# *Nom*: Resource Location and Discovery for Ad Hoc Mobile Networks

Diego Doval and Donal O'Mahony
Networks & Telecommunications Research Group (NTRG)
Trinity College Dublin, Dublin 2, Ireland
{diego.doval, donal.omahony}@cs.tcd.ie

***Abstract*—**
**Mobile networks typically involve a variety of network protocols and devices, in many cases coupled with quickly evolving topologies (as devices are turned on and off and devices move across network boundaries). In this context, resource location and discovery becomes a difficult problem. Current large-scale systems designed for IP-based networks such as the Internet rely on static, hierarchical name resolution and resource location systems such as DNS or LDAP, or on limited autoconfiguration systems such as DHCP, that are not well-suited to quick updates and changes in topology because of their dependence on a central server and its static, hierarchical nature. A resource location and discovery system for mobile networks must be able to adapt to an evolving network topology and maintain correct resource location even when confronted with fast topological changes. It should also work in an ad hoc network environment, where no central server is available and the network can have a short lifetime, and it should connect across networks that might be operating on different network protocols or be connected through a third network (e.g., the Internet) as long as the nodes in the network implement the system. To avoid redundancy, the system must be suited to resolve the location not just of node names but of any resource, such as services, people, etc.**

**In this paper, we present *Nom*, a resource location and discovery system based on a Peer-to-Peer protocol that fulfills those requirements. We analyze the protocol and we discuss how a testbed implementation of the protocol was used to provide name resolution for a completely distributed, self-organizing voice-over-data application.**

**KEYWORDS**
***Mobile Ad Hoc Networks, Resource Location and Discovery, Name Resolution, Self-Configuration, Self-Organizing Networks***

## I. INTRODUCTION

Mobile networks typically have highly dynamic topologies, not only in terms of frequent *membership changes* (i.e., nodes entering and leaving the network), but also in terms of *node mobility* (i.e., changes in the physical location of a node and its relation to other nodes in the network while maintaining its logical identifiers and resources). For example, devices in a mobile network might be turned on or off and mobile phones might quickly switch locations, sometimes switching to environments where a central server is not directly accessible or might not be available. Additionally, typical scenarios for mobile computing include a mix of devices (e.g., handhelds, embedded, base stations, desktop computers) that use different physical layers (such as Bluetooth, IEEE 802.11b, Ethernet) with different protocols, depending on various factors such as location and power consumption requirements. These unique qualities mean that certain protocols and systems that work well in more static, homogeneous topologies (of which the best example is the Internet) might perform badly or not work at all in these kinds of networking scenarios.

*Resource location and discovery* is a common problem in networking. On static or slowly evolving (e.g., IP-based) networks, resource location is managed by a variety of hierarchical, centralized services. Our algorithm should be able to provide resource location at many different levels and eventually replace the functionality of most of those centralized services. Of those services, resource location and discovery services such are extremely important to bootstrap the network communication process. Throughout this paper, we will look at the the problem of resource location for mobile heterogeneous networks in the context of name resolution as a starting point, because it is the lowest level of logical-to-physical mapping on the network and the most basic service necessary. Our work, however, can be easily applied to any resource-location and discovery/autoconfiguration[1] task in networks with highly dynamic topologies such as ad hoc networks[2].

On the Internet, it is the Domain Name System [9] (DNS) that provides the lowest level of name resolution service available. Name resolution creates mappings of easy-to-remember names to physical nodes. In the case of the Internet, this means mapping service names to IP numbers. DNS is hierarchical and highly static, and requires propagation of names from a root name server to a series of slaves across the network. Each IP subnetwork has a fixed static reference to the physical location of the local name server node, using it to resolve the names of all other nodes into IP numbers so that communication with them can be established. In a mobile network, this scheme fails not only because nodes might quickly change their physical location or disappear, but also because a node might often find itself in "unknown" networks where the physical name server node is not known.

[1] In general, autoconfiguration can be considered a subset of the resource location problem if resource location provides a "bootstrap" mechanism that can then initialize a node's operating/connectivity parameters. These parameters would normally be found in centrally managed locations but in a mobile network they might be distributed across nodes or even networks, or they might depend on the network topology at the time the node is initializing.

[2] In several instances, we will mention specific solutions to the problem of providing these services in an ad hoc network, which provide the most common example of wireless self-organizing network that don't depend on servers at any point in the network's lifetime.

The most common solution for node mobility in IP networks [14] uses autoconfiguration methods such as DHCP [1] to assign temporary IP addresses and provide IP addresses for name servers, gateways, etc. Autoconfiguration methods, however, still rely on a static server properly configured and maintained within the range of the ad hoc network. While this is not unreasonable in an office or home environment, it is not feasible in a completely autonomous ad hoc network with a short lifetime or where the setup of permanent or semi-permanent base stations is not possible (e.g., when a network is formed for emergency crews at a disaster site).

Even in cases where Mobile IP is an acceptable solution to provide name resolution between nodes and to the Internet, assuming an appropriate gateway is available, it doesn't address the need to resolve node names from other nearby networks. Currently, autonomous wireless networks (ad hoc networks) have no way of referencing each other's nodes, and Internet-based nodes have no way of referencing nodes within an ad hoc network, even if Internet connectivity is available to the network nodes themselves. For example, a Bluetooth node has no way of referencing a handheld device that might be running on a wireless network (e.g., IEEE 802.11b) even though the Bluetooth node might have a connection path available through a desktop computer connected to it. This kind of functionality would be useful for global communication systems, such as Internet-based VoiceIP phones or SMS (Short Messaging Systems) designed to run on mobile ad hoc networks. In the example mentioned before, of emergency crews at a disaster site, even if a base station was set up to provide connectivity within the nodes in one network (e.g., for Paramedic crews) and to the Internet, nodes from nearby ad hoc networks (e.g., Fire Department) and from the Internet (e.g., a government official trying to contact somebody on site from a remote location) would not be able to locate those nodes. In a simpler scenario, a Bluetooth-enabled cell phone would not be able to find a non-Bluetooth device such as a handheld computer. Using simple resource location, the cell phone might be able to determine that the handheld device is in range and therefore it should not display appointment reminders, since a device that is better suited for it (the handheld) is active in the vicinity.

The problem is then how to provide resource location between nodes running on different networks, and on networks without a central server that can provide the resource location service.

Peer-to-Peer [13] (P2P from now on) provides a solution for this problem. P2P is the name given to applications and protocols that create global behaviors from a set of local node-based rules. P2P networks thus avoid reliance on centralized management and provide support for highly dynamic topologies at the cost of potentially reduced performance and higher processing requirements on each node.

In this paper, we present *Nom*, a P2P-based system that addresses the resource location problem on heterogeneous networks. Our system is designed to address the issues mentioned above, by providing:

- Local resource location within a single network, including networks with no central server, such as ad hoc networks.
- Resource location between nearby heterogeneous networks regardless of their type.
- Two-way resource location between distant networks as well as between ad hoc and Internet-based nodes by using Internet-based P2P networks.

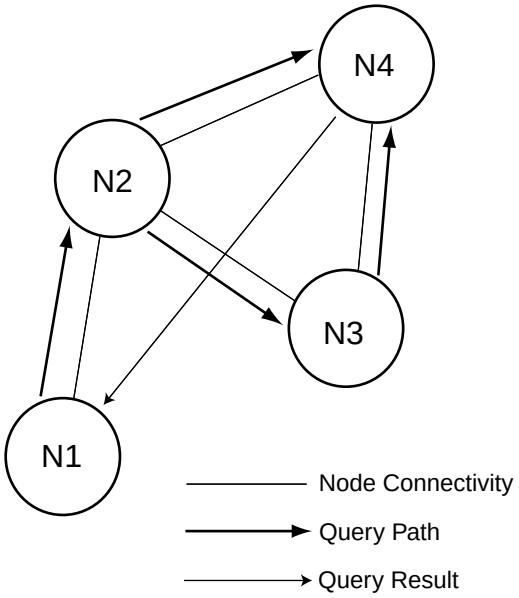


Fig. 1. A Sample P2P Network and Query

The system we describe could eventually be used to support a multitude of different applications, from basic network services, such as (as mentioned before) name resolution, to finding public information for the owner of a particular network node, to finding other network resources such as base stations and printers. More importantly, our system supports this kind of communications across heterogeneous networks, allowing remote server-based or server-less networks to communicate, creating a truly global framework for communications.

## II. Peer-to-Peer Technologies and related work

Peer-to-peer protocols and applications are distributed systems without centralized control or any kind of hierarchical organization, where the software running in each node is equivalent in functionality and capabilities to any other node. Nodes in the network form a graph; each node in the graph is an individual computer or a program and an edge between two nodes implies that information can be exchanged between those two computers or programs.

Centralized network topologies (such as that defined by DNS) restrict information access to resources only to that available in servers (in the case of DNS, name-to-address mappings). In contrast, P2P networks and applications take advantage of resources —storage, processing, content— available anywhere in the network. Global behavior in a P2P network (be it of applications or protocols) *emerges* from the sum of local behavioral patterns in each of the nodes, as opposed to a centralized network where behavior is the result of policies set and enforced by a server. P2P has been used for several different applications, from persistent distributed storage to data search on vast Internet-based networks.

Figure 1 illustrates a simple P2P-style search. In the figure, node $N1$ is requesting the value associated with a string located in $N4$ (and for which only $N4$ can provide the value). The query is transmitted across the network while the result returns directly to the node requesting the information. In a sense, the network itself resolves the resource mapping requested. Once the lookup has been solved, it can be cached by the requesting node so future requests will resolve faster. Additionally, intermediate nodes can cache it as well.

The P2P scheme described in the diagram is the one used by the Gnutella Network [7]. In Freenet [2] the query returns through the same node-to-node path established by the request, therefore guaranteeing local anonymity (i.e., each node only knows the next node in the chain, though access to the complete network traffic would still expose the origin and destination of the information).

Chord [19] uses a variant of hashing called *consistent hashing* to create "buckets" of name-to-location mappings. Chord's hashing scheme can be proven to provide consistent, scalable performance, and it is an example of a new type P2P self-organizing networks, commonly referred to as an *Overlay Network*. Other examples of Overlay Networks are CAN [15] and Pastry [17]. Overlay Networks create a topology where neighbor nodes are defined by content values rather than physical location. In doing so, they turn the problem of search from a graph-traversal problem into a set of steps that evolve according to a mathematical function, reducing the load in the network and making queries deterministic. However, because of their nature, Overlay Networks can't perform substring queries (i.e., only full strings can be searched). We will come back briefly to this in the conclusions of this paper.

Interestingly, P2P networks have a lot in common with wireless ad hoc routing algorithms, which are also designed to function without a central server and must deal with highly dynamic network topologies (i.e., topologies with frequent membership changes, or where nodes are mobile thus often changing the connectivity/routing patterns). Ad hoc routing protocols, however, deal with actual physical mappings (i.e., physical path to get information to that node) while P2P networks act based on a *logical* topology that doesn't necessarily map to the physical topology and are therefore free to implement more complex or more efficient connectivity or caching schemes.

While all of the P2P methods mentioned above share common features with the system we propose here to our knowledge there is no other system that provides resource location for heterogeneous networks, both within each network and across networks that might or might not be in range with each other.

## III. Resource Location in Mobile Networks

Before setting out to solve the problem of resource location in ad hoc mobile networks, we will first mention the main requirements such a system should satisfy. Specifically, a resource location system, and in particular a name-resolution system must ideally provide:

- Support for fast membership changes, providing resource location for the new nodes and maintaining validity of the query results, i.e., providing results that are up to date in terms of topology.
- Support for nodes changing their physical location within the network.
- No special configuration requirements to provide resource location capabilities for nodes joining the network.
- Correct results for queries for the remaining nodes of the network when one or more nodes leave the network or are switched off.
- No special protocol configuration requirements for nodes moving between heterogeneous networks.
- No dependence on any particular node or underlying transport protocol to provide resource location.
- Two-way resource location between heterogeneous networks that might be physically out of range, might use different routing protocols, or both.
- Two-way resolution between other network nodes (particularly Internet nodes) and mobile (including ad hoc) nodes.

The last point in particular underlies an interesting problem. While many applications of mobile networks focus on providing internet access to mobile nodes, little emphasis has so far been placed on supporting the opposite: resolution of mobile nodes' names *from* Internet nodes. This might be useful in a variety of situations, including two-way communications that don't depend on a particular server or infrastructure. In looking for generic solutions to the resource location problem, we have considered this as an important requirement to be fulfilled.

## IV. *Nom*

*Nom* is a fully decentralized resource-location system, based on a simple P2P protocol. *Nom* operates in completely distributed fashion, with each node running a copy of the *Nom* code, monitoring its local node network traffic to detect resource location queries and using standard messages to resolve those queries and provide resource location to application-level code.

The *Nom* algorithm is therefore composed of a loop that monitors messages coming both from the network and the application level code (i.e., *query*/*query-reply* messages from the network and *query-initiate* messages from the application) and reacts according to the message type received, either forwarding the message received if it doesn't apply to the current node, creating and forwarding an appropriate *query-reply* message if the node should respond to the query, or creating a *query* message and inserting it into the network.

The following set of steps describe the basic *Nom* algorithm running on every *Nom*-enabled node:

1) Receive *Nom* message. If the message has already been processed (i.e., its Message ID is found in the internal Message ID[3]) list, ignore it.
2) Retrieve a list of neighbors (obtained from the underlying routing protocol). Alternatively, in a mobile environment a direct broadcast can be attempted.
3) If message is a *query-initiate* message, build the query message and send it to the neighbors.

[3]Message IDs must be globally unique. Universally unique IDs can be obtained by concatenating a variety of data including current system time, node ID, and other elements such as ethernet address. Some operating systems (such as Microsoft Windows) allow creation of globally unique IDs via API calls.

4) If message is a *query* message, check whether this node contains the information requested on the query. If so, create a *query-reply* message and send it to the neighbors so it can go back to its destination [4]. If the message's number of hops is past the TTL limit, ignore it. If the information requested is not in the current node, increment the number of hops in the message and re-send the query message to the node's neighbors. After re-sending the *query* message, store the Message ID in the ID list (for use of the list see step 1).
5) If the message is a *query-reply* message, check whether the request was sent by this node (see Step 3). If the *query* was sent by this node, return the result to the application layer that made the resource-location request. If the query was not sent by this node, increment the number of hops in the *query-reply* message and re-send it to the node's neighbors, storing the Message ID in the ID list (for use of the list see step 1). Note: the query-reply contains the original query information, making storage of queries unnecessary. Since the query also has a timestamp, the node is able to determine that a timeout on the query has already occurred (and only cache the query for future use instead of passing the result to the application).

The above steps describe the basic *Nom* protocol. *Nom* is a library that exposes a set of functions to be implemented for the particular underlying platform (Operating System and Routing Protocol used). Once implemented for a particular platform, several optimizations are possible, including caching of neighbors' physical addresses (depending to the dynamics of the network), return of the query messages directly to the requester (using the underlying routing protocol, such as DSR[8], and others).

### A. Nom *and Gnutella*

A resource location system for heterogeneous mobile networks is useful in its own right, but we are interested in using an Internet-based P2P network to create connectivity between mobile networks that, while not within connectivity range with each other, might be able to use the Internet to route information between them, potentially creating a global resource location system for mobile networks. Using one of the already running Internet P2P networks has the obvious advantage of providing a readily available network of large numbers of nodes. As an example, the Gnutella network has a minimum of 100,000 nodes within any other node's "horizon" at any given moment, while the Kazaa Network, on which P2P systems such as MusicCity.net are based, have a minimum of a million nodes accessible by any other node at any given moment [5].

Gnutella is a P2P network widely used on the Internet, mainly used to exchange data (Digital Music, Software, etc) [18]. We chose the Gnutella network for two main reasons:

- It has several open-source implementations, which simplify access to the protocol if it were necessary.
- Its network has wide reach across the Internet.

While *Nom* is a new P2P protocol, its low-level primitives are easily translatable to Gnutella primitives. Several Gnutella features that *Nom* doesn't implement are tied to assumptions on topology and network behavior that apply well to the Internet but do not apply to a protocol that must to run both on mobile (including ad hoc) and Internet-based networks. Despite those differences, Gnutella nodes should be able to route *Nom* queries without knowledge of the particulars of the *Nom* algorithm, allowing resolution between distant networks as long as both are connected to the Internet and they are within range of each other in the Gnutella topology.

## V. Algorithm Analysis and Simulation

As with any service that will be used frequently by network applications, it is important to understand both its performance impact on the network and the performance on the algorithm itself under different circumstances. To that end, we are currently working on a detailed theoretical analysis of the algorithm and on a simulation to obtain statistical data for the algorithm's performance under different network configurations (number of nodes, average number of neighbors) and network loads. Elements of this problem have already been analyzed for the Broadcast Storm problem, in [6].

*Nom*, much like Gnutella and Freenet is an example of a P2P network that performs TTL [6]-controlled flooding. While it is clear that flooding-based schemes present scalability problems, the size of networks such as Gnutella, with hundreds of thousands of nodes operating concurrently, makes it clear that they *can* work, and this is certainly true for the typical network sizes encountered in ad hoc mobile networks.

The three main factors to take into account when analyzing the algorithm are:

- The total traffic it generates depending on the number of nodes in the network.
- The traffic it generates within a node's range, which could potentially limit the bandwidth available to each node if the local (i.e., in-range) traffic generated by the algorithm is too high.
- The speed with which an answer can be received is also important. This *query-reply* speed is directly related to the average path length for the network. As mentioned in [3] the average path length (number of hops required) for a given transmission between nodes is expected to grow with the spatial diameter of the network, that is, the square root of the area, or $O(\sqrt{s})$ for a fixed transmission range capacity per node.

In the paragraphs that follow we provide results that already highlight some limits to the algorithm as well as potential points for improvement.

Our simulator is built on top of the Swarm Simulation System[5], a software package widely used for multi-agent simulation of complex systems. The system allows one to emulate a world consisting of several nodes, each running the *Nom* protocol, each with a unique physical ID and node name.

[4] In most cases, the reply could be sent directly to the requester as an optimization. The basic *Nom* algorithm, however, makes no assumptions in that regard.

[5] Gnutella and Kazaa Simultaneous usage average obtained through the month of Nov. 2001.

[6] Time-To-Live

For each run of the experiment, a node chosen at random inserts query for another random node. In each cycle, every node processes the messages that arrived in the previous cycle. This simultaneous processing of messages is a simplification of the real world case, but it allows us to find the upper bound of messages set by maximizing the number of simultaneous messages that could be theoretically be sent at any given instant.

Our measurements indicate that the main factor conditioning performance for a network running *Nom* is not the total number of nodes, but rather the average number neighbors for a given node[7].

Based on the values of Table I, it is possible to find the maximum bandwidth "allocated" to resource location by using the following formula:

$$\text{number of messages} = \frac{BP}{S}$$

Where $B$ is the total bandwidth (in bytes per second), $P$ is the proportion of bandwidth to allocate to resource location (which can be configured by an end-user application or operating system setting) and $S$ is the average message size. As an example, assuming resource location was to be confined to a peak of 20% of bandwidth, for a 2Mbps system, it would mean a bandwidth usage limit of 45 KBytes. This value translates into 1800 messages per second. Therefore according to table I *Nom* would support at most 20 average number of neighbors for the system at the desired 20% bandwidth, allowing one query per second at a constant rate. This result assumes a query-resolution cycle of a duration of 250 msec. If the value is smaller, more queries can be resolved per second at a constant rate for that bandwidth usage proportion. While this value might seem low, in general, current mobile networks (and ad hoc networks in particular) rarely involve either a large number of nodes or large average number of neighbors (since other factors, such as power consumption, limit the range available for transmission and therefore the number of neighbors). The number-of-neighbor limit therefore points to an issue that will have to be solved in future versions of the algorithm so that it maintains its usefulness in high-density networks.

Because *Nom* in its current form is essentially performing a controlled flooding on the network, large networks will create a performance problem both in the query resolution time and the resources needlessly used throughout the network (i.e., by propagating the query when it has already been resolved). While this is not a problem for current mobile network sizes (consider the total number of messages required to reply a query in table I for networks of 20 to 30 nodes) it will certainly be an element to consider for future mobile networks, which will be larger, or applications such as sensor networks where the number of nodes in the network can easily grow to the thousands. The conclusion mentions this an other challenges to overcome before *Nom* can be used for networks of any size and characteristics.

[7] On Table I the value "Peak Messages In Range" represents the average maximum number of messages within a range in the network, and it can therefore be used to calculate the "bandwidth cost" of the *Nom* service. This value is obtained by averaging the peak number of messages per simulation cycle.

## VI. Other Issues

### A. Security

A key issue in resource location is security, particularly in a decentralized network such as the one created by *Nom*. When resolving the physical location of resources, a node should be able to verify that the location resolved is valid and current. Without security, a malicious network user that has access to the message flow in the network could:

- impersonate other nodes and resources by answering requests for them.
- modify query results being passed along and change the values passed.

In DNS, security is largely an issue of trust between clients and servers. Both of the problems described can happen in DNS, assuming that a name server (slave or master) has been compromised (commonly called DNS "spoofing"). Additionally, if a gateway has been compromised, the DNS requests themselves can be manipulated by a malicious third-party.

In *Nom*, the verification of the identity of the node location result is currently left to higher level application layers with enough information to make these checks (for example, by verifying signed security certificates), just like DNS does. In the future, we plan to add capabilities to Nom so that initial verifications can be made directly at the resource-location level by using cryptographic digital signatures. Similar modifications to DNS are currently under consideration at the IETF[16].

### B. *The* Nom *Namespace*

The *namespace* of a resource location protocol is the standard used to identify resources in the network. In DNS, the namespaces are defined by alphanumeric combinations ending in a particular string that identifies the registering organization, such as ".net" or ".com" and they are part of the protocol definition, i.e., DNS can only resolve queries for resources defined in its namespace.

Since *Nom* is intended to operate in heterogeneous networks and support different applications (from person to person calls to machine identification), enforcing a DNS-style namespace is not acceptable. *Nom* can therefore resolve any string stored as a mapping in its internal database for a particular node. Applications can then make use of it in different ways, for example, storing DNS-style names or using SIP[10]. The application can simply store the mapping desired and other applications running on different nodes will be able to resolve it regardless of the type.

## VII. Case Study

*Nom* is a generic library that can be used in any network, provided that a set of functions are implemented for the underlying platform. To create a testbed for *Nom*, therefore, we chose an already implemented network stack and provided it with resource location capabilities that could be in turn be exposed to applications built using it, initially keeping the resource location functions for name resolution only.

| Average Neighbors | Avg. Total Cycles Until Finished | Avg. Cycles Until Reply Received | Total Msgs. To Resolve Query | Peak Msgs. In Range |
|---|---|---|---|---|
| 3 | 22 | 13 | 715 | 16 |
| 6 | 14 | 9 | 1356 | 67 |
| 10 | 12 | 7 | 1990 | 145 |
| 15 | 9 | 5 | 3130 | 415 |
| 21 | 7 | 4 | 4191 | 791 |
| 26 | 7 | 4 | 5189 | 1304 |
| 33 | 6 | 3 | 6552 | 2315 |
| 52 | 5 | 3 | 10330 | 6724 |
| 55 | 5 | 3 | 10962 | 7396 |
| 69 | 4 | 3 | 13672 | 9604 |
| 79 | 4 | 2 | 15740 | 9801 |
| 95 | 4 | 2 | 18955 | 9801 |
| 99 | 3 | 2 | 19604 | 9801 |

TABLE I
SIMULATION RESULTS

### A. Implementation

For our *Nom* implementation testbed we chose the ad hoc network stack developed by the NTRG at Trinity College Dublin[12]. The NTRG stack was chosen because of its simplicity and extensibility, and the availability of different routing algorithm implementations as well as physical connectivity layers, including software radio and 802.11b. NTRG is also supported by the JEmu [4] radio simulator, greatly simplifying creating and running tests scenarios of several devices for debugging and evaluation.

The NTRG stack uses the concept of *layers* to define abstraction boundaries behind which different elements of a complete ad hoc stack can be implemented, including low-level connectivity, routing and security, among others. Therefore, for this case study, we implemented the low-level platform dependent functions of the *Nom* library to match the requirements of the NTRG layer abstraction.

### B. The Nom Resource Location Layer

The current version of the *Nom* library is built for Windows operating systems. It uses the lowest common denominator of Windows-based API functions (Win32 base) so it is portable across a variety of Windows-based systems, including Windows 98, Windows 2000 and Windows-CE based platforms such as Microsoft PocketPC. While the current implementation of the *Nom* library is Windows-based, porting the library to other operating systems or platforms should not be difficult due to the simple nature of the operations required (i.e., thread management) and because most of the platform-dependent functions are abstracted into a set of function calls that are implemented as necessary for each platform on which *Nom* is deployed.

The *Nom* testbed was thus implemented as a library where the platform dependent details interacted with the capabilities provided by the NTRG stack. In particular, the *Nom* interface functions that had to be implemented were:

- *SendMessage(Message m. Node target)*
- *Message WaitForMessage()*
- *Array GetListOfNeighbors()*
- *Message BuildMessage(String msg)*
- *String DecodeMessage(Message m)*

*Nom* allows use of the resource location system using these simple functions. *Nom* requires no changes in its code as long as the library that implements these functions adheres to the interface specified above.

*Nom* exposes two functions to the application-level layer:

- *Initialize(Array names)*
- *String resolveName(String name)*

Once the NTRG-compatible low-level implementation for *Nom* was completed, we integrated it with one of the applications that uses the NTRG stack, the 4GPhone [11]. The 4GPhone uses an implementation of the NTRG stack on Microsoft PocketPC to provide voice service over an ad hoc network that uses NTRG protocols for routing and security. 4GPhone is designed to be completely independent of servers or switches, creating a foundation for a global, voice-over-data phone system. The 4GPhone implementation used for the *Nom* testbed was running on top of 802.11b in ad hoc mode, with an encryption layer to provide secure communications and a DSR routing layer for general packet routing. Figure 2 illustrates a sample configuration of the ad hoc network used. Figure 3 shows the particular NTRG layer-connection scheme used in the 4GPhone application.

In Figure 2, each of the dotted circles represent the wireless range for the device at its center. A device is considered within range of another if the dotted circle that defines its range intersects or includes the other device. Note that *Nom* would also work properly even if the devices were connected asymmetrically, i.e., D2 within "output" range of D4 but D4 only within "output" range of D3. Each of the devices included a stack configuration as follows:

The 4GPhone first initializes the *Nom* library to map the name of the owner of the device to the physical ID of the node in the ad hoc network. Bob can type "John" in the input dia-

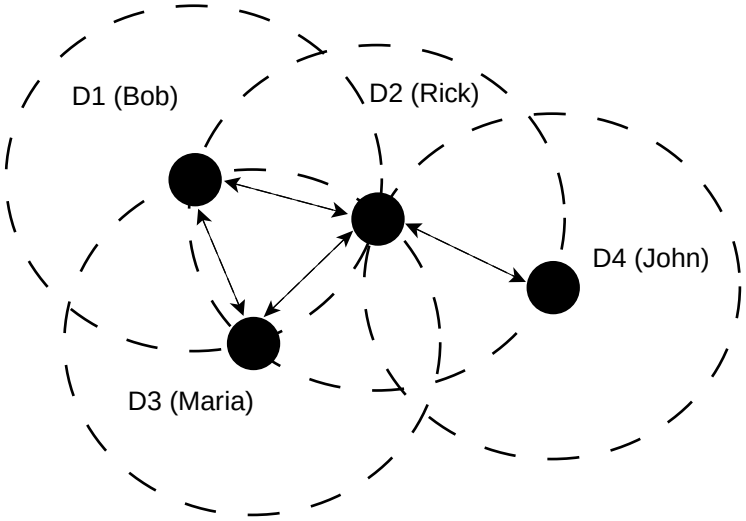


Fig. 2. A Sample Testbed Configuration

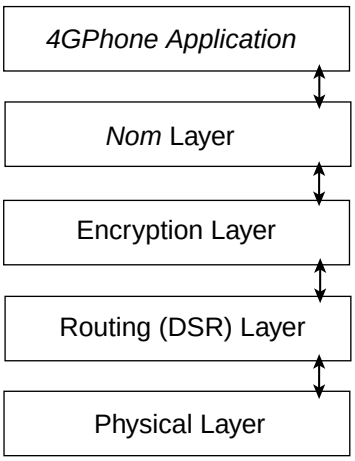


Fig. 3. The NTRG Layer Configuration

log box for who he wants to call, and the mapping will happen automatically. The application can then make a function call to the resource location layer with the string "John" as a query, and waits for the *Nom* layer to return the result. In this way, the name John (the owner of the machine) is dynamically mapped to the physical node ID for the device, in this case D4. With the physical address resolved, the ad hoc routing protocol (DSR) is now able to establish a route to that destination. The actual voice communication can then begin without the need for centralized management of mappings or configurations.

## VIII. Conclusions and Future Work

Today's variety of mobile network implementations implies a lack of standard for resource location system, and there is no way to route resource location or autoconfiguration queries between different networks or between networks that are physically beyond each other's range even if those networks had connectivity to other network with more reach (i.e., the Internet).

We have described a system that provides resource location in a simple, platform-independent way and that fulfills the special requirements that mobile networks have, namely, support for mobile and highly dynamic topologies, exact results even when confronted with network changes (i.e., a query will be satisfied if a path exists between the origin and the target during query-cycle time), and independence of routing protocols. In particular, ad hoc wireless networks, that don't rely on servers, are also supported. We have also created a test implementation of the algorithm that lays the groundwork to expand our research in other directions.

In particular, we are currently working on providing other message-conversion "gateways" that will enable routing between different network types and between ad hoc networks that are out of physical radio range, but that have connection to the Internet, as mentioned previously. We plan to use the Gnutella network as an existing platform on which to route queries between remote networks, to provide, at least initially, information regarding the location of the resource that is being searched. Additionally, we plan to conduct more tests, looking for new ways to improve the performance of the system while maintaining its generality and portability.

We are also working on improving *Nom*'s security, to ensure that query responses are not tampered with along the way, and that malicious users are not able to impersonate other nodes within the network.

Since in many cases the query involves a specific name (rather than a substring) we are investigating ways of integrating the basic *Nom* design with an Overlay Network, to provide fast access with low network resource usage.

Further ahead, we plan to extend *Nom* to other domains where physical to logical mapping is required for highly dynamic, mobile networks.

## References

[1] R. Droms. *Dynamic Host Configuration Protocol, RFC 1541*. IETF Network Working Group, November 1997.

[2] I. Clarke et.al. Designing privacy enhancing technologies: International workshop on design issues in anonymity and unobservability. In H. Federrath, editor, *LNCS 2009*. Springer: New York, 2001.

[3] Jinyang Li et.al. Capacity of ad hoc wireless networks. In *Proceedings of ACM Sigmobile*, pages 61 – 69. ACM Press New York, NY, USA, 2001.

[4] Juan Flynn et. al. A real time emulation system for ad-hoc networks. In *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference 2002 (CNDS '02)*, San Antonio, Texas, January 2002.

[5] Nelson Minar et. al. The swarm simulation system: A toolkit for building multi-agent systems. Technical Report 96-06-042, The Santa Fe Institute, June 1996.

[6] S. Ni et. al. The broadcast storm problem in a mobile ad-hoc network. In *Proceedings of the fifth annual ACM/IEEE international conference on Mobile computing and networking*, August 1999.

[7] Gnutella protocol v0.4, 2001. http://www9.limewire.com/developer/gnutella_protocol_0.4.pdf.

[8] David B. Johnson and David A. Maltz. *Dynamic Source Routing in Ad Hoc Wireless Networks. In Mobile Computing*, chapter 5, pages 153–181. Kluwer Academic Publishers, 1996.

[9] P. Mockapetris and K. J. Dunlap. Development of the domain name system. *ACM SIGCOMM Computer Communication Review*, 18(4):123–133, August 1988.

[10] et. al. M. Handley. *SIP: Session Initiation Protocol*. IETF Network Working Group, March 1999.

[11] D. O'Mahony and L. Doyle. Architectural imperatives for 4th generation ip-based mobile networks. In *Fourth international symposium on wireless personal multimedia communications, Aalborg, Denmark*, September 2001.

[12] D. O'Mahony and L. Doyle. *Mobile Computing: Implementing Pervasive Information and Communication Technologies*, chapter An Adaptable Node Architecture for Future Wireless Networks. Kluwer series in Interfaces in OR/CS. Kluwer Publishing, August 2001.

[13] Andy Oram, editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly, March 2001.

[14] C. Perkins. *IP mobility support, RFC 2002*. IETF Mobile IP Working Group, October 1996.

[15] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. Technical Report TR-00-010, Berkeley, CA, 2000.

[16] S. Rose. *DNS Security Document Roadmap, RFC 1541*. IETF DNEXT Working Group, November 2001. http://www.ietf.org/internet-drafts/draft-ietf-dnsext-dnssec-roadmap-05.txt.

[17] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, pages 329–350, Heidelberg, Germany, November 2001.

[18] Kunwadee Sripanidkulchai. The popularity of gnutella queries and its implications on scalability. February 2001.

[19] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of ACM SIGCOMM*. ACM SIGCOMM, August 2001.