Self-Organizing Collaborative Environments

Hari Balakrishnan, MIT LCS
Srinivasan Seshan, IBM Watson
Pravin Bhagwat, IBM Watson
M. Frans Kaashoek, MIT LCS

1. Introduction

This position paper outlines a vision for smart environments based on self-organizing collaborative regions. It describes important research challenges in networking infrastructure and middleware to be tackled over the next few years, including networking over Bluetooth, ad hoc topology formation and configuration, intentional naming and discovery, and self-configuring overlays.

Our vision of smart environments of the future is motivated by three main goals:

1. Enhance collaboration: Today, it is cumbersome to set up ad hoc collaborating communities of people and their computing devices; for example, when a group of people gather together in a room and wish to exchange information, it requires significant manual effort and causes great frustration. Ideally, ad hoc interaction between people and devices should be as easy as assembling lego-style building blocks. Our vision is a collaborative region of mobile people, their networked devices, and their distributed applications, that is self-organizing.

2. Integrate the physical world: The future is certain to see a large number of networked devices that will allow us to interact with the physical world and control it better. However, the large number and diversity of such entities requires scalable techniques for the discovery and control of these devices.

3. Reduce management cost: Over time, management and operational costs have started to dominate other costs; this is because of the excessive amount of difficult and manual configuration and maintenance required both in the underlying network layer and in the organization of networked services and applications. Furthermore, with rapidly diminishing hardware cost, the number of networked devices and appliances will rapidly increase, thus increasing overall operating costs.

Our vision is shared by many others; our main contribution is a set of ideas to realize the vision: (i) ad hoc networking protocols over Bluetooth, (ii) intentional naming, and (iii) self-configuring overlays.

2. Networking with Bluetooth

A variety of technology trends have made it possible to incorporate computational capability in all devices. Unfortunately, although shrinking technology has made the devices around us "smart," it has not always improved their usability. What we need is an infrastructure via which people and systems with more usable interfaces can in turn control and interact with these smaller more ubiquitous devices. The easiest way to do this is to provide IP-level network connectivity between devices and the systems that interact with them.

The first step in this infrastructure is to provide a low cost, low power, wireless communication system. One of the key technologies we are planning on using to interconnect devices is Bluetooth. Bluetooth is an emerging, promising, low-power (0 dbm), short range (10-15 meters) RF wireless technology with the primary objective of eliminating cables between devices. Bluetooth allows the formation of relatively static short range pico-nets where a designated master node can be connected to up to seven slaves. In principle, using Bluetooth interfaces it should be possible to form an ad hoc network of devices, but the techniques for forming such networks are largely unknown today---i.e., today, Bluetooth is "wireless cabling" and not a true link layer over which IP can run.

Because of the short range nature of wireless links, not every device in our network will be able to communicate with every other device or with the wired infrastructure. We need to support routing among ad hoc groupings of devices to ensure ubiquitous reachability. The characteristics of our simple devices connected via Bluetooth make the problem different from traditional wireless ad hoc routing. In traditional ad hoc routing, the topology of the nodes is dictated by their current location and reachability. In a densely concentrated Bluetooth network, the interconnection of nodes is largely determined by the set of pico-nets that each node is participating in. In addition, the traditional ad hoc routing solutions assume that most nodes are similar in capabilities and are willing to forward packets and participate in the route discovery algorithms. Many devices may not be willing to forward packets or participate in the control protocol due to energy or computational constraints. New protocols must be developed for this environment: The combination of point-to-point and master-slave link abstractions, and energy/computation constraints affect routing protocol design.

We need protocols to allocate network addresses for these devices and automatically configure them. The scalable, automatic approach is decentralized, unlike existing schemes like DHCP that rely on a server to assign addresses and prevent address collisions. In particular, we are designing a distributed, randomized protocol for this. Because addresses signify location in a network topology, the ad hoc addressing protocol also has to handle the joining and splitting of subnetworks as nodes move and new nodes join a region.

3. Intentional naming

The large number of networked devices and services requires scalable techniques to discover and manage them. We believe that one of the core technical challenges here is in the design of a naming system allows service-providing nodes to easily advertise what they provide, and nodes that desire services to express what they want. One possible approach to this is to design a large-scale intentional naming system. Today, most network naming schemes (including the DNS) are built around the implicit assumption that what applications want to retrieve are the contents of an address, where the address signifies location in the network topology. In fact, what distributed applications want to retrieve is either information or functionality. The key insight behind intentional names is that while applications may not know best *where* to find what they are looking for, they often can express *what* they want. Thus, the first step is to support a more intentional network naming scheme in which applications describe what they are looking for (their "intent"), not where to find it.

Intentional names must be descriptive, using a carefully designed language that allows applications to express a rich set of properties and perform queries for services they want. A language that allows exact (and fuzzy) matches of attributes and values, range operations, and wild-card matches is a good starting point. The precise set of operators in the language is an important research question.

To illustrate the power of an intentional name, notice that a networked audio speaker in MIT LCS's Room 510 of Building NE-43 can be expressed using the following intentional name (or something similar):

```
      [owner = lcs.mit.edu]    /* owner's domain name */
      [entity = audio-speaker] /* type of service */
      [building = ne-43
                [room = 510]]  /* example of a hierarchical attribute */
      [id = 18310149]          /* this is a unique ID, like an IP address */
      [access = public]
```

This contrasts greatly with what one would do today, which is hard-code and manually maintain a DNS entry such as speaker510.lcs.mit.edu (and have to maintain elsewhere, manually, that this corresponds to a particular speaker). To access services, applications construct query expressions based on intentional names, describing what they desire.

To take this desired behavior and turn it into something real requires a system architecture for intentional names that scales well with a large number (potentially hundreds of millions) of names (services). We intend to use our self-configuring overlay framework to deploy intentional-name resolvers (the next section discusses this). Supporting "late binding" between names and network addresses allows us to address mobility, replication, and distribution of services. Here, name resolvers do not simply return the handle to the network location of a name; rather, they forward messages in the overlay network to the eventual intended destination(s). This integration of resolution and message forwarding enables highly dynamic name bindings, since the application is never left with a stale binding even if bindings change while the message is in transit. We believe that enabling late binding in the resolution process is key to handling dynamism in self-organizing networked systems.

4. Self-configuring overlay networks

Numerous networked services form an application-level overlay network over a network-layer such as IP. Examples of this include naming systems (Section 3), transcoding middleware, cache and replica management, performance discovery systems, etc. Overlay networks are a convenient way to add new functions to the network and they will become more important as we try to move the Internet into new application domains.

Currently overlay networks are not general (one for each application) and are static (e.g., DNS, MBone, 6Bone). To work well in dynamic environments, middleware to organize, deploy, and administer an overlay is necessary. To make self-configuring overlay networks a reality, the following issues need to be addressed:

1. Bootstrap. In the absence of even one overlay node (e.g., a name server, a transcoder, a cache, etc.), the protocol must have a bootstrap mechanism to spawn one. This is especially important in ad hoc networks without a pre-configured infrastructure.

2. Neighbor formation. One of the features of overlay networks is that they exchange information between other overlay nodes, such as meta-data, routing information, etc. To scale well and be efficient, nodes need to establish neighbor relationships with other nodes.

3. Load management. As the load on an overlay node increases, nodes need to be able to offload work on to other nodes, perhaps spawning them if necessary. This needs to be done in a decentralized manner.

4. Dissemination. Overlay nodes need to exchange task-specific information between each other (e.g., cache contents, mobile node locations, transcoder state, etc.). To achieve robustness and availability, we believe that the use of "soft-state" protocols is crucial. In this respect, the overlay architecture shares common properties with the successful Internet routing infrastructure.

5. Application control. Overlays should be able to participate in message routing, allowing applications to influence routing metrics (compared to using hop count alone).

The key difference between overlays and network-layers, such as IP, is that overlays incorporate application-specific information. We expect that a self-organizing overlay in which new network services can plug in application-specific forwarding metrics and meta-data that will allow networks to quickly adopt to new application domains.

5. Summary

As many others, we believe that smart environments should improve collaboration, integrate the physical world, and reduce management costs. To enable this shared vision, we propose three ideas: networking with Bluetooth, intentional naming, and self-configuring overlays.