WIND: Wireless Networks of Devices
9807-04

Progress Report: July 1, 1998—December 31, 1998

Hari Balakrishnan and John V. Guttag

Graduate Students: William Adjie-Winoto and Elliot Schwartz

Project Overview

The goal of the WIND project at MIT is to design and implement self-organizing networks with a high degree of decentralization, robustness and distributedness in their operation. The WIND protocols and middleware provide built-in support for user and device mobility, resource discovery, service location, and group communication in a decentralized manner. Our key innovation is the use of an intentional naming architecture where applications describe what they are looking for (i.e., their intent), not where to find it in the network topology (which is how most network naming schemes work today). In this architecture, name resolvers can also route messages to the eventual destinations, leading to an integrated approach to resolution and routing. Thus, WIND is built around the premise that a flexible naming architecture and the ability of the name resolution process to be involved in data routing is a key enabler of large-scale self-organizing device networks.

Our application scenario uses these building blocks to demonstrate a self-organizing wireless network over heterogeneous indoor wireless technologies (in-building RF wireless LAN and Infrared), running location-dependent applications. Here, users can access data and control dynamic, mobile information sources (e.g., mobile cameras, sensor nodes) as well as enable devices to obtain information based on their location (e.g., if you walk into a specific room, the devices, nodes and users in that room automatically become known to your handheld or laptop computer) or other system characteristics. Our vision is to achieve all this with no prior manual configuration.

Progress through December 1998

RadioLAN wireless LAN installation

We have performed preliminary testing and setup of the wireless network using RadioLAN for experimenting with WIND. The wireless testbed allows experiments with various networked, mobile devices. Our performance experiments show median throughputs on the order of 5-7 Mbps.

At this point, only a portion of one floor (the 5th) at LCS is covered with this network, and it is restricted to one subnet. We plan to significantly expand the coverage of the testbed in the near future.

WIND software design and implementation

The WIND software system consists of the following components:

(i) Intentional Naming System (INS)
(ii) Wireless beaconing protocols (for discovery and network/application self-configuration)
(iii) Applications (e.g., mobile camera application)

Intentional Naming System

The INS has the following sub-components:

• Intentional naming scheme

INS names are represented as query expressions, also called name specifiers, in a carefully designed query language. At this point, INS supports exact matches of arbitrary application-specifies attributes (variables) to values. We plan to extend the operator set in the future to support range matches.

• Intentional name resolvers

Intentional Name Resolvers (INRs) to perform name lookups and message routing. INRs are deployed thoughout the network, in a similar way to current application-level infrastructure (such as DNS servers, web caches, directory servers).

The main duty of the INR is to resolve intentional names to their destinations in the network. The INR supports two methods of doing this: early binding, in which the INR returns a handle (such as an IP address) to the application (currently not implemented), and late binding, in which the application passes its data to the INR, which then forwards it along a chain of INRs to the destination; late binding has the advantage of making the name resolution process dynamic enough to support highly mobile computing, and allows the INRs to provide group communication capability even if there is no support for network multicast (the case with most of the current Internet).

• Entity discovery protocols

To discover new entities and services and track dynamism and change, the INS incorporates a name discovery process. As part of this process, the INR announces the names that it knows about to its neighbors, which are either other INRs or applications that have asked to know about particular names. The INR also learns new names, both from its neighbours (which are other INRs or applications that are announcing a particular name), and via inference, a novel way of passively learning about names while handling resolution requests.

• Resolver self-organization protocols

This is described in the six-month plan.

• INS API

INS exports a simple API for applications to create intentional names, find the best INRs to communicate with, and use the INRs to resolve names and forward their data. The API allows user applications to create, understand, and manipulate intentional names. Since these names are fairly flexible, a large library of functions is provided to make dealing with them as easy as possible. The API also allows the applications to communicate with INRs for resolving names and requesting particular services (such as anycast, multicast, late- or early-binding, notification of new names, name announcements, etc.) Together the INR and API implementations, which were written in Java to take advantage of its propensity for rapid and cross-platform development, are about 5000 lines of code (including documentation).

Prototype application

Leveraging the INS software, we have started building a prototype mobile camera application for remote surveillance. So far we have implemented the application on networks within a single administrative domain.

This application allows users to acquire images from a number of cameras located at strategic spots in the building. Features of the application include group communication, mobility, resource discovery support, and data caching, all provided by WIND middleware.

Users can control and access the camera network using attributes of the data they are interested in. For example, they can send a query of the form "location=5thfloor", "domain=lcs.mit.edu", "city=cambridge", "state=ma", "country=usa" into the network without knowing the IP address or network location of the camera(s) satisfying these attributes, and receive responses from them (the cameras
currently on the 5th floor of the LCS building at MIT). Alternatively, a user may want to send a message to all other users currently observing images from a particular location.

For example during surveillance, a security guard who observes suspicious activity in the 2nd-floor lounge can send an alert message to all security guards currently closest to that region of the building.

Because support for mobility and resource discovery are inherently provided by WIND, the camera application can take advantage of it. When a mobile camera moves, it only needs to advertise its new location to a closely located INR, and this information gets propagated through the network. Furthermore, messages currently in transit “notice” this movement and are forwarded to the new location.

WIND also enhances the basic resolution and routing support by incorporating data caching for repeatedly accessed data. This is done by setting the data source setting fields in the message header indicating that the data is cacheable, and the intermediate INRs will be automatically caching this data and responding to requests for them.

Documentation

We have written (and submitted for publication) a draft paper on our completed work as well as a detailed WIND design document. These have been sent separately to Dr. Kogiku at NTT and will also be available from us upon request.

Six-month plan

Self-organizing protocol for INRs

INR neighborhoods are currently very static: each INR has a list of neighbors that it reads at boot time. While this is very similar to the way DNS nameservers are configured today, many aspects of the system could benefit from a having neighborhoods be self-organizing. It would reduce the manual configuration that users have to do, since they wouldn't have to tell their machine which INR to use, and keep it up to date. It would reduce the amount of work that administrators have to do, since relationships could be set up automatically, based on the underlying network; this would make the INR neighborhood topology more efficient too, since it would be based on actual criteria and measurements of the network, rather than the administrator's best guess. The system would also be more scalable and reliable, since INRs could adjust the number and location of neighbors based on load, network and host outages, etc. We are beginning work on a set of self-organization protocols that will facilitate this type of operation.

Wireless beaconing protocols

In the current camera implementation, the spatial location of a camera is not automatically obtained; it is pre-configured manually. We are planning to use a scheme such as intelligent beaconing to enable mobile cameras to deduce their location and use it while

communicating with other WIND applications. We also intend to deploy the mobile camera application on larger scale in our building.

We are planning to design and implement other WIND applications using INS middleware, focusing on other devices including palm pilots and handhelds to enable remote monitoring and control of the camera network as well as other equipment (e.g., televisions).

Experiments with expanded wireless coverage and device mobility

We are planning to expand the coverage of RadioLAN network to cover more than one floor of the LCS building. Because there are different subnetworks on each floor, supporting IP mobility will be essential for seamless operation. For WIND applications, the INS already enables application-level mobility. For legacy applications, we intend to investigate two approaches for supporting IP-level mobility in the building: (i) deploy Mobile IP and evaluate it, and (ii) design a protocol to enable IP-level mobility on top of INS' intentional naming architecture. Preliminary discussions indicate that the latter might be feasible, which augurs well because we would then be able to support IP-level mobility without any kernel changes (required in contrast by Mobile IP) and without any changes to existing applications. Furthermore, unlike mobile IP, this would enable mobility without inefficient routing or single points of failure. We intend to perform extensive performance experiments of these approaches and the wireless infrastructure in the next six to nine months.