USENIX Association

Proceedings of MobiSys 2003: The First International Conference on Mobile Systems, Applications, and Services

San Francisco, CA, USA
May 5-8, 2003

© 2003 by The USENIX Association All Rights Reserved For more information about the USENIX Association:
Phone: 1 510 528 8649 FAX: 1 510 548 5738 Email: office@usenix.org WWW: http://www.usenix.org
Rights to individual papers remain with the author or the author's employer.
Permission is granted for noncommercial reproduction of the work for educational or research purposes.
This copyright notice must be included in the reproduced paper. USENIX acknowledges all trademarks herein.

Contact Networking: A Localized Mobility System

Casey Carter Robin Kravets Jean Tourrilhes
Department of Computer Science, UIUC *Hewlett Packard Labs*
{ccarter,rhk}@uiuc.edu jt@hpl.hp.com

Abstract— **MobileIP, the standard for Internet mobility, enables transparent mobility for a mobile node, but requires communication to take a multihop path through the node's Home Agent. Although a user with a multiple-interface mobile node may desire the ability to communicate locally, perhaps while disconnected from the Internet, MobileIP offers no such support.**

Contact Networking provides lightweight, localized network communication to a node with diverse network interfaces. The goal is to provide support for local connectivity equivalent to that provided by MobileIP for remote connectivity. The concept of link-layer awareness enables Contact Networking to tailor its operation to different links, using link-layer native services to implement abstract services when possible. Interface management and autoconfiguration insulate the user from concerns about the number and type of interfaces available.

In this paper, we motivate the need for localized mobility, and present the design and architecture of Contact Networking. Details of our prototype implementation illustrate the complexities of providing a localized mobility facility.

1. Introduction

Recent years have witnessed the development of a new class of mobile computing devices, merging personal communications and data access into a single device. One part mouse and two parts remote control, these mobile nodes are the interface point between users and the information environment. As users move through the world, they want to use their mobile nodes to maintain reachability through access to the global Internet, a form of mobility that is already well supported by MobileIP [21]. However, users would benefit from avoiding expensive multihop Internet connectivity and taking advantage of computing resources and services discovered locally, using whatever network interface is most convenient.

The inspiration for the name Contact Networking is to orchestrate direct communication between a mobile node and its neighbors. This focus on last-hop connectivity addresses the shortcomings of earlier work that assumes the presence of a network interface, with no concern for how that interface is link-layer configured. Contact Networking's approach to mobility support is diametrically opposite that of MobileIP. MobileIP treats all communication as though it is remote, even if local, whereas Contact Networking treats all communication as though it is local, even if remote. This difference in philosophy enables Contact Networking to provide lightweight service by being link-layer aware, tailoring itself to the characteristics of each link layer.

Contact Networking manages both IP and link-layer configuration, connecting to neighbors using all available interfaces. Exploring the environment informs Contact Networking when communication is possible. As long as some pair of compatible interfaces exists, Contact Networking can maintain connectivity between neighbors. To preserve battery power and maximize flexibility, interface activation is delayed until applications indicate demand to communicate with a given neighbor. Contact Networking configures network interfaces to provide link-layer connectivity and bidirectional routing between the two nodes using a mechanism developed in earlier work [33]. Communication is maintained as links form and break by rerouting to other interfaces as necessary. This *neighbor handoff* mechanism, P-Handoff in earlier work [32], is transparent to the user and ensures persistent connectivity.

In addition to active interface configuration and routing support, Contact Networking provides name resolution to enable operation without infrastructure. The Contact Naming System (CNS) allows users to refer to local resources with friendly names. CNS is implemented by mechanisms native to each link layer. Since distant communication relies on the primitive hop-by-hop interaction of neighbors, Contact Networking extends to manage multihop end-to-end communications when infrastructure access is available.

In this paper, we present the design of and case for Contact Networking. Section 2 motivates better mobility support for mobile nodes, particularly localized mobility. We present the requirements for localized networking in Section 3, which is realized by the Contact Networking architecture of Section 4. Some examples detail the operational requirements for localized networking in Section 5. Section 6 discusses our prototype implementation of Contact Networking, including limitations and lessons learned. Section 7 contains the natural progression from Contact Networking to future work.

2. Motivation

We define local and remote communication, and detail the lack of support in mobility solutions for localized networking. We advocate link-layer awareness, the

This work was sponsored by Hewlett Packard Labs and National Science Foundation grant number ANI–0081308.

idea that the network layer should incorporate link-layer knowledge to provide better service. The following hypothetical scenario provides a context in which to discuss the issues of localized networking:

You and your colleague Xue board the train to travel to the client presentation. Xue hasn't seen the final version of the video, so you both point your PDAs' infrared ports at each other, and you push the video file to her PDA. Your PDAs connect over infrared, establishing a local communication channel without requiring Internet connectivity. Seconds after the transfer is initiated, you put your PDA back in your briefcase. This breaks the infrared link, so your PDA switches automatically to 5GHz radio to continue the transfer.

After a few moments, you decide to walk back to the dining car. Due to the limited range of 5GHz radio, that link also soon breaks, forcing the communication to fall back to 2.4GHz radio. As you walk away, you leave 2.4GHz radio range and your PDA's only option to preserve communication is to activate its GSM radio and communicate over the Internet with Xue's PDA. For the first time, your PDA registers with MobileIP and begins using multihop remote communication.

In the dining car, you use 5GHz radio to exchange business cards with a new acquaintance, without even needing to see his PDA – you simply push the card to whichever device in your vicinity responds to the nickname "Gerard." This transaction does not interfere with the ongoing transfer to Xue's PDA three train cars away.

As you return to Xue's car, the PDAs discover they can again connect to each other over local links. They shut down the expensive GSM radios and handoff the communication to the faster, cheaper 5GHz radio channel.

A. Local and Remote Communication

Mobile nodes carry out two fundamentally different kinds of communication, which we term local and remote communication. Local communication is between a mobile node and another node with which it shares direct connectivity. The defining characteristic is locality—interacting with "this" device that the user discovers in the environment, such as a local printer. A local communication is a simple unplanned association between two hosts, requiring no infrastructure access, but only that the two nodes share compatible communication hardware. Two users who meet each other and wish to exchange data between their mobile nodes, like the video in the scenario, use local communication.

Remote communication is between a node and some other explicitly addressed node that is not in the immediate neighborhood. Web browsing is the classic example of remote communication. The web browser knows the web server's identity, but not its location. The mobile node must have access to the network infrastructure to communicate. Unlike the case of local communication, there is no possibility of spontaneous interaction. In the scenario, when local communication links no longer provide connectivity, the PDAs automatically fall back to remote GSM communication over the Internet.

Remote communication is more resource-intensive than local. A short local transaction could be completed by briefly powering up a network interface and exchanging packets. A remote communication requires a mobile node to power up an interface, find an infrastructure provider, register with its home agent and then perform the communication before unregistering and powering the interface back down. The setup time necessary to register the mobile node's location, together with much longer delays incurred by the multihop path, keep the network interface powered over a longer time.

A mix of concurrent local and remote communication is not supported by current mobility approaches. MobileIP focuses on multihop remote communication [21]. MobileIP even explicitly forbid a node to directly communicate with any neighbor except its Foreign Agent:

> While the mobile node is away from home, it MUST NOT transmit any broadcast ARP Request or ARP Reply messages. Finally, while the mobile node is away from home, it MUST NOT reply to ARP Requests in which the target IP address is its own home address. . .

The PDAs in the scenario would be unable to communicate via MobileIP had access to the Internet been unavailable, even though they share compatible hardware. If a network administrator was available, the devices could be manually configured. The administrator must choose between two alternatives:

- Use the shortest range, lowest power link supported by both nodes. This conserves battery power for the mobile nodes, but constrains the users' motion within the limited range provided.
- Use the longest range, highest power link supported by both nodes. This provides a large area within which users can move freely. However, they pay for this flexibility with battery power.

Neither approach is optimal in all cases. From the users' perspective, the optimal solution would be to use the lowest power, shortest range links that provide connectivity at any given time. This dynamic approach is untenable for manual configuration.

B. Multiple Wireless Technologies

There is no perfect wireless technology for all applications. The design of a wireless technology is a tradeoff between range, capacity and power consumption. As soon as a technology comes to market it is obsoleted by newer, faster, more efficient technology. There will always be periods of transition from the current to the new wireless technology during which mobile nodes will support multiple links.

The management of multiple links is not integrated well into the traditional network stack. A multi-interface mobile node should be able to simultaneously utilize as many interfaces as necessary to satisfy its communication needs. In fact, users should not need to be aware of the number or type of interfaces available.

Among the many types of wireless technologies we pay special attention to a distinguished class we call *Directional Area Networking* technologies. These Directional links are characterized by short range and directional transmission characteristics, for example, infrared and laser. Although directional technologies are often deprecated, we see their limitations as features. Short range and directionality make directional technologies an ideal choice for a picking mechanism – selection of a particular device in the environment simply requires the user to point and click.

C. Link-layer Awareness

One of the best qualities of the Internet Protocol is its support for a variety of link layers by providing a uniform interface at the network layer. Although not every link layer was designed for IP, IP works over every link layer [3], [5], [10], [15], [23], [25], [26], [27], [29]. This least-common-denominator approach loses link-layer specific advantages. By restricting its expectations to simple datagram delivery, IP loses the richness of the individual link layers.

Link-layer awareness provides access to the capabilities of each link layer. The advantages are 1) lightweight discovery mechanisms for finding new neighbors, 2) tighter attachment to neighbor nodes, allowing rapid detection of link failure, and 3) the ability to perform link-specific optimizations, such as header compression, when appropriate. The richness of wireless links in particular makes link-layer awareness more attractive.

To clarify, consider two different wireless link layers: IEEE 802.11 [13], and IrDA (Infrared Data Association) [17]. Both links natively detect link breakage. These link failure mechanisms provide useful information to a link-layer aware network layer. The aware network layer can immediately stop using the broken route and more rapidly find a new route. In the absence of link-layer failure indications, some periodic messaging system must be used to monitor link integrity. MobileIP monitors links between mobile nodes and foreign agents with periodic beacons. Link-layer awareness conserves resources—the link layer monitors the link, whether the network layer pays attention or not—and can provide more rapid notifications than higher-layer monitoring. In general, link-layer awareness enables higher layers to transparently take advantage of implementations tailored to each link layer by providing abstract APIs for low-level services.

In the absence of link-layer awareness, the network layer must provide basic services like neighbor discovery. This duplication of service already in the link layer wastes resources. In addition, the network layer cannot possibly provide service of the same quality provided by the link layer mechanism without tailoring its approach to specific link layer characteristics. The end result of this approach is a fragile service only suitable on a single link layer, precluding true interface heterogeneity.

3. Design Requirements

The user experience of localized networking must be identical to that of infrastructure networking. The user should not need to use different interfaces or tools for local vs. remote communication. With that goal in mind, we propose the services necessary to support local connectivity and configuration-free multi-interface networking. We also address deployment, since a scheme to support localized networking is useless if not deployable in the current Internet.

A. Support Services

The critical services for localized networking are:

1) Neighbor Discovery
2) Name Resolution
3) On-demand Interface Binding
4) IP Autoconfiguration
5) Neighbor Routing
6) Channel Management
7) Infrastructure Access

We present a justification for each service requirement.

1) Neighbor Discovery: Neighbor discovery allows a mobile node to determine who its one-hop neighbors are. Neighbor knowledge enables a node to determine which interfaces may be used to contact a particular neighbor. In the absence of a neighbor discovery service, a mobile node would be incapable of performing local communication at all.

IPv6 Neighbor Discovery (IPv6ND) [18] is unsuitable for providing discovery to mobile nodes. First, its operation is not fully specified for multihomed hosts. Second, Neighbor Unreachability Detection may be too slow to support efficient handoffs when a neighbor is actively supporting a flow. Foremost, any IP-based solution obviously requires IP to be up and configured, conflicting with the goal of on-demand interface binding. Using IP as part of the mechanism to bootstrap IP creates a chicken-and-egg problem that is avoided by performing discovery at a lower layer.

2) Name Resolution: To support localized naming, a name-to-address mapping mechanism is necessary, like DNS provides in the Internet. If mobile nodes are to communicate directly each node must participate in the name-to-address mapping. Additionally, it is convenient

for the user to select a nearby device without needing to know its proper name. For example, a name that denotes "the device my Directional Area Networking port is pointing at" enables the picking mechanism discussed in Section 2-B.

3) On-demand Interface Binding: *Binding* is the process of configuring an interface to use a particular link-layer medium, like plugging an Ethernet card into a jack. Interfaces have an unbound state in which the interface is not capable of full communication but may be able to perform some tasks, such as scanning for neighbors. Once bound, the interface can send and receive IP packets. Although they lack jacks, wireless interfaces also require configuration to select the link-layer medium. Even broadcast interfaces like 802.11 have parameters that bind the interface to one physical channel at a time. To be link-layer neighbors, two nodes must have interfaces that are bound to the same medium.

There are two reasons to delay interface binding. First, it is important to keep interfaces in a low-power state, since mobile nodes are battery powered. Second, some interfaces provide point-to-point links and can be used to communicate with only one neighbor at a time. To save power and allow flexibility in choosing to which link to bind, a local communication scheme must delay binding as long as possible. Therefore, interface binding should occur on-demand, when an application indicates demand to communicate with a neighbor. Certainly, it is inefficient to maintain connectivity with neighbors with which the node has no need to communicate. Interface unbinding must also be automatic when demand is removed.

4) IP Autoconfiguration: Interface binding does little good if the user must manually configure IP address and netmask. To enable transparent network access, interface IP configuration, including assigning IP address and netmask, must also be automatic.

Link-local (LL) addresses, in either IPv6 [6], [9] or IPv4 [4], provide autoconfiguration. Unfortunately, addresses with scope restricted to a single link provide little support for persistent communication. Transient addresses cannot be used as endpoints for a transport layer connection, or that connection will fail upon moving. There is no mapping from LL addresses to home addresses that identify mobile nodes. These two deficiencies render LL addresses little more than IP-layer aliases for link-layer addresses. Further, both IPv4 and IPv6 address require an initial period on the order of several seconds for duplicate address detection before the address can be used. This delay is hardly conducive to performing rapid handoffs across multiple interfaces.

5) Neighbor Routing: The final step necessary to neighbor communication is the establishment of symmetric routing state. Neighbor routes are all that is needed to complete the local communication. Once these routes are available, application layer communication can continue normally.

6) Channel Management: The previous services provide local communication to the mobile node, but do not shield the user from managing links and interfaces. Two abstractions are fundamental to the operation of channel management. First, a *connection* is a link-layer abstraction that is formed between two neighbors who wish to communicate. Presence of a connection indicates that a handshake has taken place between these neighbors signifying that both have agreed to the communication. Second, a *channel* is the end-to-end abstraction through which transport flows propagate. We say that the channel between two nodes is realized by a particular connection at any given time.

Although connections can break due to mobility, the goal of mobility support is to preserve the channel across these breaks. If the user is actively communicating with a neighbor through a channel over a connection that fails, the mobile node must maintain the channel by connecting over another interface if possible (see Figure 1). Similarly, if a mobile node is connected to a neighbor over technology A and a new, better path over technology B is discovered, the channel should automatically switch to the new interface.

7) Infrastructure Access: Enabling a mobile node to communicate locally is an achievement, but not if the local focus completely occludes remote communication. The ability to access the Internet is far too critical to sacrifice in the name of other goals. In addition to the locally-oriented scheme, remote communication is necessary to complete localized networking support.

B. Deployment Considerations

The goals of autoconfiguration, neighbor routing, and channel management are not achievable by only a single node of a neighbor pair. Bidirectional communication

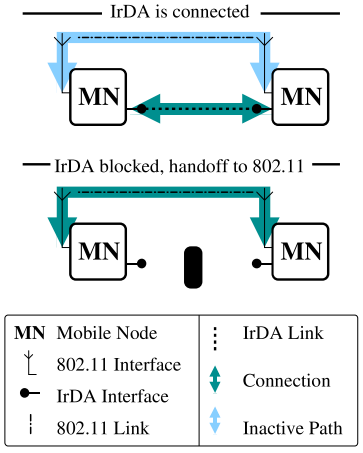

Fig. 1. Link-layer feedback and route failover

necessarily requires support from both neighbors. Consequently, it is impossible to provide localized communication support when directly communicating with legacy network nodes.

To be deployable, a localized communication scheme must not require support from every node in the Internet. The localized scheme can put no special requirements on applications; it should allow unmodified binaries to work with existing networking APIs. One way to achieve this level of transparency is to isolate the scheme at the network layer or below, so that transport and application layers see the standard IP network-layer interface. Applications and users are presented with the illusion of continuous network service, with no effect other than a temporary aberration in service during handoffs.

Application- and transport-layer mobility support advocates claim that applications must see mobility events in order to adapt to changed network service [20], [30]. Note that providing the illusion of network stability does not necessarily preclude the potential to provide mobility notifications to interested applications. It only reduces the impact on applications that are *not* mobility-aware.

4. Architecture

We present Contact Networking's architecture and discuss how its individual modules fit the requirements presented in Section 3. Since the requirements span route management to interface configuration, Contact Networking lies at the junction between the link and network layers in the OSI network stack, extending into both layers as depicted in Figure 2. Contact Networking includes a link-layer agnostic network layer module and several link-layer aware modules. The components of the network layer module manage routing and select between multiple paths available to a neighbor through different interfaces.

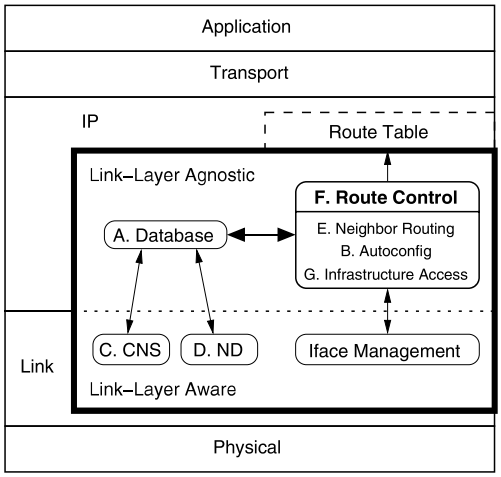

Fig. 2. Introducing Contact Networking into the OSI stack

Components of the link-layer-aware modules monitor the environment and report discovery events and link transitions. For local communication, link-layer modules also monitor link activity and inform the node when neighbors move out of range (i.e., a link breakage event). Contact Networking's link-layer discovery mechanisms incorporate the functionality of the Address Resolution Protocol (ARP) [24], mapping IP addresses to link-layer addresses.

Our description of Contact Networking architecture begins with its detailed component structure, as depicted in Figure 2. The specific architectural components discussed are:

A. the network database, where the mobile node stores its model of the environment,
B. the IP autoconfiguration technique, which assigns addresses to bound interfaces,
C. the Contact Naming System (CNS), a name resolution scheme for Contact Networking nodes,
D. the Contact Networking approach to neighbor discovery,
E. the neighbor routing component,
F. the routing control, which directs interface binding, neighbor routing, channel management and infrastructure access,
G. the infrastructure access component, which Contact Networking uses to access the Internet.

For each architectural component, we describe its responsibilities in fulfilling the design requirements, as well as how it interacts with other components.

A. Network Database

Contact Networking includes a database that models a node's current view of the environment. This database is the central communication point for all parts of the architecture. The four basic abstractions with which it is populated are interfaces, links, neighbors and paths.

- Interfaces are network attachment points on a node, such as Ethernet interfaces or cellular modems.
- Links are the communication media to which interfaces bind. An interface can bind to only one link at a time. Treating links as first-class entities makes it possible for the database to model both point-to-point links, with one neighbor per link, and broadcast links, with many neighbors per link.
- Neighbors are the nodes in the environment with which the user may wish to communicate.
- Paths are a pairing of a link and a neighbor augmented with the neighbor's address on that link.

An interface can bind to only one link at a time. Contact Networking can simultaneously establish connections with any and all neighbors to which it knows paths on a bound link. To establish communication with a neighbor, route control chooses a path to that neighbor

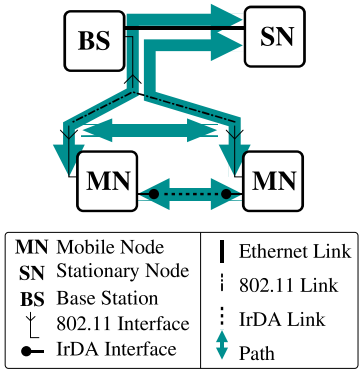

Fig. 3. Interfaces, Links, Paths, & Neighbors

from the database to activate. Path activation may necessitate interface binding and IP autoconfiguration using the configuration state for the link in the database. Contact Networking synchronizes path activation between neighbors, ensuring that both neighbors use symmetric routes. Once bidirectional path activate is complete, the paths are considered connected.

A path can be in one of four states:

- Available - The path is believed to exist but is currently inactive.
- Connected - The path is currently capable of transporting packets to the neighbor.
- Blocked - The path was connected, but is currently in a transient failure condition.
- Unblocked - The transient condition has passed, but the path has not been reactivated.

The blocked and unblocked states make it possible for Contact Networking to temporarily reroute traffic to another path while the primary path is recovering.

To clarify these abstractions, Figure 3 depicts a network composed of two mobile nodes and one stationary node. The base station simply bridges the wired and wireless links, and is therefore not visible to Contact Networking. The stationary node has one interface, and each of the mobile nodes has two. There are two links in the figure, one IrDA and one 802.11/Ethernet. The two bridged segments form a single link. Each mobile node knows a path to the stationary node, and two paths to its mobile counterpart, one over 802.11 and one over IrDA.

B. IP Autoconfiguration

A network interface must be configured with an address and subnet mask before IP can use it. In traditional IP, providing unique interface addresses is a problem in its own right, requiring either configuration or protocol solutions [4], [6], [7], [22]. These heavy-weight solutions require significant setup time to allocate an interface address, and create another problem when considering mobility—a transport flow cannot outlive the IP addresses used to identify its endpoints. If those addresses are transient, interface-specific addresses then mobility is impossible. The goal of mobility is to provide a communication channel that persists beyond the lifetime of a single address-to-interface configuration.

Contact Networking embraces and extends the MobileIP approach to providing a persistent communication channel in the face of link mobility. A *home address* is associated with the mobile node itself with no subnet mask. MobileIP configures the home address on a particular interface providing mobile networking through only that interface, or selectively configures the home address on the single interface that currently has connectivity, enabling vertical handoffs [31]. Contact Networking extends this notion to its natural conclusion, configuring the home address on *all* network interfaces simultaneously.

Due to the expanded role of the home address in Contact Networking, we designate it the Globally Routable IP address (GRIP). Every node is identified by a unique GRIP. Contact Networking requires GRIPs to be permanently assigned static home addresses, so that the presence and uniqueness of the GRIP can be guaranteed even when disconnected from the infrastructure. Contact Networking associates each neighbor's GRIP with its entry in the network database. Use of the GRIP for local communication in Contact Networking neatly sidesteps the problem of providing distinct addresses for each network interface. Configuring the GRIP on all interfaces allows the mobile node to persistently communicate through all interfaces simultaneously. The GRIP is guaranteed to live longer than any communication channel.

Using the same address on several interfaces is in direct conflict with the traditional IP addressing model in which one IP address denotes one network interface. Semantically, Contact Networking removes the emphasis that IP places on the role of the network interface. To Contact Networking, interfaces are transient entities, but the GRIP always stays the same. Since all application communication between nodes uses the GRIP, mobile nodes are accessible and successfully mobile as long as at least one network interface is present. The choice of which interface to use for a particular channel becomes a pure routing problem, a choice of path from the network database, and not a question of interface management or configuration.

C. Name Resolution

The Contact Naming System (CNS) is Contact Networking's name resolution and name-to-address mapping facility. CNS names are structured exactly like DNS names, with textual names separated by dots. CNS names enable users and applications to transparently control the operation of Contact Networking by specifying CNS names in place of DNS names.

Every node has a full CNS name, which is exactly the fully-qualified DNS name that resolves to its GRIP.

Agreement between CNS and DNS on the proper name for a node enables remote and local communication to use the same name.

CNS also supports link-layer specific name aliases, "wildcard" aliases and service aliases that give a user flexibility in choosing a neighbor. Link-layer specific aliases combine a CNS name prefix with a suffix denoting a particular link layer. This facility enables a user to specify that they want, for example, `bob.irda`, which resolves to any neighbor whose CNS name starts with `bob` reachable via IrDA. The wildcard alias `any` can be used to select any reachable neighbor, perhaps additionally qualified with a link-layer specific suffix. The infrared picking mechanism described in Section 2-B can be easily implemented with the alias `any.irda`.

A node replies to queries with its CNS record, containing its name and GRIP, and possibly service information. For example, the node `bob.cs.uiuc.edu`, which provides print service and acts as a MobileIP foreign agent, might have the following CNS record:

GRIP: `128.174.244.37`
Name: `bob.cs.uiuc.edu`
Services: printer, MIPFA

Contact Networking attaches each CNS record to the corresponding neighbor record in the network database. Name queries that can be satisfied from the database require no network traffic. Link-specific alias queries that cannot be answered from the database only result in network traffic on links of the specified type. An request to resolve `bob.irda` queries for `bob` only on IrDA.

CNS is similar to other multicast name resolution schemes [1], [8], [19]. The main difference is in implementation. Our goal of link-layer awareness means that CNS is transported by link-layer specific mechanisms instead of relying on IP multicasting support as in other approaches. Furthermore, the lifetime of CNS records is tightly coupled with neighbor knowledge in the network database. This tight coupling allows CNS records to be cached long-term without refreshing, as long as the neighbor is still present.

Each node monitors its interfaces for CNS queries requesting its configuration, although CNS may not use this request-reply model on every link layer. A Contact Networking node uses link-layer native mechanisms when available to disseminate and retrieve CNS records. Some links perform discovery while in the unbound state, so link-layer aware CNS conserves power and lowers delay over an approach that uses IP to resolve names. Nodes with sufficient power may also actively advertise their CNS records either through a link-layer discovery method or by periodic broadcast. These advertisements enable other interested nodes to easily collect CNS records by passive listening.

A node returns its CNS record in response to a the following types of CNS queries:

- Name queries
- GRIP queries, enabling reverse lookup
- Service queries (e.g., a node that provides MobileIP Foreign Agent service might respond to the name `MIPFA`)
- Wildcard queries, possibly link-layer qualified

These CNS query types provide several ways for users to choose machines in their environment with which to interact, without requiring changes to IP network software that uses DNS names. CNS access is integrated alongside the normal DNS name resolution mechanism on the nodes, so that application name lookups try CNS resolution before DNS. A unified name resolution API makes localized networking easily available to network applications. For example, to purchase a can of Dr. Pepper™ from a vending machine, a user would point the infrared port of their device at the vending machine and point their web browser at `http://any.irda`. The CNS resolver would return the vending machine's CNS record that specifies its name and GRIP. An exchange of business cards with another user named Bob would probably use the name `bob` to locate the GRIP of Bob's mobile node. We call this "getting a GRIP" on the destination.

D. Neighbor Discovery

Neighbor discovery must incorporate link-layer awareness. Determining when another node is within the sphere of communication of a particular interface on a particular link is necessarily link-layer specific. Link-layer adaptation modules perform neighbor discovery and indicate neighbor appearance/disappearance to the link-layer agnostic module. Each link-layer aware discovery mechanism is tuned to the particular link, using native discovery mechanisms when available. This level of link-layer awareness provides high quality discovery on each link layer without compromising the generality of the rest of the Contact Networking system.

What other systems term neighbor discovery, finding a reachable peer on a link, is really path discovery in Contact Networking. Neighbor discovery in Contact Networking is only complete after two neighbors have exchanged CNS records. After the exchange of CNS records Contact Networking updates the network database with a new path entry and a new neighbor entry, if the neighbor is not already present.

Contact Networking's use of CNS records for neighbor discovery combines neighbor discovery, address resolution and name resolution into a single protocol. This combination reduces total overhead in contacting a neighbor, as well as allowing for long-term caching of name-to-IP address and IP-to-LL address mappings. The tight coupling of address mappings to neighbor lifetime means that mappings need never be refreshed and are simply flushed when a neighbor goes away.

E. Neighbor Routing

Contact Networking takes link-layer network semantics, a flat address space with one-hop communication, and presents it to the transport layer as a set of host-specific routes. This IP facade on link-local connectivity enables IP applications to use the localized networking facility equivalently to traditional IP support.

Routing necessarily affects the address model of Contact Networking. The use of GRIPs for local communication creates a class of *local* routes administered by Contact Networking. There is exactly one such route corresponding to each neighbor in the network database. A local route can be inactive, serving as an indication that Contact Networking can provide a path to that neighbor on demand, or active, enabling packet forwarding across some connected path.

Traditional IP routing, which views IP addresses as specifying a location within the internetwork, is retained in Contact Networking for routing to remote destinations. The result is a bipartite routing system that provides maximum flexibility in routing directly to neighbors—via local routes–but does not lose the ability to interface to the Internet with traditional routes. Integrating both styles of routing enables transitional deployment. Contact Networking mobile nodes can contact the current Internet through access points much like MobileIP foreign agents that provide Contact Networking service in addition to traditional IP routing. In the absence of Contact Networking-enabled access routers, a mobile node must resort to traditional techniques for unreachability detection and cannot provide rapid handoff when the link to the access router fails.

F. Route Control

Route control in Contact Networking handles on-demand interface binding and channel management. When a Contact Networking node first wishes to contact some IP address, it performs a lookup in the network database for a neighbor whose GRIP is that address. If the neighbor entry is present, its CNS record and at least one path will be known—the path over which the neighbor was discovered. Route control selects one of the available paths to connect and negotiates connection with the neighbor. If no record is present, the node falls back on traditional IP networking to find a remote route to the destination, possibly initiating infrastructure access as detailed in the next section. Route control also performs channel management by disconnecting idle connections and unbinding idle interfaces when the last connection on that interface is dropped. When better paths are discovered, route control actively switches channels to better connections.

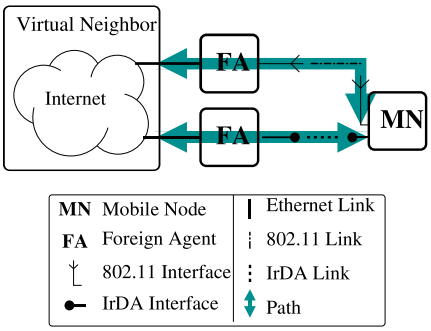

Fig. 4. Infrastructure as a virtual neighbor

G. Infrastructure Access

Since Contact Networking views all communication as direct neighbor communication, some extension is necessary to support remote communication. Contact Networking treats remote communication as local communication with a virtual neighbor that represents the entire infrastructure network. This conversion of remote to local communication enables Contact Networking to apply its normal mechanisms for on-demand establishment and channel management to infrastructure access.

In Contact Networking, a network node that wishes to provide infrastructure network access to others by performing as a MobileIP Foreign Agent advertises its willingness to do so through the Contact Naming System. Mobile nodes that need infrastructure access then discover the infrastructure provider normally through neighbor discovery. Upon discovering an access provider, Contact Networking creates a virtual path to the virtual neighbor that corresponds to the real path to the access provider (see Figure 4).

Integrating dial-up access into this architecture is trivial. A dial-up phone number is essentially a path description for a type of virtual path. Whenever a mobile node can use a dial-up interface, it is one hop away from the virtual neighbor. This virtual path would be statically configured into Contact Networking, with a simple monitor module that "discovers" the virtual path when a modem is plugged in or when within GSM coverage.

Contact Networking route lookups for IP addresses which do not correspond to neighbors in the network database are redirected to the virtual neighbor. Connecting a path to the virtual neighbor also connects its corresponding real path and performs MobileIP registration through the neighbor on the real path.

If a mobile node needs to maintain Internet access to enable externally initiated channels, Contact Networking can be configured to never drop virtual connections due to idleness. Normal neighbor handoff mechanisms would then preserve the channel to the Internet whenever possible, keeping the mobile node accessible to the rest of the world.