A Mobility Gateway for Small-Device Networks

Robert C. Chalmers and Kevin C. Almeroth
Department of Computer Science
University of California
Santa Barbara, CA 93106-5110
{robertc,almeroth}@cs.ucsb.edu

Abstract

Networks of small devices, such as environmental sensors, introduce a number of new challenges for traditional protocols and approaches. In particular, the extreme resource constraints characteristic of these devices force the engineering of device and application-specific optimizations in order to reduce complexity. By offloading some, if not most, of the complexity out of the small device and into the network, we simplify the design of individual devices, and may make otherwise infeasible applications possible. In this paper, we look specifically at the problem of global mobility, and its associated protocol Mobile IP (MIP). We introduce the ***mobility gateway*** *which performs MIP processing on behalf of registered devices. The mobility gateway provides an interface between the optimized world of small devices and the larger Internet. Through simulation, we investigate the potential savings offered by this technique in terms of bandwidth and power. In a sample scenario, we find the total number of bytes transmitted over the wireless channel reduced by as much as 70%, and the battery life of the device extended by more than 100 hours.*

1. Introduction

Over the last few decades, we have witnessed a dramatic reduction in the size of computers. This reduction has, in turn, revolutionized the ways in which we make use of computation, imbuing a certain sense of freedom. With the advent of hand-held computers and advancements in wireless technology, mobile, untethered computing has become possible. As this miniaturization continues, many consider ubiquitous computing as the next logical step, embedding small computational devices within the environment around us [24]. Compelling applications exist for these small devices, such as Personal Area Networks and sensor networks integrated into the structure of buildings.

Current efforts to build millimeter-scale sensors illuminate the extreme resource constraints characteristic of such small devices. For example, *Smart Dust motes*, sensors developed at UC Berkeley [13], utilize a 4 MHz 8-bit processor and have only 8 KB of memory [8]. When working with such small devices, the primary goal is to reduce complexity to an absolute minimum. One solution is to ignore standard protocols and approaches, applying optimizations that depend upon device and application-specific knowledge [6–8, 21]. An alternate direction is to push some, if not most, of the complexity out of the small device and into the network.

Adding intelligence into the network can help to solve a number of problems faced by small devices. However, this goes against over two decades of practice in the traditional Internet. The end-to-end model advocates pushing complexity to the end-host, freeing the core network to simply route packets [19]. When discussing small devices, though, we are focused on edge networks, not core routers. By providing services within and at the edge of a small-device cloud, we can amortize their complexity over a larger number of nodes. Rather than forcing each node to implement a complete protocol suite, we can relax the notion of what an Internet-capable node can be. This simplifies the design and reduces the cost of individual devices, and may make otherwise infeasible applications possible.

Applying this approach to the specific problem of global mobility, we can leverage intelligent agents within the network to inter-connect small-device clouds with larger networks. In this paper, we introduce the concept of a *mobility gateway*, an intelligent router at the border of a small-device cloud. The mobility gateway performs Mobile IP[1] (MIP) [11, 16] signaling and maintains protocol-specific state on behalf of each device within the cloud. This greatly decreases the signaling load within the small-device network, and reduces the processing, memory, and power requirements of each device.

To evaluate the potential benefit of the mobility gateway, we develop a realistic simulation in which we measure the bandwidth overhead incurred due to MIP signaling. We explore a number of parameters, such as the mobility pattern of devices as well as the number of nodes communicating with each device. We find that providing the protocol as a service in the gateway results in a considerable savings in both per-packet overhead, as well as explicit signaling messages. In one simulation, the presence of the gateway eliminates up to 70% of the total bytes transmitted within the small-device cloud, and extends the battery life of a *dust mote* by almost 100 hours.

The remainder of the paper is organized as follows. We review related work in Section 2, including a short introduction to Mobile IP. We define the architecture of the gateway in Section 3, and identify its functional requirements. In Section 4, we investigate, through simulation, the source of overhead in MIP and its impact on small devices. Finally, we present our concluding remarks in Section 5.

2. Related Work

The mobility gateway is positioned between two distinct classes of network: the Internet and small-device clouds. Each class brings with it a large body of existing research that relates to our study. In this section, we begin with a short overview of Mobile IP and then discuss the most applicable projects, focusing on their distinguishing features and identifying the differences of our approach.

2.1. Mobile IP

In this paper, we focus on the specific problem of device mobility. Mobility complicates the design of network protocols, in general, because it introduces a dichotomy in addressing. Typically an address, especially an IP address, indicates both *who you are* and *where you are* in the network. To overcome this dichotomy, Mobile IP employs two addresses: 1) a home address (HoA) and 2) a care-of address (CoA). The home address is simply the node's normal IP address when it is

1 Due to the expectation of large populations of small devices and the increased address range of IPv6 over IPv4, we have chosen to focus on MIPv6. Henceforth, we will refer to IP and MIP without specifically stating the version.

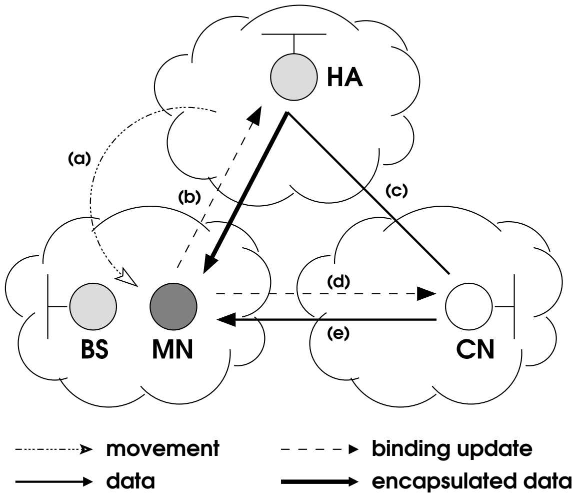

Figure 1. An example exchange in Mobile IP.

not mobile; thus, it implies *who* the mobile node is in terms of name resolution and security. The care-of address indicates *where* the node is currently in the network. As the mobile node (MN) moves between access networks, it creates a new care-of address that is topologically correct for its current point of attachment (e.g., a base-station (BS) or current access point). The primary mapping between these two addresses is maintained by the mobile node's home agent (HA), an agent in the MN's home network that acts on the MN's behalf while it is away.

Figure 1 presents an operational overview of MIP. The mobile node starts in its home network using just its home address. It has an ongoing TCP connection with a correspondent node (CN) in another network. In **step a**, the MN visits a new network and acquires a local care-of address. It then sends a binding update (BU) to its home agent (**step b**). The home agent creates a binding entry which associates the MN's home address with its current care-of address. Meanwhile, the correspondent node continues to send packets addressed to the MN's home address. The home agent intercepts these packets and tunnels them to the MN via the care-of address (**step c**).

This creates a sub-optimal route through the home agent for all packets exchanged between the mobile and correspondent nodes. To overcome this indirect route, the mobile node can send a separate binding update to the CN (**step d**). The correspondent node establishes its own binding entry, creating a secondary mapping between the two addresses. Then, packets exchanged between the two nodes can use the current care-of address directly, optimizing the route (**step e**). Since TCP and many other higher-level protocols do not allow the end-points to change during the lifetime of the connection, the home address is added to each message and transparently replaced at the IP-layer prior to passing the packet up the protocol stack.

2.2. Small-device Networks

Small-device networks have attracted considerable attention in recent years. Efforts in this area can be categorized into two main categories: projects building small-scale sensors, and protocols to efficiently deliver sensor data. Smart Dust from UC Berkeley and the WINS project from UCLA have proposed architectures for building millimeter-scale sensors [13, 17]. The key challenge has been to keep the designs simple and cheap; imagine thousands or even millions of these devices embedded in the environment around us. Providing an operating system that functions at this scale has also been difficult. TinyOS presents a simple strategy that blurs the line between kernel and application in order to reduce the size, processing, and power demands of the code [4, 8]. Most communication protocols designed for small-device networks also blur the line between protocol and application [7, 9]. The overriding theme in this area has been that standard approaches to system and protocol design cannot be directly translated onto small devices. With the use of intelligent gateways, however, standard protocols become services provided by agents within the network in order to inter-connect small devices with larger networks.

2.3. Micro-mobility Protocols

Micro-mobility protocols address the specific problem of reducing the latency of handovers. Fast MIP allows a mobile node to either start the registration process prior to handover, or continue to use its old CoA after handover until registration completes [15]. Other projects, such as Cellular IP and Hierarchical MIP, use intelligent agents in the access network to completely mask local mobility from the rest of the network [3, 18, 20]. In each of these protocols, the end-devices still use IP to communicate. The proposals simply augment normal protocol processing to improve the latency of handovers. In most cases, the solutions actually require additional complexity in the mobile node. For small-device networks, more efficient, non-IP solutions could seriously reduce the complexity of the devices, eliminating the additional overhead required to stay IP-compliant. For these new solutions, the mobility gateway provides the necessary bridge between the internal protocols and the larger, more complex world.

2.4. Mobile Networks

In an effort to integrate ad-hoc networks with the Internet [2], MIPMANET extends a basic MIPv4 foreign agent to act as a gateway between the two networks [12, 22]. Another project that more closely resembles our mobility gateway defines an extension to Mobile IP that accommodates mobile routers [5]. A mobile router moves relative to the Internet, like a MN, but also supports other nodes as a router. The authors propose the use of binding updates for an entire prefix rather than a single address. Using this technique, a mobile router would acquire a new care-of address from the local base-station, and bind its home prefix to that CoA. Any packet destined to the home prefix would be forwarded by the home agent to the mobile router which would then deliver the packet to the appropriate node. However, there are still a number of open questions concerning how to properly authorize prefix-wide bindings. Moreover, this approach requires that the nodes within the mobile network are themselves immobile, excluding many interesting scenarios where small devices move within and between mobile clouds.

3. Mobility Gateway

The goal of the mobility gateway is to offload protocol complexity and overhead from small devices while still achieving global mobility. In this section, we provide a general architecture for the gateway, and detail the required functionality. The mobility gateway sits at the border of the small-device cloud. On one side, it must appear to the larger network as a collection of fully-functional MIP nodes. On the other side, the gateway must provide an interface to the devices that requires a minimum of overhead and complexity. We start with a concrete scenario to help illustrate.

3.1. Scenario

For this example, we choose a popular scenario for sensor networks: disaster recovery. Imagine a thousand airborne sensors moving throughout a disaster area. Along with those sensors, a few larger, ground-based robots move through the scene, acting as base-stations for the sensors. The base-stations cooperate to form an ad-hoc backbone, thus interconnecting all of the devices. One robot acts as a gateway for the cloud through a satellite up-link to the Internet. From this connection, disaster relief workers, stationed around the world, access sensor readings from the site, and coordinate the tasking and movement of each sensor.

In this scenario, most if not all of the sensors can be contacted directly from the control facility. This implies that the sensors should be individually addressable from the Internet. However, these sensors have been deployed with a specialized communication protocol, such as pseudo-IP [1], intended to minimize power consumption. Moreover, within the cloud, sensors use 16-bit addresses in order to reduce per-packet overhead. In some sense, this scenario is slightly contrived to better illustrate the benefit of the mobility gateway. In truth, however, it is quite realistic since a layered approach is more cost-efficient. The thousand sensor nodes are interchangeable, simple, and cheap. The fewer, yet more expensive, robots provide an infrastructure for the sensors, amortizing their own cost and complexity.

A number of possible solutions apply even to this specific scenario. Rather than contact the sensors directly, one could deploy an application-specific border gateway. Commands would be addressed directly to the application gateway which would then disseminate them to the appropriate sensors according to its own application-level logic. All sensor readings would be collected by the gateway and possibly forwarded to the control center after some degree of aggregation. Although simpler in some regards, this solution is less flexible since each gateway is applicable to a single scenario or application. On the other hand, the mobility gateway offers a flexible approach that decouples global mobility from the application-specific nature of the small-device cloud. In fact, the mobility gateway provides a generic deployment platform for a wide range of solutions, including the application-level gateway just described.

3.2. Architecture

Figure 2 presents the general architecture of the mobility gateway. The gateway's functionality is partitioned into six conceptual components. Each component addresses a specific problem faced by the overall design. In this section, we consider each of these components in turn, exploring their individual responsibilities, as well as their interactions.

3.2.1. Protocol Translation. Working on the premise that protocols within small-device clouds are highly optimized, it is reasonable to assume that packets originating from within one cloud cannot be routed through another without some form of translation. In our scenario, the gateway must translate between specialized internal protocols and IP. In many cases, sensor protocols can be mapped to IP packets through simple encapsulation; the gateway adds an IP header around the full interior protocol. This, of course, assumes that the CN receiving the packet can process the sensor protocol in its raw form. For many of the applications envisioned, this is a reasonable assumption. Correspondent nodes acting as controllers would have specific knowledge of the sensor network and its interior protocols. In other scenarios though, like a Personal Area Network, one might imagine the gateway exporting a web interface to the correspondent node. The device, a button camera for instance, would appear as if it were a fully-capable web server; the gateway would accept HTTP requests for current snapshots, and format responses as HTML pages.

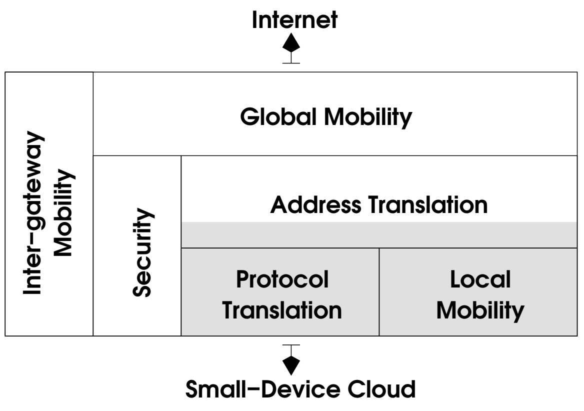

Figure 2. Architecture of the mobility gateway. The grayed region indicates application/protocol-specific components.

3.2.2. Address Translation. In a general sense, the mobility gateway can be configured to act as a mobile NAT (Network Address Translation) box. For devices that require access to external services but never need to act as servers themselves, the mobility gateway can use its own home address as the source address of packets. Similar to standard NAT boxes, separate state is maintained for different connections based on the destination CN and port numbers. The mobility gateway differs from standard NAT, however, in that it and the devices behind it are mobile.

For more advanced scenarios, like the one presented in Section 3.1, where nodes behind the gateway must be directly addressable, the gateway must maintain a mapping between each device's global home address and its internal address. Actually, the gateway maintains three distinct addresses per device:

1. **Device-only Address (DoA)** - address of the device within the cloud;

2. **Home Address (HoA)** - global IP address assigned to the device in its home network; and
3. **Care-of Address (CoA)** - topologically correct IP address maintained by the gateway.

The device-only address can be either statically assigned, or change dynamically as the mobile node moves within the cloud. The size and semantics of the DoA are determined by the application(s) running on the devices (e.g., the 16-bit addresses from our scenario). The home address is assigned to the device statically. It provides a means to uniquely access the device from anywhere in the Internet. Typically, all of the devices within a cloud would have home addresses from the same home network (i.e., share a common prefix), but this is not a requirement. The gateway maintains a bi-directional map between the two addresses: DoA and HoA. As packets enter the cloud, the home address is replaced with the DoA prior to protocol translation. As packets leave the cloud, the opposite mapping occurs.

In addition to each address maintained for a mobile node, the address of each correspondent node can be dynamically mapped to a valid device address. In our example, the gateway would choose an unused 16-bit address upon receiving the initial packet from the CN. Reply packets are reverse mapped to restore the original, 128-bit IPv6 address. The gateway maintains a single CN map across mobile nodes; thus, different devices have a consistent view of the outside world. This technique, however, has a few drawbacks. Primarily, devices have no means to directly contact a new correspondent node since a valid mapping will not already exist. To overcome this problem, a lightweight discovery protocol should be available, analogous to DNS, that results in a valid device address being assigned to the requested target. Furthermore, a special range of addresses could be pre-allocated and assigned to well-known CNs, such as sensor controllers or data collection sites.

3.2.3. Local Mobility. Mobility management within the small-device cloud is independent of the mobility gateway. Specialized protocols can be tailored to the characteristics of the devices and applications. The gateway, however, does require some information about each device in order to translate addresses properly, as well as manage global mobility for the end-device.

In the simplest scenario, most of this information can be established manually prior to deployment. Considering our example in Section 3.1, the entire network is deployed together, and sensors do not move between clouds. In this case, the home address and the associated details of the home agent would be known a priori. As the devices move within the cloud, though, the current DoA must be updated at the gateway. If the local mobility protocol allows the device to maintain a single DoA as it moves, this is trivial. Otherwise, the infrastructure (e.g., the set of base-station robots) can be leveraged to initiate this signaling on behalf of the mobile node. In more complicated scenarios where MNs move in and out of the cloud, the infrastructure would query the mobile node for the required information, and then register the device with the gateway.

3.2.4. Global Mobility. Once a device is registered, either statically or dynamically, the gateway manages global mobility for that device. In other words, the gateway performs all MIP operations that the device would normally be required to implement. This includes the functionality of both mobile and correspondent nodes.

If the gateway itself is moving within the Internet, it must function as a mobile node in its own right, as well as update the care-of address for each registered device. The gateway can allocate individual CoAs per device, or employ a single CoA for all devices. In the latter case, individual devices are still identifiable via the home address included in each message (see Section 2.1). Mobile IP state is maintained separately for each device since the binding lifetime and security properties of a single correspondent may differ across devices.

Maintaining separate data structures and sending multiple binding updates raises a question of scalability. By offloading the complexity from individual devices, we have increased the necessary complexity of the gateway. To manage a large number of devices, a gateway must be properly provisioned. In a static scenario where devices do not move between clouds, this can be accomplished prior to deployment. In more dynamic scenarios, the gateway must be capable of rejecting device registrations in the face of inadequate memory or limited bandwidth between the gateway and the Internet. The resource demands at the gateway can be reduced by only providing MIP support for those devices that must be accessed as servers. For other devices, the gateway acts as a mobile NAT (see Section 3.2.2) which requires less state and no additional signaling. Finally, if prefix-wide bindings [5] (see Section 2.4) are available, static devices (i.e., remain within the cloud) can be assigned home addresses from a specific home network, reducing the overhead to one binding update and a single data structure for the entire prefix.

Looking beyond the base Mobile IP protocol, the mobility agent could also implement companion protocols [14, 15, 23], such as Fast MIPv6 or Candidate

Access Router Discovery, to decrease the latency and improve the overall quality of handovers. By pushing global mobility management to the edge of the small-device cloud, a larger array of mobility related protocols can be deployed without affecting the cost or complexity of the individual devices within the cloud.

3.2.5. Security. To date, little attention has been devoted to general security for small-device networks. Here, we focus on the security issues directly associated with MIP. Mobile IP requires that binding updates are properly authenticated. In other words, a third party should not be able redirect a mobile node's traffic by forging a BU. This creates an interesting challenge in designing a mobility gateway. To properly register the care-of address for the mobile node with its home agent, the gateway must obtain a key for the MN. For simple cases, the key may be manually configured with the gateway. In more general scenarios, the device would provide this key as part of its registration with the gateway. By providing this key, the mobile node is affording a certain level of trust to the gateway and possibly the rest of the infrastructure within the cloud. This trust is necessary while the device is inside the cloud since the gateway is providing a critical service to the MN. Once the device moves outside the cloud, however, should the mobile node continue to trust the old gateway with the ability to re-route packets?

To address this issue, we propose the use of a temporary session key for communication with the home agent. It is assumed that the mobile node and the home agent share some secret, K_s. Upon entering a cloud managed by a mobility gateway, the MN generates a session key, K_i, from the current time, T_i, using a keyed hash function, such as MD5:

$$K_i = MD5(T_i, K_s) \tag{1}$$

This session key, along with the timestamp, T_i, is communicated to the gateway,[2] and subsequently used for BUs and encrypted tunnels between the gateway and the home agent. This technique requires that the standard binding update be enhanced to carry T_i as an option. Knowing both K_s and T_i, the home agent can reproduce K_i and validate the binding. By using timestamps for the keying material, the home agent is able to detect the use of old session keys. Any key generated with an earlier timestamp is automatically invalidated.

2 Whether the session key can be securely communicated within the cloud is dependent upon the interior protocols. The key should at least be encrypted within the infrastructure, leaving the first wireless hop as the only possible point of attack.

For bindings with correspondent nodes, security is more complicated. In a general sense, a mobile and correspondent node do not necessarily have a prior security association (e.g., a secret key). Mobile IP provides a mechanism referred to as return routability (RR) in which the CN attempts to verify that the mobile node does indeed *own* both the home and care-of addresses. This is achieved by sending multiple packets, some to the HoA and some to the CoA. The mobile node must respond to each of these packets properly before the BU will be accepted. For the mobility gateway, this requires no extra information from the mobile node. In fact, by performing RR on behalf of the device, the gateway is considerably reducing the overhead normally incurred by the mobile node.

3.2.6. Inter-gateway Mobility. After moving between clouds, a device must register with the new mobility gateway. The problem, however, is that two gateways, GW_i and GW_{i-1}, will now be managing MIP for the device. Thus, when registering with GW_i, the device provides the previous session key, K_{i-1}, which GW_i uses to authenticate a request to unregister the device at GW_{i-1}. Moreover, since the MN generates a new session key, K_i, for GW_i, the previous session key will be invalidated; any attempt by GW_{i-1} to update bindings at the home agent will fail.

4. Evaluation

Our evaluation focuses on quantifying the potential benefit of using a mobility gateway. To this end, we measure the communication overhead incurred by MIP signaling as a sensor moves within the small-device cloud. In other words, *we measure the potential overhead necessary to manage MIP if the gateway were not present.* We also consider the impact that this overhead has on power consumption in a sample device.

We continue with the scenario presented in Section 3.1. In this scenario, a sensor periodically reports readings to a controller, and the controller occasionally sends commands to the sensor (e.g., a request for a certain type of data). We begin with the simplest case where the sensor is away from home but not actively moving. This static case focuses primarily on per-packet overhead. Allowing the sensor to move within the small-device cloud causes more frequent binding updates, and an overall increase in signaling overhead. With the addition of security, the impact of Mobile IP begins to overshadow the application-level data that is being exchanged.

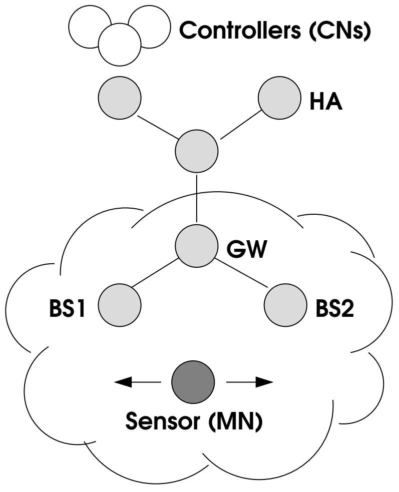

Figure 3. The topology used for simulations.

4.1. Methodology

To perform our evaluation, we implemented a modular design for MIPv6 [10] in the Network Simulator (ns-2.1b9). We chose not to use an existing implementation of MIPv6 since it lacked key functionality such as inserting the Home Address Option and providing separate timers for individual binding entries. Figure 3 presents the topology used for each of our simulations. At the bottom of the figure, the circled region depicts the small-device cloud consisting of our mobile sensor, two base-stations and a static gateway.[3] The controllers reside in the wireless network at the top of the figure; they do not move in the simulations. As parameters to the simulation, we vary the number of correspondent nodes communicating with the MN, the mobile node's mobility pattern, and the security model used to authenticate binding updates. We run each simulation with 1, 5, 10, and 20 correspondent nodes. To generate mobility patterns, we use the *setdest* utility distributed with ns-2. *Setdest* produces random way-point profiles based on the maximum speed of the node, and an optional pause time between movements. Table 1 lists the four patterns used for our simulations; five instances were generated for each pattern.

The MIP specification dictates that binding updates must be authenticated. We study the additional overhead imposed by four alternatives. The initial model, **none**, provides a baseline for the other approaches. The **auth** model uses the Authentication Data Sub-option defined in older MIP specifications [10]. It assumes the existence of a shared key between the communicating parties, and carries a message authentication code (MAC) to authenticate the update. Both **hreg** and **rr** employ a variant of return routability (RR) (see Section 3.2.5) where a session key is created when a new CoA is registered. With this variant, the session key can be used with subsequent updates that refresh the current binding, but a new key must be generated whenever the MN registers a new care-of address. The difference between the two alternatives is that **hreg** uses RR only between correspondent nodes and the MN. Home registrations use a shared key; thus, **hreg** is a hybrid of the **rr** and **auth** models.

Overall, the range of parameters provide us with 256 different simulation instances. Throughout the evaluation, we refer to a particular instance by its defining parameter values. For example, a simulation run with five CNs, mobility pattern 1, and no security is labeled as **c5m1-none**. In each simulation, the sensor issues 100 byte packets every second to each correspondent node. The CNs send 200 byte packets, exponentially distributed with an average interval of 90 seconds. All simulations are run for one hour (3600 s) in order to capture multiple cycles of expired and refreshed binding updates. For each run, we measure the total number of bytes transmitted and received over the sensor's wireless interface. From this count, we separate the overhead due to MIP from the actual data being communicated.

4.2. Results

We begin by considering the static case where the sensor is away from home, but not moving within the cloud. To better understand the source of overhead, we divide it into six categories: 1) binding updates, 2) binding acknowledgments (BA), 3) return routability, 4) home address options (HAO), 5) routing headers, and 6) encapsulation. Encapsulation occurs when the CN does not have a valid binding for the mobile node, and sends a packet through the MN's home agent. The home address option and routing header represent per-

Pattern	Speed	Pause	Interval
m0	0 m/s	0 s	∞ s
m1	10 m/s	3 s	38.9 s
m2	20 m/s	2 s	28.1 s
m3	30 m/s	1 s	21.1 s

Table 1. Mobility patterns used in simulations. m0 represents a static device - no mobility. The interval indicates the average time between changing base-stations.

3 In these simulations the gateway acts as a simple IP router.

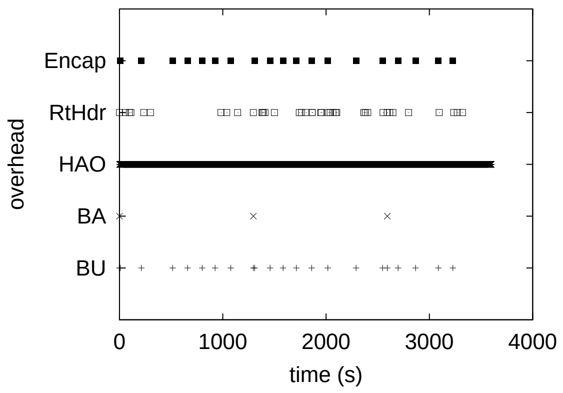

Figure 4. The evolution of protocol overhead through time for c1m0-none.

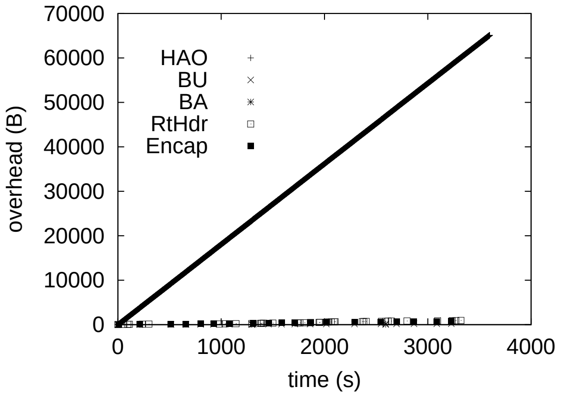

Figure 5. Protocol overhead accumulated over time for c1m0-none.

packet overhead, carrying the home address in each packet during route optimization. The BU, BA, and RR are signaling messages triggered by node movement or the expiration of a previous binding.

Figure 4 presents the evolution of each category, simulating one correspondent node, no mobility, and no security (**c1m0-none**). In this example, binding updates are mostly generated in response to encapsulated packets; when a BU expires, the correspondent node reverts to sending packets through the home agent. The pattern is periodic due to the limited lifetime of CN bindings (120 s).[4] Home registrations have a longer lifetime, around 1300 seconds, which is dictated by the lifetime of the care-of address. Three home registrations can be identified in the figure by their corresponding acknowledgments. The most interesting part of the figure is the continuous line of home address options. One is sent with each sensor report.

If we look at each category in terms of bytes accumulated over time (see Figure 5), it becomes clear that the per-packet overhead (HAO) dominates, accounting for 98% of the total. Moreover, Table 2 shows that overhead accounts for nearly 16% of the total bytes sent and received, including both data and protocol overhead. This measure, which we refer to as the *overhead ratio*, is mostly independent of the number of correspondent nodes since the cost of binding updates is far outweighed by the per-packet overhead. Once we introduce security, however, differences begin to emerge. For 20 CNs, the overhead ratio grows to as much as 25% (**c20m0-rr** in Table 2).

Introducing mobility to the scenario heightens the effect that binding updates have on the total overhead. Although, without considering security, per-packet overhead still dominates. By comparing **c1m0-none** and **c1m1-none** in Table 2, we see that in the case of a single CN, mobility increases the overhead ratio by approximately 4% (from 16% to 20%). For 20 CNs, the increase is only 6-10%. If we do consider security, however, the overhead begins growing more dramatically. For the **auth** model, the overhead ratio climbs to 35-55%, based on the number of active correspondent nodes. In the most extreme case (**c20m3-rr**), signaling dominates per-packet overhead (see Figure 6). In this case, total overhead accounts for nearly 72% of the total bytes communicated.

Protocol overhead clearly has a detrimental effect on the utilization of wireless bandwidth. Another interest-

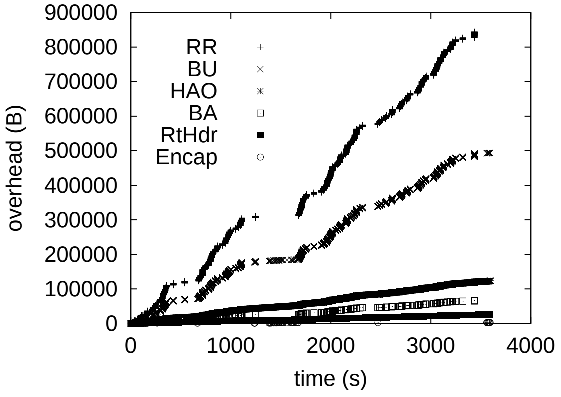

Figure 6. Protocol overhead accumulated over time for c20m3-rr.

4 The lifetime is on the scale of minutes to reduce the chance of hijacking a node's address.

	none	auth	hreg	rr
c1	**m0**: 0.155	**m0**: 0.165	**m0**: 0.167	**m0**: 0.170
	m1: 0.197	**m1**: 0.353	**m1**: 0.382	**m1**: 0.460
	m2: 0.209	**m2**: 0.398	**m2**: 0.434	**m2**: 0.516
	m3: 0.206	**m3**: 0.384	**m3**: 0.419	**m3**: 0.500
c5	**m0**: 0.156	**m0**: 0.179	**m0**: 0.187	**m0**: 0.189
	m1: 0.196	**m1**: 0.359	**m1**: 0.446	**m1**: 0.493
	m2: 0.224	**m2**: 0.447	**m2**: 0.549	**m2**: 0.596
	m3: 0.225	**m3**: 0.450	**m3**: 0.551	**m3**: 0.598
c10	**m0**: 0.159	**m0**: 0.201	**m0**: 0.216	**m0**: 0.218
	m1: 0.213	**m1**: 0.426	**m1**: 0.559	**m1**: 0.589
	m2: 0.242	**m2**: 0.501	**m2**: 0.638	**m2**: 0.665
	m3: 0.229	**m3**: 0.465	**m3**: 0.595	**m3**: 0.624
c20	**m0**: 0.161	**m0**: 0.223	**m0**: 0.244	**m0**: 0.245
	m1: 0.219	**m1**: 0.451	**m1**: 0.610	**m1**: 0.627
	m2: 0.257	**m2**: 0.542	**m2**: 0.699	**m2**: 0.714
	m3: 0.260	**m3**: 0.547	**m3**: 0.701	**m3**: 0.717

Table 2. Ratio of MIP overhead to total bytes sent and received by a MN for each mobility pattern. All measurements have a 99% confidence interval of no more than $\pm$0.004.

ing effect is the impact on the device's consumption of power. To estimate the power consumed by transmitting overhead, we use the energy specifications of UC Berkeley's *dust mote*, a millimeter-scale sensor [8]. At peak load, the *dust mote* consumes 19.5 mA of current at 3 volts (see Table 3). In idle mode, the current drops to 3 mA, and when switched to inactive mode, the sensor draws only 10 μA. The battery is rated at 575 mAh which equates to about 30 hours, 200 hours, and more than a year of lifetime in the active, idle, and inactive modes, respectively.

In the **c20m3-rr** datasets, the sensor sends, on average, over 1 MB of overhead and receives more than 600 KB over the hour-long simulation. The sensor's radio consumes approximately 1.0 μJ to transmit a single bit, and 0.5 μJ to receive. The total energy spent on overhead comes to about 11 J. Considering the energy budget in active mode is 211 J ($19.5mA * 3V * 3600s$), 11 J seems inconsequential. However, if the time spent transferring these extra bits had been spent in a non-active mode, the effect is more dramatic. In idle mode, 11 J equates to 1222 seconds, or about half an hour. If the sensor had spent that time inactive, the battery life would be extended by more than 100 hours.

Mode	Load	Budget	Lifetime
active	19.5 mA	211 J	30 h
idle	3 mA	32.4 J	200 h
inactive	10 μA	5.4 mJ	> 1 yr

Table 3. Power profile of the Dust mote. The budget represent the amount of energy consumed in one hour.

In this evaluation, we have identified only potential savings. We do not consider the overhead required to perform local mobility within the cloud. In other words, we compare extreme alternatives; either the device implements MIP itself, or mobility is free. We do not expect local mobility to come without its own complexity, but separating the explicit overhead incurred by global mobility offers a better understanding of the impact of local solutions. Although the specific numbers reported here are a reflection of our sample scenario, we feel that the conclusion is unavoidable. By offloading protocol complexity from the small device, we can maximize the utility of the device while still providing a full range of functionality.

5. Conclusions

The fundamental goal driving this work is to make small-device networks both viable and flexible. Most current research has focused on application-specific approaches. By pushing protocol complexity out of small devices and into the network infrastructure, we make possible a new range of applications. By providing standard protocol support in intelligent agents, we offer small-devices citizenship in the realm of larger networks.

In this vein, we have focused on providing support for global mobility. The mobility gateway offers Internet connectivity to mobile devices that would otherwise be confined within their small-device clouds. We have shown that, by implementing Mobile IP on the gateway rather than on each device, huge savings are possible for already heavily constrained resources. As devices shrink even further, intelligent agents will have an ever larger role to play in making small-device networks feasible.

References

[1] K. Almeroth, K. Obraczka, and D. De Lucia. A lighweight protocol for interconnecting heterogeneous devices in dynamic environments. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems (ICMCS '99)*, Florence, Italy, June 1999.

[2] J. Broch, D. Maltz, and D. Johnson. Supporting hierarchy and heterogeneous interfaces in multi-hop wireless ad hoc networks. In *Proceedings of 4th International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN'99)*, Perth/Fremantle, WA, Australia, June 1999.

[3] A. T. Campbell, J. Gomez, S. Kim, A. G. Valkó, C. Wan, and Z. R. Turányi. Design, implementation and evaluation of Cellular IP. *IEEE Personal Communications*, 7(4):42–9, Aug. 2000.

[4] A. Chandrakasan, R. Amirtharajah, S. Cho, J. Goodman, G. Konduri, J. Kulik, W. Rabiner, and A. Wang. Design considerations for distributed microsensor systems. In *Proceedings of IEEE Custom Integrated Circuits Converence (CICC'99)*, San Diego, CA, USA, May 1999.

[5] T. Ernst, C. Castelluccia, and H. Lach. Extending Mobile-IPv6 with multicast to support mobile networks in IPv6. In *Proceedings of Universal Multiservice Networks*, Colmar, France, Oct. 2000.

[6] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar. Next century challenges: Scalable coordination in sensor networks. In *Proceedings of Mobicom'99*, Seattle, WA, USA, Aug. 1999.

[7] W. R. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of Mobicom'99*, Seattle, WA, USA, Aug. 1999.

[8] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. Culler, and K. Pister. System architecture directions for networked sensors. *Operating Systems Review*, 34(5):93–104, Dec. 2000.

[9] C. Intanagonwiwat, R. Govindan, and D. Estrin. Directed diffusion: A scalable and robust communication paradigm for sensor networks. In *Proceedings of Mobicom'00*, Boston, MA, USA, Aug. 2000.

[10] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. Technical Report draft-ietf-mobileip-ipv6-15.txt, Internet Engineering Task Force (IETF), July 2001.

[11] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. Technical Report draft-ietf-mobileip-ipv6-*.txt, Internet Engineering Task Force, Jan. 2003.

[12] U. Jönsson, F. Alriksson, T. Larsson, P. Johansson, and G. Maguire. MIPMANET - mobile IP for mobile adhoc networks. In *Proceedings of Workshop on Mobile Ad Hoc Networking (MobiHOC'00)*, Boston, MA, USA, Aug. 2000.

[13] J. M. Kahn, R. H. Katz, and K. S. Pister. Next century challenges: Mobile networking for "Smart Dust". In *Proceedings of Mobicom'99*, Seattle, WA, USA, Aug. 1999.

[14] J. Kempf (Editor). Problem description: Reasons for performing context transfers between nodes in an IP access network. Request for Comments (Informational) 3374, Internet Engineering Task Force (IETF), Sept. 2002.

[15] R. Koodli (Editor). Fast handovers for Mobile IPv6. Technical Report draft-ietf-mobileip-fast-mipv6-*.txt, Internet Engineering Task Force, Sept. 2002.

[16] C. Perkins and D. Johnson. Mobility support in IPv6. In *Proceedings of Mobicom'96*, Rye, NY, USA, Nov. 1996.

[17] G. L. Pottie and W. J. Kaiser. Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–8, May 2000.

[18] R. Ramjee, T. F. La Porta, L. Salgarelli, S. Thuel, K. Varadhan, L. Li, and S. Y. Wang. HAWAII: A domain-based approach for supporting mobility in wide-area wireless networks. In *Proceedings of 7th International Conference on Network Protocols (ICNP'99)*, Toronto, Ontario, Canada, Oct. 1999.

[19] J. Saltzer, D. Reed, and D. Clark. End-to-end arguments in system design. *ACM Transactions on Computer Systems*, 2(4):277–88, Nov. 1984.

[20] H. Soliman, C. Castelluccia, K. El-Malki, and L. Bellier. Hierarchical MIPv6 mobility management (HMIPv6). Technical Report draft-ietf-mobileip-hmipv6-*.txt, Internet Engineering Task Force, July 2001.

[21] L. Subramanian and R. Katz. An architecture for building self-configurable systems. In *Proceedings of Workshop on Mobile Ad Hoc Networking (MobiHOC'00)*, Boston, MA, USA, Aug. 2000.

[22] Y. Sun, E. Belding-Royer, and C. Perkins. Internet connectivity for ad hoc mobile networks. *International Journal of Wireless Information Networks*, 9(2), Apr. 2002.

[23] D. Trossen, G. Krishnamurthi, H. Chaskar, R. Chalmers, and E. Shim. A dynamic protocol for candidate access-router discovery. Technical Report draft-trossen-seamoby-dycard-*.txt, Internet Engineering Task Force (IETF), Oct. 2002.

[24] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, Sept. 1991.