other standard calls that are used when writing any traditional Internet enabled application.

The extended functionality is enabled by the use of the Simple Networking Management Protocol (SNMP) [5], which is an IETF standard for querying or manipulating the variables in a network device. It is significant to note that the traditional way of using SNMP is to query remote devices of interest from a central location. In a robot team where each autonomous robot was interested in status and statistics of the other robots in the team, this would introduce significant processing and packet overhead since the network would be maintaining $N(N - 1)$ SNMP connections over the wireless links. Instead we expect our ad hoc enabled robots to query only their own *local* networking protocol modules. The networking protocols typically keep large amounts of useful information around for the purpose of performing their tasks, and this information is already efficiently shared across the network by the protocols. Therefore the typical network overhead of SNMP does not affect us and we can allow the networking protocols to distribute the information in the way that is most efficient for the system.

Note that SNMP only provides a standardized method to access to the extended functionality of a locally operating ad hoc network. It does not provide a simple way for multiple modules, including the network, to work together within a single architecture. For that functionality we have created the ERNI (Exploiting Robotic Mission Networking Interactions) architecture, which is described following some details regarding the networking protocols themselves.

3.1 Network protocols

Existing and ad hoc networking protocols fall into one of two groups – *on-demand* or *proactive*. The difference between these protocols lies in when and how the network determines what links are available in the network. Pro-active networks are regularly trying to determine what links (e.g. neighbor adjacencies) are available in the network and therefore what are the sets of paths that could be used to carry traffic. A pro-active network can provide a view of the connectivity and can make routing decisions based on near global knowledge. Alternatively, “on-

demand” routing protocols only attempt to set up routes for a new traffic flow when the traffic first arrives. Although this minimizes control overhead when flows and topology are regular and stable, the cost is a significant delay for establishing a new route, suboptimal routes after movement or mid-hop link breakage, and a lack of complete knowledge of potential alternative paths. Pro-active routing is far more appropriate for robotic systems, since the information that the network actively gathers will then be available to other robotic modules such as mission and tasking control.

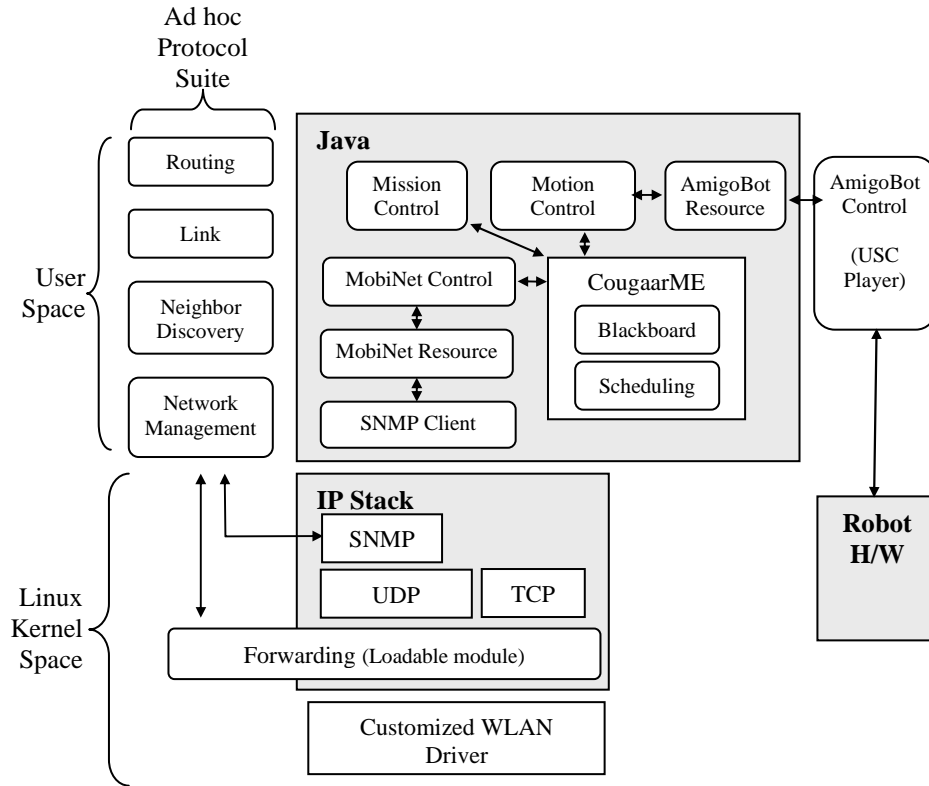


Figure 1. ERNI Architecture

3.1.1 System and Information Available. Our networking protocol is divided into multiple interacting modules which each has a unique responsibility. The division of labor across these modules creates a modularized software design that allows us to experiment with different algorithms inside each of these modules independently.

- **Neighbor Discovery** – Tries to discover new one-hop neighbors. Has the responsibility for declaring a link to a neighbor as being “up” or “down”. Discovery is typically done by sending out periodic or semi-periodic beacons, where reception of beacons may indicate the existence of a new node to his neighbors and missing beacons can indicate the loss of a link to a neighbor.
- **Link Characterization** – This module interfaces with the radio hardware and attempts to determine the long-term metrics of a link (power required, data rate, etc), from promiscuous and directed transmissions and receptions. It provides these metrics to other modules to help them gauge the costs of a link.
- **Routing** – This module is responsible for receiving, processing, and storing the link-state information received from other nodes and issuing link-state updates when significant changes occur. It also executes routing algorithms on the link-state data to determine paths to any possible destination. The routing tables generated as a result of this process are passed to the forwarding module.
- **Forwarding** – This module is responsible for processing all incoming and outgoing packets. For incoming packets this involves determining the local module that is the intended recipient of the packet, or the next-hop that the packet should be forwarded to. For outgoing packets this involves determining the one-hop neighbor that is the next hop along the path to the packet’s destination, which is performed by referring to the table created by the routing module.

3.1.2 Information exchange. In Table 1 we list a few selected pieces of information that are available from our networking protocols. The challenge from the networking perspective is to provide these in an accurate and timely way. The challenge from the robotics perspective is to use these in a way that enhances the team’s mission.

Note that the topology table effectively provides of a directed graph where the vertices are all the reachable robots in the network, and each edge contains a vector of weights corresponding to the costs associated with transmitting between two nodes. Perhaps the most important cost is that of required energy to close a link, since it defines effective bit error rate, which can suggest the estimated number of retransmissions and therefore delay or quality of bandwidth on the link.

We are currently in the process of extending our networking software to also use information provided by the robotic autonomous mission con-

Table 1. Example information available from the ad hoc network.

Data Item	What it means	How it can help
Next-hop Table	Location and existence of one-hop neighbors	Who can help me with local tasks?
Topology Table	Existence of all nodes in the partition and the energy, delay and bandwidth between them	Should we move to enhance comms?
Path Characteristics Table	Cost in energy, delay and bandwidth for the entire path through the network.	Can the comms support the mission?

Table 2. Information available from robotic mission tasking modules that the network can use.

Data Item	What it means	How it can help
Network Mode	Should the network try to optimize latency, reliability, or energy?	Extend understanding of the mission goals to networking
Environment	What are the physical characteristics? (forest, etc.)	Network can tune itself to location
Current and future motion	Where are we going to go and how fast?	Proactively modify the topology
Future traffic	Who is expected to be communicating with whom	Ensure bandwidth is available by policing traffic or modifying routes.

trol modules. Table 2 provides some examples of that information. As in any complex software system, ad hoc networks contain a large number of variables that can be used to control the reactivity and efficiency of the network. For example, we can limit the number of neighbors that we would allow, or perhaps change the periodicity of heartbeats used for detecting new neighbors. At the simplest level we can provide the functionality described in Table 2 simply by selecting an appropriate pre-set family of parameters for particular situations. However, we are currently experimenting with adaptive and learning approaches to respond to this information.

3.1.3 Implementation and Testbed. We use the Linux operating system on our robots and insert the forwarding module at the sub-IP level. The other protocols are implemented in application space since they do not have time critical constraints. To the IP protocols and applications that use it, the forwarding module appears to be just another Ethernet-like interface. The forwarding module uses the ToS bits in the packet along with the final destination IP address to determine the appropriate next-hop and radio parameters for getting the packet to the next hop with the required quality of service.

It should be noted that this architecture is what is called a “3a” subnetwork. That is, we do not use IP forwarding or routing services at all. The whole multi-hop network appears to be a single one-hop LAN from the perspective of IP. The advantage of this method is that the system can easily be adapted to a new version of IP, or not use IP at all if the system is severely size constrained. Additionally, it allows the forwarding module to directly interface with the lower layer radio driver and hides the rapidly changing topology from any attached IP devices (such as routers) that might be using the network. One way to think of this is much like running IP over a particular technology (i.e. IP over ATM, IP over Ethernet, IP over ad hoc).

Our robotic testbed consists of 8 ActivMedia Amigobots outfitted with a Nano-engine from Bright Star Engineering, and a custom radio interface board designed by Navy SPAWAR and BBN. The Nano-engine is a 2 inch by 1 inch board that contains a StrongARM processor, 4 Mb of flash RAM and 32 Mb of RAM. The Nano-engine runs a compressed version of Linux and our networking protocols. The radio interface board allows the Nano-engine to be connected to a PCMCIA interface where we use standard 2.4 Ghz Cisco or Agere/Orinoco WLAN cards for most experiments.

3.2 ERNI – Exploiting Robotic Mission Networking Interactions

We have designed and implemented a collaborative architecture that is built on top of an existing publish-and-subscribe, multi-threaded framework. Our design goal was to provide the robot mission control modules with loosely coupled access to the ad hoc networking sub-system. We use the Cognitive Agent Architecture, Cougaar, to provide such a framework [1].

3.2.1 Cougaar. Cougaar (cognitive agent architecture) is a public domain framework for developing multi-agent distributed applications

in Java. It was originally developed under DARPA’s Advanced Logistics Program (ALP) with the target domain being logistics and planning. However Cougaar has also been used for projects ranging from peer-to-peer resource management, data mining, and distributed sensors. There are two versions of Cougaar, standard edition (SE) targeted at workstations and micro edition (ME) targeted at embedded devices. While Java affords portability and ubiquity on many platforms, Cougaar’s shared blackboard model enables loosely coupled communication between sub-systems.

A group of interacting Cougaar modules operating together is called an Agent. A group of collaborating Agents is called a Society. Typically there would be a single Agent operating in a robot.

The modules within an agent communicate with one another by publishing and subscribing to objects on a shared blackboard. Each agent owns its own blackboard and its contents are visible only to that agent. All sharing of blackboard state across Agents is done by explicit push-and-pull of data through inter-agent tasking and querying. In this way, Cougaar is able to maintain fine-grained state in individual agents while sharing only high-level synopsis information around the society, making the management of information scalable and efficient. The communications mechanisms of Cougaar use Java which is layered on top of IP, so all inter-Agent communications can be transported seamlessly over the multi-hop network.

CougaarME provides the core execution model and messaging infrastructure for ERNI. The basic unit of communication is a task. Tasks are published to the blackboard by a requestor (manager). Tasks contain verbs and prepositional phrases that define the language of the task domain. For ERNI, we developed two intra-agent tasks: `BotMotion` and `NetWatch`, and two inter-agent tasks: `Formation` and `Rescue`. Table 3 shows the prepositional phrases for the tasks, their meanings and parameters.

Once a task is published to the Cougaar blackboard, other modules (also called “plugins”) are activated based on their subscriptions. In a typical logistics applications, there are expander plugins that break complex tasks into smaller sub-tasks, allocator plugins that assign a task to a resource and assessor plugins that monitor the quality of the execution or plan for a task. In the robot control domain a new type of plugin is needed to control real-time resources such as the robot’s motors, the network and other sensing devices. These control plugins subscribe directly to tasks: `NetInfo` gives access to the network resource; `BotMotion` provides an interface to the motion control resource.

Table 3. The Cougaar tasks used in ERNI and their prepositions. Some of the prepositions take parameters, e.g., path cost takes source node, destination node, and type of service.

Verb	NetInfo	BotMotion	Formation	Rescue
<i>Semantics</i>	Subscribe to information about the network	Control the robots motion	Task a robot team	Solicit help from a peer
<i>Prepositions</i>	Node position, network topology, path cost, link quality, neighbor list	Randomwalk, goto, stop, turn, velocity	Disperse, contract	Location
<i>Scope</i>	Local (intra-robot)	Local (intra-robot)	Team (inter-robot)	Team (inter-robot)

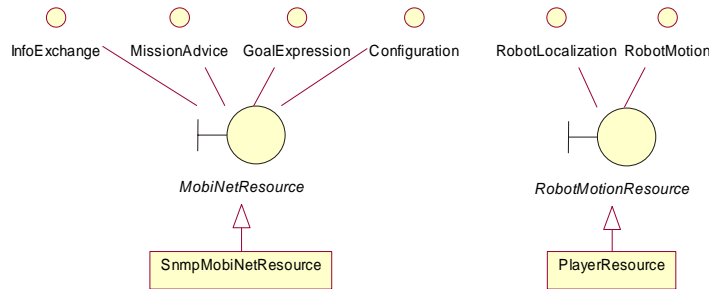


Figure 2. Class diagram shows (top to bottom) interfaces, abstract resources and device specific resources used by ERNI.

Using object-oriented design, we defined the interfaces to both the multi-hop network and for the robot motion control. Then we developed abstract Cougaar resources that implement these interfaces, and finally we constructed device specific subclasses of these resources. This three tiered design enables device independence and the ability to swap in different resources without changing the rest of the mission control subsystem. Currently we have implemented a robot control resource that connects to the USC Player server [4]. Our current mobile network resource is an SNMP-based client to our ad hoc networking protocols. Figure 3 shows the class diagrams of our three tiered design.

3.2.2 Intra-robot (local) communication. The data flow in the ERNI architecture is illustrated in the sequence diagram shown in Figure 3. It is useful to note the procedure call isolation between the two plugins in the sequence diagrams. This is a result of the Cougaar framework.

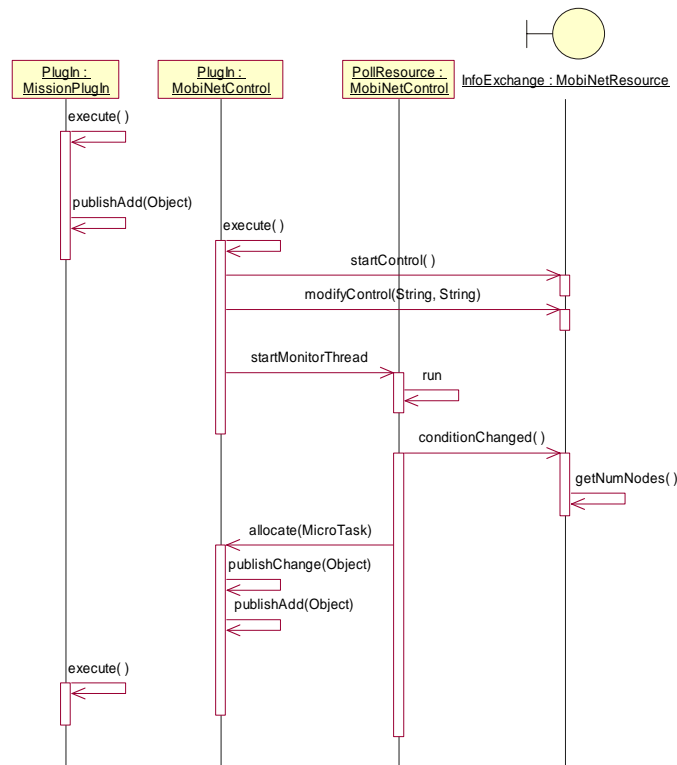


Figure 3. Sequence diagram (time flows from top to bottom) for processing a `NetInfo` task. Mission control plugin issues a `NetWatch` task with the `NumNodes` prepositional phrase. `MobiNetControl` plugin is woken up (based on its subscription to `NetWatch` tasks). `MobiNetControl` finds a `MobiNetResource` and tells it that someone wants to monitor the number of nodes in the network. `MobiNetControl` then creates a thread that will poll the resource periodically to see if its condition has changed. The `MobiNetResource` will, in turn, query the underlying network management module (in our case via SNMP). When a change is detected the control plugin will publish the change to the blackboard. As the originator of the task, `MissionControl`, is woken up as well as any other subscribers to `NumNodes` objects.

3.2.3 Inter-robot (team) communication. Inter-node communication is similar to intra-node communications. Tasks are created

locally and published to the blackboard, however, then they are allocated to a foreign CougaarME node or *MicroAgent*. The CougaarME messaging system handles the transport (using TCP/IP) of the task to the remote MicroAgent. The Formation and Rescue tasks from Table 3 are examples of inter-robot tasks.

One of the challenge of inter-node messaging is discovery and addressing of peer MicroAgents. One needs to know who your neighbors are and what capabilities they have if you want to task them directly. For looser control one could use broadcast. Peer-to-peer frameworks for service advertisement and discovery, such as JXTA [2] provides a possible solution. One might enhance this framework by using knowledge of the ad hoc network via our *MobiNet* resource to assist in peer/neighbor discovery. In this approach some kind of name service would run locally on each node constantly monitoring the network state to check for new members. It could then send queries to the members to discover what type of services they support. The benefit of this approach is that unlike JXTA alone, we don't duplicate the efforts of the network layer in discovering neighbors and nodes in the active network. Instead we are activated by changes in the state of the network, saving valuable bandwidth over wireless ad hoc links.

Typically the semantics of the inter-node comms are at a higher, more abstract level than their intra-node counterparts. This hierarchical approach is scalable as the detailed reporting is kept within a single node, and thus kept off the wireless network. For example a centralized mission control node might issue a high-level Formation task with preposition "disperse" to a team of robots. Each robot's Mission plugin would receive this task (assuming that they subscribe to the task type) and would translate it into a local *BotMotion* command (or sequence of commands). Depending on robot role, and its pre-designed logic, one robot might interpret it to mean that it should go to a certain location (e.g. a sentry), whereas another might go into a random walk state.

4. Summary

Multi-hop network communication for teams of robots is an important asset for robots which may have complex group taskings or data distribution requirements. These networks have the potential to provide significant amounts of detailed information on their operation as well as adapt to specific information provided on the goals or expectations of the applications. In this paper we have outlined an architecture called ERNI (Exploiting Robotic Mission Networking Interactions) which allows the networking protocols to interact and operate with the robotic

tasking algorithms in a fluid and seamless way. We have implemented this architecture in an embedded and portable system, and provided enough generality to the interfaces and operation that it will be useful for a wide variety of robot controller and tasking designers.

References

- [1] www.cougaar.org.
- [2] www.jxta.org.
- [3] Balch, T., and Arkin, R., "Communication in reactive multiagent robotic systems", *Autonomous Robots*, 1(1): 27-52, 1995.
- [4] Gerkey, B., Vaughan, R., Sty, K., Howard, A., Sukhatme, G., and Mataric, M., "Most Valuable Player: A Robot Device Server for Distributed Control," Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2001), pp 1226-1231, 2001.
- [5] Harrington, D., Presuhn, R. and Wijnen, B., "An Architecture for Describing SNMP Management Frameworks", RFC 2261, Standards Track, Internet Engineering Task Force, January 1998.
- [6] Nitzel, R., Benton, C., Chappell, S., Blidberg, D., "Exploiting dynamic source routing to enable undersea networking over an ad-hoc topology," Proceedings of the 2002 International Symposium on Underwater Technology, Tokyo, Japan, pp 273-277, 2002.
- [7] Ramanathan, R. and Redi, J., "A Brief Overview of Ad Hoc Networks: Challenges and Directions", IEEE Communications Magazine, May 2002, pp 20-22.
- [8] Sibley, G., Rahimi, M., and Sukhatme, G., "Robomote: a tiny mobile robot platform for large-scale ad-hoc sensor networks," Proceedings of the IEEE International Conference Robotics and Automation 2002 (ICRA '02), Vol. 2, pp 1143-1148, 2002.
- [9] Welsh, B., Redi, J., Pointer, S., Davies, J., "Advances in Efficient Submersible Acoustic Mobile Networks," Proceedings of the International Unmanned Undersea Vehicle Symposium, Newport RI, April 2000.
- [10] Winfield, A., "Distributed sensing and data collection via broken ad hoc wireless connected networks of mobile robots," *Distributed Autonomous Robotic System 4*, Springer-Verlag, pp. 273-282, 2000.