# Enabling Subscriber-to-Subscriber Communication in an NBMA DSL Network

M. MacGregor, L. Stewart
Dept. of Computing Science, University of Alberta
Edmonton, Alberta, Canada

## Abstract

Residential broadband digital subscriber loop (DSL) networks are used for many applications, such as surfing the Web, ICQ and e-mail. One of the more performance-intensive applications is distributed gaming. Participants in an online game may subscribe to multiple ISP's, or they may share the same ISP. In the case where several gamers share the same ISP, they will likely all reside in the same non-broadcast multiple access (NBMA) network. In this situation, manual configuration of each subscriber's machine is often required for them to communicate. In this paper we detail a method that removes the need for manual intervention, and enables communications between machines on the same NBMA network.

**Keywords**: ARP, proxy ARP

## 1. INTRODUCTION

In this paper we document a method for enabling communications between subscribers in different premises connected by the non-broadcast multiple access (NBMA) network of an Internet services provider (ISP). The environment we discuss is typical of those used to provide high-speed digital subscriber link (DSL) services. A typical DSL network (see Figure 1) concentrates traffic from hundreds of subscribers at a DSLAM. The aggregate traffic is carried over a 45 Mbps DS-3 link from the DSLAM to an ATM edge switch. The traffic from several DSLAMs is aggregated at the edge switch and carried over a 155 Mbps optical OC-3 link to the ISP's core switch. The core switch passes all the DSL traffic to a device which is a combined bridge / router. This last device is configured to bridge at layer 2 whenever possible, and only resorts to routing when it must.

This network is called a "non-broadcast" multi-access (NBMA) network because broadcast traffic originated by a subscriber is not rebroadcast out the same physical interface on which it arrives at the ISP. This is the bridge / router interface labeled "1" in Figure 1. This single physical interface may carry thousands of subscriber virtual circuits so that a misbehaving client PC could flood the network with broadcasts if all broadcasts were echoed at interface "1". One consequence of NBMA behavior is that ARP requests from subscriber machines are terminated at the ISP. This is problematic for ISP's whose subscribers want to do such apparently simple things as running multiplayer games over the network.

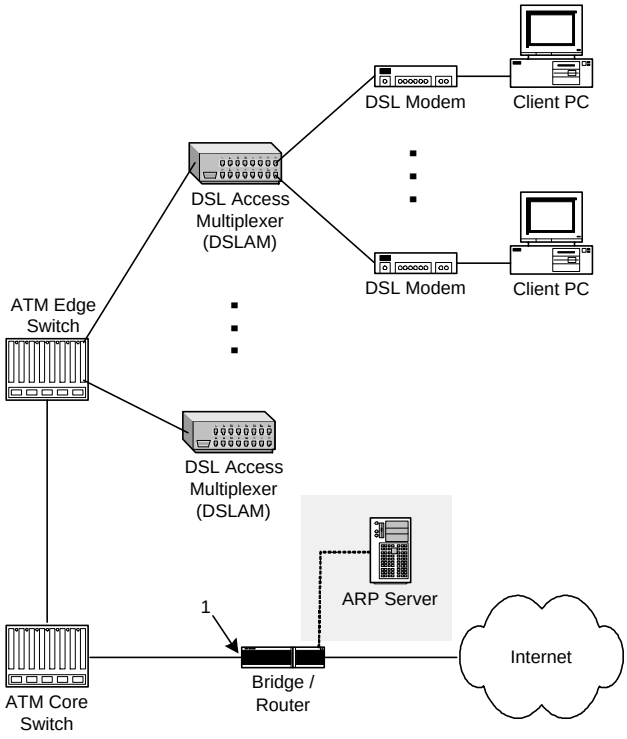


Figure 1. DSL Network

The ISP's network is usually configured at the bridge / router so that all subscribers are in the same IP subnet. This allows traffic between subscribers, and traffic incoming from the Internet to be bridged at layer 2 rather than routed at layer 3. Bridging is much less costly in terms of machine resources because the typical filter / forward / flood decisions can be implemented in terms of looking for an exact match in a table. In contrast, routing is based on a much more expensive "longest match" algorithm. If the subscribers were routed rather than bridged, then all incoming traffic (e.g. Web browsing) would have to be routed to each of the potentially thousands of subscribers. This would be incredibly wasteful of router cycles as all those subscribers are actually connected to the same physical interface, although using different virtual interfaces.

When one subscriber's machine wishes to communicate with another on the same subnet, it sends out an ARP

request. The ARP request is sent as broadcast traffic, and in an NBMA network it will be terminated at the ISP. The target will not hear the ``broadcast", and the requesting client will not get an ARP reply.

The ARP broadcast problem could be solved by placing each subscriber premise in a separate subnet, but this is not feasible as it would require routing at the ISP. Proxy ARP, in which the bridge / router would reply with its own MAC address, is not feasible either because this would result in all inter-subscriber traffic being routed because the bridge / router would see a frame with its MAC address, but an IP address belonging to another device.

ISP staff could walk each subscriber through the process of adding static ARP entries for each of their friends' machines to their PC, but this is very expensive in terms of support staff time. In fact, the single most critical factor in an ISP's profitability is its ability to minimize time on the phone with subscribers. Further, dynamic address assignment through DHCP is common in these networks, so subscribers will have to be prepared to update their static entries whenever a friend's address changes.

To solve this problem for a particular ISP, we began the process of developing a server running on a PC attached to a dedicated interface on the bridge / router. Our focus was on enabling subscriber-to-subscriber communications while relying as much as possible on the existing functionality in the network, and minimizing any manual configuration. In particular, we wanted to keep the PC out of the series path of normal communication, and minimize the functionality embedded in the server.

We describe our work in the following, beginning with a review of previous work. This is followed by a description of the tests we performed at the ISP's premises. After describing our solution we present the performance of the server in order to address scalability concerns. The paper concludes with a summary and suggestions for future work.

## 2. PREVIOUS WORK

The original specification of ARP appears in RFC-826 [1] ``An Ethernet Address Resolution Protocol." This RFC states explicitly that ``hosts do not transmit information about anyone other than themselves," so that the application outlined here is outside the original specification for ARP.

The most closely related work is described in RFC-1735 [2], ``NBMA Address Resolution Protocol (NARP)," which deals with address resolution in NBMA networks. The focus of RFC-1735 is on resolving addresses belonging to foreign subnets within the same NBMA cloud. If the destination belongs to the same subnet, RFC-1735 specifies that ARP or preconfigured tables are to be used. NARP is used only to resolve the address for a host in another subnet. Even in that case, the NBMA ARP server (NAS) is assumed to reside at an explicitly configured IP address.

In RFC-1620 [3] the use of Address Resolution servers for shared media is outlined, but the document goes on to say that end stations must know the layer 2 address of the server - there is no intent to serve address resolution via broadcast.

Finally, RFC-1433 [4] deals with a type of proxy ARP service in which foreign IP addresses are resolved to layer 2 addresses. Again, this is quite different than the current problem, in which we are dealing with addresses within the same subnet.

As there was no existing standardized method for solving the problem without having to configure each subscriber's machine(s), we proceeded to experiment with possible solutions.

## 3. ISP PREMISE TESTS

The solution we developed was to connect a server to a physical interface at the ISP's bridge / router distinct from the interface terminating the subscribers, as indicated in Figure 1. As a result, the ARP requests are broadcast to the server. The server then broadcasts an ARP request to the destination on behalf of the source. Once the destination replies, the ARP server caches the reply as well as forwarding it to the original client.

The test environment used to develop this solution consisted of two clients in separate geographic locations attached to the same NBMA network by DSL modems. The PVCs from the modems were terminated on a bridged virtual interface (BVI) within one router card of the bridge / router, which was a Cisco 6400. An Ethernet interface on the router card was used to connect a PC running the test software to the BVI.

When a client sends out an ARP request, the bridge / router will broadcast the request out the interface to which the ARP server is attached because it is on a different physical interface than the one on which the broadcast originally arrived.

We first tried to have the PC create the ARP replies itself, but the 6400 refused to forward frames containing the containing its own MAC address (the MAC address of the BVI). We then configured the ARP server to react to an ARP request by originating a new ARP request for the destination on behalf of the original client. On receipt of the reply, the result is cached at the server and also sent back to the original client.

This is a much better solution than just turning the PC into a special-purpose router, or finding some way to coerce an additional router into performing this service. Once the ARP server has returned the ARP reply to the original source, subsequent traffic flows via the 6400 without involving the ARP server on the PC. This removes the PC from the series path of the subsequent data flow.

Note that the ARP server is performing a service quite different than simple proxy ARP, in that it is exploring the subnet and replying on behalf of clients in the local subnet. The service is actually more like ARP spoofing, except that the server actually points the requesting client at the correct destination machine. This solution allows providers to maintain their current configurations while removing the need for any manual setup or maintenance of clients.

## 4. SCALABILITY

Once we had a working solution to the problem, we needed to determine how scalable the idea was. Even though the ARP server was not in the direct data flow of the DSL service, we wanted some idea of the rate of ARP requests it could be expected to handle. A simple PC running Linux might be acceptable in a production environment because this particular function may not be sufficiently critical that a highly reliable platform is required.

Performance tests were run in the Communication Networks Research Lab at the University of Alberta. The lab test environment consisted of a PC that hosted the software, a Wandel & Goltermann DominoFE internetwork analyzer that generated ARP requests, and a Cisco 2505 router. The PC was an Intel 400 MHz Celeron-based machine with 64 MB of SDRAM and five 3Com 3c905B Ethernet interfaces. The operating system used was RedHat Linux 6.1, kernel 2.2.12. We used three different test configurations to determine the effects of network rate and number of interfaces used.

### 10 Mbps, HDX

In this test, the DominoFE was attached to one port of the 10 Mbps half-duplex Ethernet hub on the Cisco 2505. An Ethernet interface on the PC was attached to another port on the hub. The DominoFE generated ARP requests and sent them to router which then flooded them to the PC. The software on the PC received the ARP requests and sent back replies. No ARP lookups, ARP table maintenance or caching was done by the server; a fixed MAC address was used in all ARP replies.

For this test, the Ethernet utilization (which was due solely to ARP traffic) was set at levels between 1% and 10%. For each value of Ethernet utilization, the CPU utilization was recorded along with the number of ARP requests processed per second. The duration of each test was 60 seconds. Table 1 shows the results of this test.

| Ethernet Utilization, % | ARP requests / sec | CPU utilization, % |
|---|---|---|
| 1 | 126 | 1 |
| 2 | 250 | 1 |
| 3 | 373 | 2 |
| 4 | 498 | 2 |
| 5 | 622 | 2 |
| 10 | 1241 | 5 |

Table 1. Results of testing with 10Mbps HDX interface

The ARP loads used during this test are quite high. Virtually all network operating systems implement MAC address caching so there will not be an excessive amount of ARP traffic. For example, if clients typically hold one ARP entry (e.g. for gaming), and they age that entry out after 60 seconds, then a load of 100 ARPs/sec represents the load that would result from 6000 concurrently active clients in a single subnet.

Our results show that the software can handle over 1200 ARPs/sec using a simple 10 Mbps HDX link. Using the rough guide of 100 ARPs/sec for 6000 clients, this would be sufficient to support 72,000 subscribers.

### 100 Mbps, FDX, One Interface

The second test involved only the DominoFE and the PC. In this configuration, ARP requests generated by the DominoFE were sent directly over a 100 Mbps full duplex Ethernet link to an interface on the PC. The software sent all ARP replies back out the interface on which they arrived.

In this test, Ethernet utilization was started at 10% and was increased until it reached 20%. Again, the CPU utilization of the software and the number of ARP requests processed per second were recorded. The duration of each test was 60 seconds. Table 2 contains the results of the second test. The data indicates that the software can comfortably handle a load of 10,000 ARPs/sec on a 100 Mbps full-duplex link. Using the same rough guide as above, this would be sufficient to support 600,000 subscribers.

| Ethernet Utilization, % | ARP requests / sec | CPU utilization, % |
|---|---|---|
| 10 | 12,376 | 51 |
| 15 | 17,808 | 82 |
| 17 | 18,298 | 99 |
| 20 | 16,816 | 99 |

Table 2. Results of testing with 100Mbps FDX interface

At loads over 15,000 ARPs/sec, some losses were noted. That is, the number of ARP requests recorded did not equal the number of replies. We suspect that NIC buffer overruns were destroying some of the traffic.

### 100 Mbps, FDX, Two Interfaces

The third configuration was an extension of the second: instead of sending ARP replies out the receiving interface, the software sent them out a second 100 Mbps interface. Ethernet utilization was increased from 10% to 25%, and the number of ARP requests processed per second and the CPU utilization were recorded. Each test lasted 60 seconds.

| Ethernet Utilization, % | ARP requests / sec | CPU utilization, % |
|---|---|---|
| 10 | 12,378 | 26 |
| 12 | 14,989 | 28 |
| 15 | 18,707 | 35 |
| 17 | 21,129 | 45 |
| 20 | 24,935 | 74 |
| 25 | 13,920 | 99 |

Table 3. Results of testing with two 100Mbps FDX interfaces

By adding a second interface for outbound traffic, we hoped to increase the rate at which responses could be generated (see Table 3). Our results show that the software operated well up to about 25,000 ARPs/sec. This demonstrates conclusively that a simple PC should be able to service even an extremely large DSL NBMA subnet of over 1,000,000 subscribers.

## 5. SUMMARY AND FUTURE WORK

NBMA environments limit broadcast traffic. While this is desirable because it decreases bandwidth consumption, it stops protocols such as ARP from working. We created an application to re-enable ARP and tested its performance on a Linux-based PC. The results show that the software can handle 1200 ARPs/sec over a simple 10 Mbps HDX link, 10,000 ARPs/sec on a 100 Mbps FDX link, and 25,000 ARPs/sec on a 100 Mbps FDX link using separate interfaces for receive and transmit. This final value represents the load that might be generated by over 1,000,000 concurrently active clients, so that the use of a simple PC is not a significant limit in a production environment.

Overall, this is a good solution to the requirement for subscriber-to-subscriber communication in a typical NBMA DSL network. The server generates an ARP request on behalf of the original client in order to discover the IP-MAC mapping needed, and then distributes this information in an ARP reply. It also caches and ages the result for future use. There is a minimal amount of functionality supplied by the server, and it is out of the series path of the normal communication between subscribers.

In the future, we propose to implement this application within the Linux kernel in order to improve performance. This could be done either as a kernel module, similar to kHTTPd and kNFSd, or as a straight modification to the ARP code. This application has also suggested a new form of bridging that we call ``active bridging" that is useful in this type of NBMA environment, where Layer 3 information can be used to give direction to Layer 2 operation. The use of active bridging in NBMA environments will also be explored further.

## References


1. D.C. Plummer, *"RFC-826: Ethernet Address Resolution Protocol"*, November, 1982.

2. J. Heinanen and R. Govindan, *"RFC-1735: NBMA Address Resolution Protocol (NARP)"*, December, 1994.

3. B. Braden, J. Postel and Y. Rekhter, *"RFC-1620: Internet architecture extensions for shared media"*, May, 1994.

4. J. Garrett, J. Hagan and J. Wong, *"RFC-1433: Directed ARP"*, March, 1993.