Resource Discovery Protocol for Mobile Computing

Charlie Perkins
IBM T.J.Watson Research Center

Harry Harjono
Columbia University Computer Science Dept.

Contents

Resource Discovery Protocol for Mobile Computing

Charlie Perkins and Harry Harjono
IBM T.J.Watson Research Center
30 Saw Mill River Road (Route 9A), Hawthorne, NY 10532, USA.
Telephone: 914-784-7350.
email: perk@watson.ibm.com, harjono@cs.columbia.edu

Abstract
The increasing complexity of modern networks prompts a need for dynamic resource discovery. Mobile clients have the additional need to rediscover the location of local area network resources each time they move to a different LAN. We present a protocol and proposal for the operation of dynamic resource discovery. Our design is simple, extensible, and light weight. We implemented and tested our design with stationary servers, and mobile clients running mobile IP.

Keywords
resource discovery protocol, light weight, service location, directory service, search engine, mobile IP, mobile computing, dynamic configuration, URN, URL, DHCP, RDP, SLP

1 INTRODUCTION

Mobile computing, mobile networking protocols, and the growth of the Internet are combining to make today's mobile computer users feel like global citizens. However, in order to make full use of the Internet, today's citizen need to be able to make numerous configuration choices. Recent efforts, (Droms (1993), Perkins and Tangirala (1995), Perkins (1995)) have begun to chip away at the requirement for reconfiguration of mobile computers as they move from place to place, but there is still much to do.

Access to local computing resources is usually required to sustain the productivity of mobile users, and the network connection to those resources is accomplished with recently developed mobile networking protocols, IETF Mobile-IP Working Group (1995). However, up until now, there hasn't been an easy way for Internet users to find and use the local computing resources without making phone calls, and then reading manuals about how to perform the necessary configuration operations. That often means finding out which of hundreds or thousands of system files need modifications. Worse, as more and more users are faced with the need to perform these administrative functions, it is inevitable that some mistakes will be made. Determining the cause of errors in the course of performing system configuration is often very unpleasant; if network configuration is involved, errors in the configuration of a new system can cause an entire network or subnet to go out of service.

Our resource discovery protocol is designed to alleviate the problem of finding and using computer resources that are located external to a mobile computer. The protocol itself doesn't depend upon mobility, but on the other hand it is especially when a computer becomes mobile that its need for system reconfiguration is drastically increased, compared to computers which are installed in one place and remain there for a long time. Even so, once the resource discovery algorithms become commonplace, we expect that they will become a natural part of the overall organization of most Internet computers.

2 UNIFORM RESOURCE LOCATORS

The recent growth of the World Wide Web has occasioned the even faster growth of the Internet. The utility of the Web is due to the ease with which even beginning computer users can browse untold gigabytes of interesting pictures, stories, games, and other programs and data. These computer resources are delivered to the browser after it finds them by following a *Uniform Resource Locator*, or URL (T. Berners-Lee and L. Masinter and M. McCahill (1994)) for short. URLs are a standardized way to locating and providing access to a large variety of computer resources located on the network. They have the general form `<scheme>:<scheme-specific-part>`, where a *scheme* is just a string which tells how to understand the *scheme-specific-part.*

URLs are made available to browsers by user selection from stylized menus, which may contain buttons, maps, or other indicators. User interaction, however, is exactly what we would like to minimize or eliminate in the configuration of mobile computers. It is possible to imagine some sort of "mobile computing butler" which interrupts the user upon any indication of resource outage, and presents a menu of newly available resources which can take the place of the resource which is no longer around. Instead, we wanted a way to discover and use Internet resources without any user interaction whatsoever. For instance, if a computer is turned on in unfamiliar surroundings, all necessary network configuration details should be acquired automatically and put into service without user intervention.

The main difference between delivering network resources to Web browsers, and the more automatic way of discovering access to resources just described, is that in the latter case there is no user interaction to specify or "name" the indicated resources. What is needed is not only a URL, but also a URN, or Uniform Resource Name (Paul E. Hoffman and Daniel, Jr., Ron (1995), K. Sollins and L. Masinter (1994)). Then, a computer could acquire the resources it needs to operate by formulating a list of URNs, and then request a URL for each of them. This is exactly the approach we took. However, URNs are not in common use with the Web, and the exact meaning of URNs is still a matter of dispute within the Web technical community. Even so, an evaluation of existing URN proposals, (Mark Madsen (1995)), pointed out that the OCLC scheme, (K. Shafer and E. Miller and V. Tkac and S. Weibel (1995)), is the most promising framework from the point of view of extensibility and future-proofness. Accordingly, we have adopted that scheme in our work.

Aspects of URNs that may be expected if consensus emerges are:

- resource names will be specified with a syntax conforming to URI (T. Berners-Lee (1993)),
- resolution of URNs will exhibit a high degree of location independence,
- URNs will be well suited for identifying and locating particular documents and versions of documents within the global Internet.

For resource discovery, we decided to make use of the possibility that URNs might resolve differently depending upon which agent was doing the resolution. This can be seen as a violation of the spirit of URNs, because we explicitly want different URLs to be associated with the same URN depending upon where the resolution occurs. Given the proposed deployment of URNs as document identifiers, one could well argue that a URN was expected to always resolve to the same document URL. Although we like URNs, our needs for resource discovery have almost nothing to do with document retrieval. Moreover, we optimistically hope that our use of URNs will influence the future direction and standardization of URNs for specifying resource location, since if the proposed Service Location Protocol succeeds in its current form it will be a motivation for the further development of URN technology and protocols.

3 PROTOCOL CHARACTERISTICS

Our intention is to make a resource discovery protocol suitable for automatic operation in sometimes crowded enterprise internetworks, to serve the needs of mobile clients. To do this, we had to adhere to a number of design requirements. The protocol is required to be

- scalable
- self-managing
- distributed, with numerous servers
- compatible with other administrative tools
- compatible with mobile networking protocols

The need for scalability is almost a given in today's Internet. Any protocol which only works well with a few computers on a network will not pass the review within the Internet Engineering Task Force (IETF), so would have no hope for standardization. Since, from our perspective, the point of creating the protocol in the first place is to eliminate the need for user configuration, the protocol must require zero user administration. As a natural consequence of the requirement for scalability, we must also demand that minimal or non-existent configuration be required for even the servers which provide the resource data for the mobile clients. Any burdensome administrative requirement for human intervention or control of the resource discovery servers will be doomed to failure as the Internet continues to provide an ever greater array of possible services to mobile clients.

DHCP (Dynamic Host Configuration Protocol, S. Alexander and R. Droms (1993), Ralph Droms (1993)) servers are one particular administrative tool with which we had to be compatible. In fact, the ability of a DHCP server to configure its client with the address of a Resource Discovery server is perfect for our needs, and shows that the original designers of DHCP looked forward to the day when such protocols as ours would become available.

Besides the above mentioned protocol requirements, we intended to produce a protocol with some additional highly desirable properties:

- lightweight (fast)
- string-based (simple parser)
- easily implemented
- use existing standards where possible
- syntactically flexible

Since our main protocol operation is to supply a pointer to a named resource, we made it one of our main goals to keep the protocol lightweight. We expect that this will go a long way towards enabling the widespread deployment of the protocol for mobile computers. Our definition of lightweight also includes minimal network cost, so that broadcasts and extended negotiations for resources are considered highly undesirable.

The network of the future is likely to be populated with resources and agencies we can only dimly imagine. These resources, although we can't name them now, will certainly have names, and their names could be used as part of a URN. We already had parsers for URLs that work with Web browser software, and those parsers could be made to work well with human readable resource names in the form of URNs. The combination of existing algorithms, flexibility, human readability, and extensibility for the future makes string-based operation quite attractive.

An alternative approach might be to assign numbers for each new resource type, and make the clients request resources by number. However, that approach relies completely on the required registration of new resources with an Internet arbiter such as IANA (Douglas E. Comer (1991)) as well as the timely dissemination of newly registered resource numbers to all interested resource discovery servers. This is fine for resources that are duly registered with IANA, but not so fine for resources that are still experimental, vendor-specific, or site-specific.

Although the string-based approach to naming also requires a conventional agreement between client and server regarding the names of resource, this agreement is more likely with strings in the abovementioned latter cases. For instance, it's a lot easier for the word "printer" to proliferate throughout the administrative and engineering community at a particular site than some arbitrary (numeric) bit string. Thus, we consider strings to be the most obvious candidate for specifying attributes or keywords for selecting among numerous resources of the same general "type", and we settled on the use of strings for naming resources. This was another motivation for our subsequent use of URNs.

Lastly, we explicitly favored ways to re-use existing protocols and language syntax, in the belief that new development is usually better and almost always faster if it uses the hard work of other people. Not only did we have to make fewer decisions about code structure and query format, but we also have been able to avoid all the mistakes that were probably made in the early design of Web protocols. This is, of course, another benefit of aiming our design towards compatibility with the World Wide Web as a collection of resources and resource locators.

4 RESOURCE DISCOVERY PROTOCOL (RDP)

4.1 Introduction

The main objective of RDP is provide a light-weight protocol which a client can use to discover network resources. It is especially targeted for a mobile client whose environment may change often. In RDP, we assume that the client has a means of obtaining the address of the RDP server, either statically via configuration file, or dynamically via DHCP. Of course mobile clients are unlikely to have any static configuration for the address of an RDP server, but stationary enterprise desktops likely would.

Mobile clients are frequently wireless, and wireless stations currently have characteristically poor interactions with TCP when the wireless medium is suffering from a high bit-error rate (BER) (Ramon Caceres and Liviu Iftode (1995)). We wanted to avoid interactions between our protocol and the timeout characteristics of TCP, to keep RDP as light-weight as possible. Thus, we rely on UDP for packet delivery. For simple query/response case, the data can fit into one UDP datagram. When the data is too big to fit into one datagram, it is broken into multiple UDP packets.

The basic operation follows a client-server query-response model. The client queries the server using a URN query; and the server replies with one or more URLs to satisfy the query. The client may then proceed to use the returned URL(s) to access the network resources. In addition to this, RDP also supports dynamic registration and deregistration of network resources which enables the server to manage the resources automatically.

4.2 RDP Database

The database is very simple. It consists of a collection of records, each with the following structure:

```
<resource URL>
        <description1>
        <description2>
        ...
        <descriptionN>
```

The description lines contain descriptions of the resource, which may consist of multiple keywords with optional attribute names. No syntactic structure is imposed on the descriptions. For example:

```
printer://dukprunz.watson.ibm.com/j1j25ps
        name=j1j25ps
        location=j1-j25,j1j25
        queues=j1plain,j1color,j1foils
        os2 postscript personal printer color foils
        access via TCP/IP lpd lpr
        local=129.34.16/24,9.2.46.0/25
```

Notice that the example above uses a non-standard printer URL*. In this paper, for simplicity we do not always append a subnet prefix length specification to relevant IP addresses, although it is done in this example. The details of managing printers and print queues may require enumerating additional parameters in the same way that is indicated here.

4.3 RDP Query and Response

To query the resource location, a client has to construct a valid URN query. The format of the query is:

```
        <service>:/[rp]/[na]/<scheme>/<key1>/<key2>/...
where:
        service = n2l or n2c
        rp      = resolution path
        na      = naming authority
        scheme  = URL scheme
        keyN    = keywords describing the URL
```

The format of the URN query is borrowed from the *URN Services* (K. Shafer and E. Miller and V. Tkac and S. Weibel (1995)) internet draft with some liberal changes to suit our purposes.

The service field specifies the desired type of resolution. *n2l* maps one URN to one URL: the server will return the first URL it find which satisfies the URN. *n2c* maps one URN to a list of URLs satisfying the URN†.

The resolution path is optionally specified by the client to direct the query to the desired RDP server. In the absence of a resolution path specification, the query will be sent to the server host returned by DHCP, if the client has requested option 11 from the local DHCP server (S. Alexander and R. Droms (1993)). Note that, in the usual case, the resolution path will just be the IP address of a nearby host computer. Also note that this procedure allows our resolution architecture to scale well.

The naming authority (*NA*) specifies the organizational entity which is authorized to

---

*The printer URL syntax is `printer://<lpd-hostname>/<queue-name>`.

†Other request services have been suggested along the lines of *n2two*, *n2three*, ...

resolve the query, and then by necessity the dictionary which is used to define the relationship between the scheme and the scheme-specific-part of a URL. The IANA naming authority already specifies some universally known schemes, including *printer*, *mailto*, *http*, and *nfs*. Upon receiving a URN query, the RDP server will verify that its naming authority is compatible with that of the query. If it does not, the query will be forwarded to the authorized server. When the naming authority is omitted, the server can skip the verification. This is useful for wandering mobile clients which do not know the naming authority of their local network.

The requested *scheme* is found in the field after the naming authority field. It can be any valid URL scheme recognizable by the server. When the resource database grows large, the resource server may partition the database into disjoint subsets based on the scheme, since the same URL cannot possibly belong to two different schemes under the same naming authority.

The last components are the keywords used to search the database for the matches. A match is found when the scheme and *all* the keywords match. These keywords should form an intuitive description of the desired resources, and could be obtained directly from the end users if necessary. An efficient multi-keyword search algorithm is presented by Sun Wu and Udi Manber (1994). For large databases, the search can be made faster using a two-level indexing scheme described by Sun Wu and Udi Manber (1993) with minimal indexing time and space.

Some examples of valid URN queries are:

```
n2l://ibm/nfs/rdp/src
n2c://ibm/http/research/homepage
n2l:///printer/local/postscript
```

Here, *n2l* means to return only one URL in response to the query, and *n2c* means to return all matching URLs, concatenated in the reply. A high-function server could possibly sort matching URLs in order of decreasing expected usefulness to the client, based possibly on distance. The word *local* has a special meaning; it means that the returned URL should be local to the client (see Section 4.4).

Some examples of valid URL replies are:

```
nfs://slag.watson.ibm.com/src/rdp
http://www.research.ibm.com/
printer://dukprunz.watson.ibm.com/j1j25ps
```

An application using RDP must come equipped with some conventional, built-in vocabulary, in order to be able to send queries to the RDP server. For example, to find a list of local postscript printers, a word processing application should know how to construct the URN query

`n2c:///printer/local/postscript`.

This query could be hard-coded into an application as long as it did not change the type of printer needed. The resolution of the query may return different printer URLs depending on which is most beneficial for the mobile client in its current environment.

4.4 Locality

Locality is a tricky problem, which is at the same time intimately tied to user convenience, but also difficult to define in a way that is useful for all contexts. For instance, a user might be interested in *geographical* locality when it is time to select a printer, but locality with respect to *network topology* is more likely to be of interest for connection to resources with which large amounts of data will be transacted. Moreover, in certain situations a sort of administrative locality will be useful. Geographic locality has received some attention, for instance in papers describing progress with active badge systems by Bill Schilit and Marvin Theimer (1994), and by Roy Want and Andy Hopper and Veronica Falcao and Jonathan Gibbons (1992).

Locality will *usually* be defined (at least partially) by the scheme. As just suggested, the `printer` scheme is more likely to evaluate locality based on the walking distance between the mobile client and the prospective printers. We do not make any attempt to refine the handling of the `local` keyword in this version of the protocol. However, we note that future versions may include `local` as a *scope* specifier, and for now only specify locality by describing subnet information. We expect that when printer URLs are registered, they will include keywords that specify which subnets are local. Clearly this handles some high percentage of current needs, while just as clearly there are many refinements to be made.

4.5 Resource Registration and Deregistration

Clients register and deregister network resources using the *reg* and *dereg* requests. The RDP database grows and shrinks accordingly. The format of the reg/dereg requests are:

```
<reg|dereg>:/[rp]/[na]/<url>;<desc1>[;desc2]...
```

where:

```
rp      = resolution path
na      = naming authority
url     = URL to be registered (or deregistered)
descN   = descriptions of the URL
```

Registration can be performed incrementally. The new URL record will be created only if it doesn't exist in the database. If the URL already exists, only the descriptions not in the database will be added. The final URL record contains the URL and the union of all the descriptions.

Similarly, deregistration can be performed incrementally. If no descriptions are specified, the whole URL record is deleted; otherwise, only the matching descriptions are deleted.

An acknowledgement is sent to the client for a successful reg/dereg. If no acknowledgement is received after the timeout, the client may retry the operation. Note that both the reg and dereg are idempotent; this is necessary to ensure the integrity of the database.

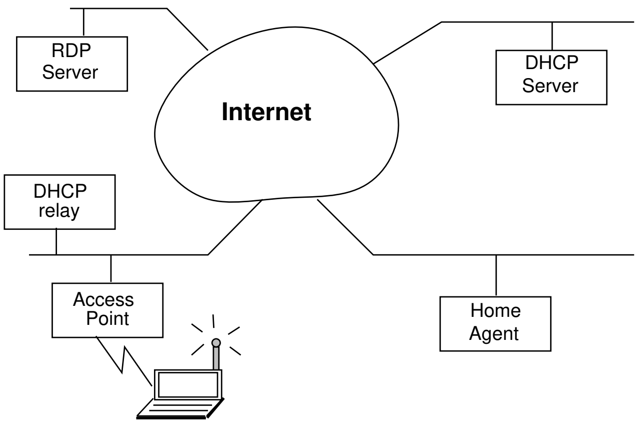

Figure 1 Mobile Client interacting with RDP and other Internet agents

5 MOBILE-IP, DHCP, AND RESOURCE DISCOVERY

Our motivation for investigating Resource Discovery comes from our conviction that it will be a requirement for the convenient operation of mobile computers. We also are convinced that such computers will use the mobile-IP (IETF Mobile-IP Working Group (1995)) protocols being advanced in the IETF. Both protocols are also intimately involved with the use of DHCP. As it turns out, it seems that there there is never conflict in the simultaneous use of the protocols of interest. This results from the fact that resource discovery is used by application level clients, whereas mobile-IP is used in the protocol stack itself, and DHCP is used for system and network configuration parameters. The low level of interaction we have observed validates the notion of modular design of network protocols. However, there is more to be said when a higher level view of the system operation is considered.

Consider the sequence of events when a mobile client first begins operation. In the most demanding case, the client will rely on DHCP for acquisition of both its *own* address (known as a *home address* (Charles Perkins and Jagannadh Tangirala (1995), and Charles E. Perkins (1995)), as well as a care-of address. Note that the RDP server, the DHCP server, the home agent, and the subnet to which the mobile client is attaching via an access point may all be on different subnets. The following might be a typical scenario during the time when a mobile client reboots:

- Mobile queries DHCP for a home address
- Mobile discovers that a mobile-IP registration is needed (if it is not attaching to its home network)
- Mobile queries DHCP for a local IP address to be used as a Care-of Address, including Resource Discovery server option
- Mobile registers the new Care-of Address with home agent
- Mobile queries the Resource Discovery server for needed resources

Note that in this case, DHCP is queried twice, and there is no need to add option 11 for the RDP server[‡] the first time. Moreover, notice that the RDP could reasonably be consulted for all possible resources, on the assumption that things may have changed significantly since the last time the mobile computer has been restarted.

Consider another case. If the mobile node registers with a Care-of Address which is advertised by a nearby foreign agent, there is no immediate need to contact a DHCP server, as long as no network resources were required during the mobile node's stay in the area being served by the new foreign agent.

A system designer could reasonably consider whether it was worthwhile to request new resource pointers for only *local* resources, instead of contacting the RDP server for every resource it might need. It is also likely that the request for new resource locations should be performed on demand instead of upon every cell switch, on the assumption that some resources will not be accessed before another cell switch. There would be no point to resolving new resources when the results might never be used.

We do not offer any conclusion about the best system design for when the Resource Discovery Server should be contacted. So far, our approach has been only to resolve needed resources when the mobile computer reboots. We plan to attack the problems introduced by cell switches after we install some ability for interprocess communications to be triggered by cell switches.

6 IMPLEMENTATION EXPERIENCES

The current implementation of RDP client and server are available in C and C++. We have compiled them for AIX, OS/2, Linux, and SunOS. We expect any BSD socket compliant system should have no problem compiling it. This is a quick implementation, and is not optimal. We just wanted to show the feasibility of the system.

6.1 RDP Client

On startup, the client host gets the address of the RDP server from DHCP, which it then saves in a permanent, conventionally known file. Applications which need to contact the RDP server get the address from this file. This step can be automated by providing a Resource Discovery API (Applications Programming Interface) so that the application does not need to statically configure the filename.

The generic RDP client program simply takes a string from the user or the command line, verifies that it is a valid URN format, and sends it to the RDP server. It then waits and displays the URL reply. The client also keeps a retransmit timer to simulate reliable packet delivery. We use this generic client program in our OS/2 REXX script to communicate with the RDP server.

[‡] Also known as Resource Location Server (RLS) option in the DHCP document (S. Alexander and R. Droms (1993))

6.2 RDP Server

On startup, the RDP server may read an initial list of resources from a configuration file. From then on, the resource database may be changed by registration and deregistration requests.

To serve a URN query, our server verifies the naming authority, and checks the validity of URN format. Then, it does a linear search through its database for matching keywords.

6.3 Mobile-IP testbed

We tested our system with stationary server and mobile clients. The resource database is populated with URL records of local resources such as printers, and NFS mount points. The mobile client obtains the address of the RDP server from DHCP during bootup. In figure 1, the mobile client performs the following steps:

- gets care-of address and address of stationary RDP server from stationary DHCP server via access point and DHCP relay
- registers new care-of address with Home Agent, allowing delivery of packets to the mobile client from anywhere on the Internet.
- contacts a stationary RDP server for each network resource needed

Given the setup depicted in figure 1, we showed that the mobile clients can access local network resources such as printers and NFS filesystems. Since our mobile client runs OS/2, we wrote some REXX scripts to query the RDP server for local resources, and then proceed to access them.

Note that, in the above situation, our mobile client accesses the RDP server at boot time and performs the necessary operations for all possible network resources at that time only. We expect that it will be much more common to access the RDP as the individual resources are needed, possibly after the mobile client has been in operation for quite a while. Moreover, the mobile client will need to contact a (possibly different) RDP server after it moves to a new access point. We haven't tested that operation yet.

Note also, in the following descriptions, that the Internet agents involved usually return the Internet addresses of target hosts, not their human-readable fully-qualified domain names. We use the domain name to make the examples easier to understand. In most cases it is preferable to allow the receiving host to avoid the extra step of having to resolve the domain name into an IP address.

Accessing a local printer is very straightforward (but see Section 4.4). We send a URN query for local postscript printer, and use the returned printer URL in the *lpr* command to send data to be printed. Figure 2 shows the following scenario:

- The wireless mobile client first gets access to a nearby RDP server by querying a DHCP server (usually via a DHCP relay, and physically by way of a wireless access point).
- The DHCP server returns the IP address of the RDP server (muffin.watson.ibm.com), which is on another subnet.
- The mobile client contacts *muffin* to get the address of the local printer service. The URN (namely, n2l:///printer/local/postscript) is delivered to the RDP server (muffin.watson.ibm.com).

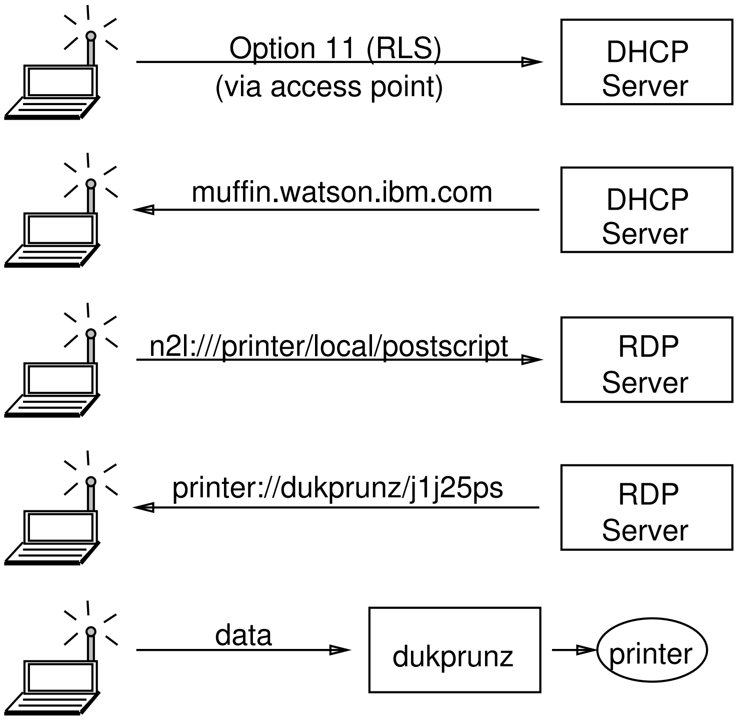

Figure 2 Discovering a Print Service

- *muffin* returns the necessary URL (printer://dukprunz/j1j25ps) to the mobile client.
- Finally, the client is able to access the printer by specifying the correct destination printer address along with the "lpr" command (on our OS/2 and AIX systems).

Notice in this case, the URN is delivered (possibly by use of intervening Internet routers) by unicast to muffin.watson.ibm.com, and no resolution path needs to be specified.

Similarly, we can mount a local NFS filesystem by sending the URN query for a local NFS filesystem, and use the return NFS URL to mount the filesystem. Accessing the NFS filesystem is a bit more work because we have to set some environment variables such as (for OS/2) the PATH, LIBPATH, etc. before the filesystem can be conveniently accessed by common applications. We do this by calling an initialization routine from the mounted filesystem. This is necessary because different filesystems require different initialization. Unfortunately, the environment variable initialization can only affect the newly created shells. Currently, there is no mechanism available to change the environment variable of existing processes – and changing environment variables would be a very tricky and error-prone operation in any case.

There is a little twist in defining the locality of mobile host running mobile IP. Using the `local` keyword will not work since the mobile host resides in its own virtual network, and is not in the same subnet as the local resources. We assume that the local resources are attached to stationary wired network. The only hint we have about the locality of the

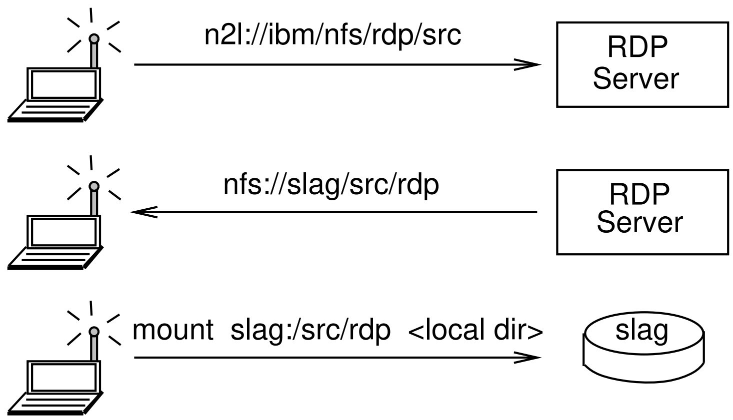

Figure 3 Mounting a Conventionally named NFS Filesystem

mobile host is gleaned from the default routing entry set by the mobile-IP daemon. This entry shows the association of the mobile host to its foreign agent at the wired network. Thus, in order to locate local resources, we have to include a local router (for instance, the care-of address of the foreign agent) in the query keywords.

7 COMPARISON

There has been a lot of work on resource discovery on the Internet. Various tools have been devised for collecting resource information. The tools were created to fill certain needs, and they offer valuable services to the Internet community. In this section, we compare RDP to some existing resource discovery tools. This is not meant to be an exhaustive comparison, but just to give some idea where RDP fits.

7.1 Web Search Engines

Perhaps the most commonly accessed resources in the Web are the search engines (C. Mic Bowman and Peter B. Danzig and Darrenn R. Hardy and Udi Manber and Michael F. Schwartz and Duane P. Wessels (1994), Oliver A. McBryan (1994), Brian Pinkerton (1994)). They collect information from the World Wide Web and condense it into fast searchable indexes. A typical user would enter some keywords and some search criteria into the search engines, and would receive in return a list all the Web resources satisfying the search parameters.

The operation of RDP is similar to some extent, except that RDP server doesn't collect information from the Web. Instead, the client has to explicitly register the URL to the appropriate server as specified by its naming authority. This dynamic registration (and deregistration) is required because RDP will have to deal with a larger, and currently unknown, set of resources, currently including printers and filesystems. For such active resources, the information cannot be collected by traversing the Web. Also, there is a need to update the resource attributes for active resources on demand. Moreover, an RDP server can offer access to resources which are different in kind than what is usually considered to reside on the Web. For instance, a fax machine isn't usually indicated by a button on someone's home page.

For passive resources, the information can be collected manually or using some of the existing collection techniques, and be registered to the RDP database. The RDP database is distributed and partitioned based on the naming authority. The naming authority is usually determined by the organizational entity. Each RDP database is localized, and mainly contains information about its local resources. This distribution is required for RDP scalability.

7.2 Connectionless Lightweight Directory Access Protocol (CLDAP)

CLDAP (A. Young (1995)) is a UDP-based light-weight version of its LDAP and DAP counterparts. The main motivation of CLDAP is to provide access to the X.500 directory service without incurring the full cost of DAP. So, CLDAP functions as a light-weight X.500 directory service front-end for simple applications.

CLDAP is based directly on LDAP, with the differences that CLDAP uses UDP and has a restricted set of operations. CLDAP clients should use a retry mechanism with timeout in order to achieve the desired level of reliability. Only one request may be sent in a single datagram, and only one response may be sent in a single datagram.

While similar to RDP in many ways in its use of UDP, some differences are noted here. CLDAP uses X.500 attribute encoding, whereas RDP uses URN and URL encoding. CLDAP uses X.500 DS, whereas RDP currently uses its own simpler database subsystem.

7.3 Service Location Protocol

The service location protocol (SLP, John Veizades and Scott Kaplan and Erik Guttman (1995)) provides a framework for the discovery and selection of local network services. RDP provides similar facility for discovering resources anywhere on the internet. SLP relies on the multicast support at the network layer; it uses multicast request with unicast response. RDP only relies on UDP, and uses unicast request and response.

Both SLP and RDP allow dynamic registration and deregistration of resources. Currently RDP doesn't have the request message for returning the URL description information equivalent to the attribute request in SLP. This feature will be added in the future to enable browsing the resources.

In terms of data encoding, SLP uses character strings represented as character strings are represented as a type,length,value tuple; and RDP uses character strings conforming to URI (T. Berners-Lee (1994)) specification. For constructing the query, SLP defines its own predicate language which is based on attributes. RDP, on the other hand, uses keywords embedded in the URN query which is far more flexible, and extensible.

Lastly, the SCOPE mechanism used in SLP is meant for use on a single LAN; whereas the naming authority in RDP will scale to a larger network. The use of URN in a RDP query also enable clients to directly specify the resolution path in the query.

We have begun to work with the IETF working group to make modifications to the existing Service Location Protocol. Some of the ideas of RDP have already been assimilated into newer Internet drafts from the working group, having the effect of narrowing the differences between our approaches. We expect the differences between our approaches to eventually disappear.

8 FUTURE WORK

The future surely holds great promise for the development of protocols to automatically locate network resources. In many ways, we have only begun to scratch the surface.

The most immediate direction for further work is to define more resource schemes. Many are possible – for instance, we could use RDP with:

- a white-pages service
- a front end for X.500 queries
- Application-specific libraries
- Local parts databases

Multiple RDP servers should be able to collaborate to provide to their clients the advantage of their combined resource databases. Note that such server-server protocols are difficult at best, and we do not believe that RDP will be an easy case. When RDP servers can cooperate, a query presented to one server might easily be forwarded by that server to another collaborative server and be resolved without undue delay to the client. The manner in which queries are presented to the RDP servers may change whenever a preferred format for URNs is defined within the IETF.

We have not designed any security mechanisms for RDP. Security is not even available with DHCP yet; when the world gets to the point of needing security bad enough with DHCP, we expect to be able to import whatever DHCP has into our work. Access controls may have to be put into place which prevent the discovery of the resources by unauthorized clients. Enabling the secure and perhaps even confidential registration and deregistration of network services is of particular importance. For this purpose, we may reasonably employ recent RFCs for authentication and encryption (Randall Atkinson (1995, 1996)).

Normally, it is expected that access control for most resources will be managed by the agent located by the URL, and not necessarily by the resource discovery server itself.

Applications which rely on environment variables for network configuration options cannot always be expected to work well with Resource Discovery. In order to improve them, we would require operating system support to enable a global change of environment variables for running processes. That seems quite unlikely to happen in the near future.

We expect to specify an applications program interface (API) that would enable applications to easily make use of our protocol. Moreover, this interface should allow applications to do the right thing when their previously obtained URLs for necessary resources are likely to have become invalid.

Last, but not least, we would like to investigate the problems of resource discovery in ad-hoc environments. The convenience of resource discovery may cause mobile nodes (and their applications) to come to rely on resource discovery in enterprise environments rich in computer resources. Applications may be written that rely on the convenience of registering arbitrary and privately named resources at the local Resource Discovery Server. This very powerful feature should then be also available when there isn't any infrastructure, and one of the mobile nodes has to be elected to perform the resource discovery function for the local population of ad-hoc mobile stations. The exact mechanisms by which this might occur will be a fascinating research area.

9 CONCLUSIONS

Obtaining a Resource Discovery Server from DHCP fits naturally within the framework provided by mobile-IP and DHCP. When the mobile node needs to access a network resource in a particular area, it no longer needs to rely on static configuration data that can be valid in only one location. Instead, it contacts the local server to fulfill its needs for the location of nearby resources.

By using keywords, our Resource Discovery Protocol provides a flexible way to get access to dynamic resource information. Since we can only start to categorize and name the kinds of resources that are available on the network, we should avoid any premature attempt to place the resource names into a rigidly controlled system using registered numbers, or sequences of $attribute = value$ pairs.

Allowing the dynamic and essentially uncontrolled registration of resources with a local Resource Discovery Protocol server may have big implications for the future extensibility of our protocol. We look forward to experimenting with such facilities, and note here that it would be much more difficult to design such systems if unnecessary structure were placed on the format of the resource specification (and selection) packets.

We feel strongly that URL strings should be used as the resource discovery medium for communicating resource location information, since that is their precise purpose. We believe that it is important to take advantage of existing work contributing to the dominant success of the World-Wide Web. Thus we claim that it is also important to take whatever lessons we can take from existing work on naming resources within the Web, and consequently our design of the resource query was modeled after URNs.

It is also important to attend to details like minimizing the number of broadcast packets, scalability, protocol complexity, and implementability. We feel that our approach is a successful attempt to balance the needs and engineering tradeoffs required for the protocol, and our ability to produce a working system within three months after starting is a solid indication that our protocol can indeed be successfully implemented and made available for use within the Internet.

10 ACKNOWLEDGEMENT

Thanks to Erik Guttman for stimulating conversations about RDP and the IETF protocol, and for his willingness to incorporate some of our ideas into the IETF draft specification for the Service Location Protocol.

REFERENCES

S. Alexander and R. Droms. DHCP Options and BOOTP Vendor Extensions. RFC 1533, October 1993.

T. Berners-Lee. Universal Resource Identifiers in WWW. RFC 1630, November 1994.

T. Berners-Lee, L. Masinter, and M. McCahill. Uniform Resource Locators (URL). RFC 1738, December 1994.

C. Mic Bowman, Peter B. Danzig, Darrenn R. Hardy, Udi Manber, Michael F. Schwartz, and Duane P. Wessels. Harvest: A Scalable, Customizable Discovery and Access System. Technical report, University of Colorado, Computer Science Department, University of Colorado, Boulder, August 1994. CU-CS-732-94, revised March 1995.

Douglas E. Comer. *Principles, Protocols, and Architecture*, volume 1 of *Internetworking with TCP/IP.* Prentice Hall, Englewood Cliffs, N.J., second edition, 1991.

R. Droms. Dynamic Host Configuration Protocol. RFC 1541, October 1993.

IETF Mobile-IP Working Group. IPv4 Mobility Support. ietf-draft-mobileip-protocol-12.txt - work in progress, September 1995.

Paul E. Hoffman and Jr Ron Daniel. Generic URN Syntax. draft-ietf-uri-urn-syntax-00.txt, April 1995.

Mark Madsen. A Critique of Existing URN Proposals. draft-ietf-uri-urn-madsen-critique-00.txt, July 1995.

Udi Manber and Sun Wu. GLIMPSE: A Tool to Search Through Entire File Systems. Technical report, University of Arizona, Computer Science Department, Tucson, Arizona, October 1993. TR 93-34.

Oliver A. McBryan. GENVL and WWWW: Tools for Taming the Web. In *Proceedings of the First International World Wide Web Conference*, CERN, Geneva, May 1994.

Charles Perkins and Jagannadh Tangirala. DHCP for Mobile Networking with TCP/IP. In *Proceedings of IEEE International Symposium on Systems and Communications*, pages 255–261, June 1995.

Charles E. Perkins. DHCP Home Address Option. draft-perkins-homeaddr-dhcpopt-01.txt - work in progress, October 1995.

Brian Pinkerton. Finding What People Want: Experiences with the WebCrawler. In *Proceedings of the Second International WWW Conference '94: Mosaic and the Web*, Chicago, October 1994.

K. Shafer, E. Miller, V. Tkac, and S. Weibel. URN Services. draft-shafer-uri-urn-resolution-00.txt - work in progress, June 1995.

K. Sollins and L. Masinter. Functional Requirements for Uniform Resource Names. RFC 1737, December 1994.

John Veizades, Scott Kaplan, and Erik Guttman. Service Location Protocol. draft-ietf-svrloc-protocol-06.txt - work in progress, July 1995.

Sun Wu and Udi Manber. A Fast Algorithm for Multi-pattern Searching. Technical report, University of Arizona, May 1994. TR 94-17.

A. Young. Connection-less Lightweight Directory Access Protocol. RFC 1798, June 1995.

11 BIOGRAPHY

`Harry Harjono` (*harjono@cs.columbia.edu*) is a PhD student at the Columbia University Computer Science Department where he works as a graduate research assistant at the Mobile Computing Lab under the supervision of Dr. Dan Duchamp, his academic and research advisor. His research interests include dynamic resource discovery, mobile computing, and distributed operating systems. He received an M.S. in Computer Science from Columbia University in 1994.

`Charles Perkins` (*perk@watson.ibm.com*) is a research staff member at IBM T.J. Watson Research, investigating mobile and ad-hoc networking, resource discovery, and automatic configuration for mobile computers. He is serving as the document editor for the mobile-IP working group of the Internet Engineering Task Force (IETF), and serves on the editorial boards for ACM/IEEE Transactions on Networking, ACM Wireless Networks, and IEEE Personal Communications. Charles holds a B.A. in mathematics and a M.E.E. degree from Rice University, and a M.A. in mathematics from Columbia University.