# IP over MSNL

Michael Roe

February 1992

## Introduction

This document describes the protocol used for carrying IP datagrams over MSNL. This is done by "tunnelling" — making one network layer appear to be a link layer and then running a second (different) network layer on top.

It is theoretically possible to run IP directly over the link layer of the Multi Service Network Architecture, by permanently allocating a virtual circuit identifier to IP. The tunnelling approach is used instead for several reasons:

- With tunnelling, the routers between different physical networks do not need to understand IP, only MSNL. This can be important because MSNL headers are far simpler to decode than IP headers, and reducing the processing carried out by routers increases their throughput.
- Tunnelling should result in lower latency over multi-hop routes, as fragmentation and reassembly is only performed at the end-points, not at every router.
- Tunnelling should also reduce the amount of jitter. It is likely that an implementation of IP over MS Data Link could only work on fragmenting one IP datagram at a time, and hence the granularity of multiplexing would be an entire IP datagram rather than an MSDL cell.

## Indirect Access to IP/MSNL

Hosts which do not support MSNL can make use of an IP/MSNL tunnel by forwarding IP packets to a host which does (using IP over link layer).

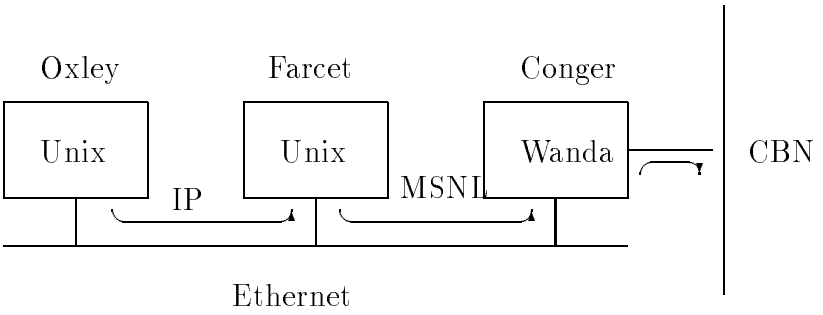


Figure 1: Indirect Access

Where possible, hosts should make a direct IP/MSNL connection rather than indirect access, as this will result in lower latency and jitter. However, not all hosts can be modified to support IP/MSNL (eg. those for which we do not have the kernel source code). This method allows such

hosts to continue to operate when the only connection from the LAN to the rest of the Internet is IP/MSNL.

This method can also be used in reverse, to enable hosts which only support IP/MSNL to send datagrams to hosts without MSNL. For example, hosts whose only network connection is the CFR can only send IP/MSNL, and must access IP/ethernet hosts indirectly.

## Address Mapping

There needs to be a means to translate a host's IP address into the MSNL address and port number to use to send it tunnelled IP packets. With this version of the protocol, there is no address mapping service: MSNL port number 56 decimal (ie. 0.0.0.56) is always used and it is assumed that the host's MSNL address is the same as its IP address. Treating the two address domains as the same gives rise to some routing problems: these are discussed in detail later.

## The Protocol

Each IP datagram is fragmented using the SAR1 protocol over MSNL. No additional protocol id, length count or checksum is added to the datagram before fragmentation. To send an IP datagram over MSNL to a host, it is sufficient to open an MSNL circuit to port 56 (or use an existing open circuit) and send the fragmented datagram over it. No additional control information need be sent over the MSNL circuit, as the IP header contains all necessary information.

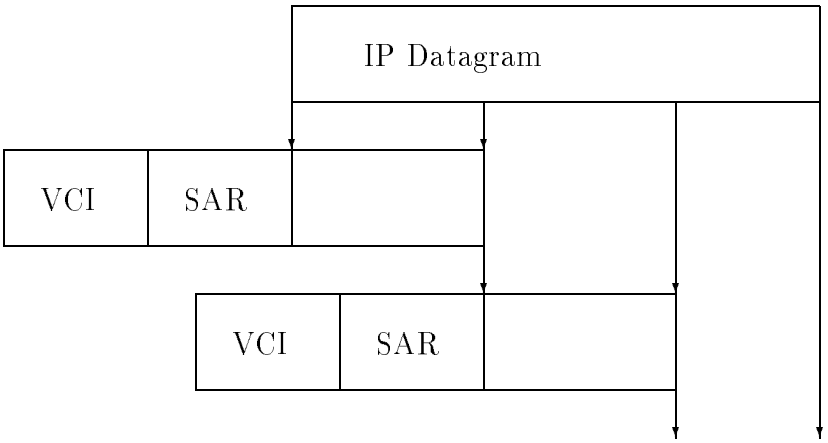


Figure 2: IP over MSNL Fragmentation

## Routing Problems

The current version of the IP/MSNL protocol assumes that a host's MSNL address and its IP address are numerically identical. For this to work, the MSNL and IP routing tables must be co-ordinated. There are several possible approaches:

### Use one IP subnet for IP/MSNL

One IP subnet of 1024-2 hosts would be designated as the IP/MSNL subnet. There would also be allocated 16 MSNL subnets of 256 hosts, chosen so that the MSNL address = IP address assumption holds.

### Examples

Here we assume that the IP/MSNL subnet is 128.232.64.0, and subnets 128.232.0.0 and 128.232.32.0.0 are only connected by IP/MSNL, with gateways *farcet* and *mune*. *faeroes* is given as an example of a machine which does not support IP/MSNL.

*faeroes* would have the following routing table:

```
   net           netmask      gateway      interface
128.232.0.0   255.255.240.0  (self)           se0
128.232.32.0  255.255.240.0  128.232.1.214    se0
128.232.64.0  255.255.240.0  128.232.1.214    se0
```

*farcet* would use 128.232.1.214 for IP/ethernet and 128.232.64.1 for IP/MSNL. It would have the following routing table:

```
   net           netmask      gateway       interface
128.232.0.0   255.255.240.0  (self)           se0
128.232.32.0  255.255.240.0  128.232.66.1     ms0
128.232.64.0  255.255.240.0  (self)           ms0
```

*mune* on the other ethernet would use 128.232.32.9 for IP/ethernet and 128.232.66.1 for IP/MSNL. It would have the following routing table:

```
   net           netmask      gateway       interface
128.232.0.0   255.255.240.0   128.232.64.1    ms0
128.232.32.0  255.255.240.0   (self)          se0
128.232.64.0  255.255.240.0   (self)          ms0
```

### Advantages

Only uses up one IP subnet.

### Disadvantages

There can be at most 256 hosts talking IP/MSNL on any one physical net. This limit may be a problem on the lab ethernet: suppose all DecStations, all VaxStations, all R140's and all Wanda machines with ethernet cards all use IP/MSNL. The small size of the host number means that it will not be possible to automatically assign hosts IP/MSNL addresses based upon their current IP address. (In the above example, farcet needs to talk IP/MSNL and gets .1 on a first-come-first-served basis). Assigning IP/MSNL in this way will be difficult to administer.

## Use a Class B network for IP/MSNL

There could then be 16 MSNL subnets of 1024 hosts, chosen so that the MSNL address = IP address assumption holds. IP hosts would treat the network as if it was not subnetted.

### Examples

Suppose we obtained 128.254.0.0.

*faeroes* would have the following routing table:

```
   net       netmask        gateway        interface
128.232.0.0  255.255.240.0   (self)           se0
128.232.32.0 255.255.240.0  128.232.1.214    se0
128.254.0.0  255.255.0.0    128.232.1.214    se0
```

*farcet* would use 128.232.1.214 for IP/ethernet and 128.254.1.214 for IP/MSNL. It would have the following routing table:

```
   net       netmask        gateway        interface
128.232.0.0  255.255.240.0  (self)           se0
128.232.32.0 255.255.240.0  128.254.2.9       ms0
128.254.0.0  255.255.0.0    (self)           ms0
```

*mune* on the other ethernet would use 128.232.32.9 for IP/ethernet and 128.254.32.9 for IP/MSNL. It would have the following routing table:

```
   net       netmask        gateway        interface
128.232.0.0  255.255.240.0  128.254.1.214    ms0
128.232.32.0 255.255.240.0  (self)           se0
128.254.0.0  255.255.0.0    (self)           ms0
```

#### Advantages

There will definitely be enough IP/MSNL addresses to give one to every host.

#### Disadvantages

Another Class B network number will have to be obtained.

### Use half of our subnets for IP/MSNL

128.232.128.0 netmask 255.255.128.0 would be used for IP/MSNL. As far as IP is concerned, this would be 8 subnets with netmask 255.255.240.0. 8 MSNL subnets of 1024 hosts would be allocated so that the MSNL address = IP address assumption holds.

#### Examples

*faeroes* would use 128.232.0.155 for IP/ethernet and have the following routing table entry:

```
   net          netmask     gateway
128.232.0.0   255.255.240.0  (self)
128.232.32.0  255.255.240.0  128.232.1.214
128.232.128.0 255.255.240.0  128.232.1.214
128.232.160.0 255.255.240.0  128.232.1.214
```

*farcet* would use 128.232.1.214 for IP/ethernet and 128.232.129.214 for IP/MSNL. It would have the following routing table:

```
   net          netmask     gateway       interface
128.232.0.0   255.255.240.0  (self)          se0
128.232.32.0  255.255.240.0  128.232.130.2   ms0
128.232.128.0 255.255.240.0  (self)          ms0
128.232.160.0 255.255.240.0  (self)          ms0
```

*mune* on an experimental ethernet would use 128.232.32.9 for IP/ethernet and 128.232.160.9 for IP/MSNL. It would have the following routing table:

```
   net           netmask       gateway          interface
128.232.0.0    255.255.240.0  128.232.129.214    ms0
128.232.32.0   255.255.240.0   (self)            et0
128.232.128.0  255.255.240.0   (self)            ms0
128.232.160.0  255.255.240.0   (self)            ms0
```

**Advantages**

Gives us enough host numbers without applying for a new Class B network.

## Current Approach

The current approach is to use one IP subnet for MSNL. We aim to switch over to the last approach, using half our subnets for IP/MSNL. This approach was not taken initially as we were not confident that the Unix implementation of IP would be able to cope with it; recent experiments have shown that it can be made to work.

The current scheme assigns Olivetti Research two of our subnets. Again, this approach was taken because we were confident it would work (even though it clearly does not scale to assign every IP/MSNL user one of our subnets). We aim to switch over to Olivetti Research using a separate Class C network for IP/MSNL. Recent experiments have shown that this will work, and will still allow direct IP/MSNL conenctions to be made between the CL and ORL.

The current subnet structure is as follows:

| Address | Netmask | Notes |
|---|---|---|
| 128.232.0.0 | 255.255.240.0 | CL ether |
| 128.232.16.0 | | |
| 128.232.32.0 | | |
| 128.232.48.0 | | |
| 128.232.64.0 | 255.255.255.0 | CL ether (CL derived) |
| 128.232.65.0 | 255.255.255.0 | CL ether (CL derived) |
| 128.232.66.0 | 255.255.255.0 | CL ether (CL derived) |
| 128.232.67.0 | 255.255.255.0 | CL ether (CL derived) |
| 128.232.68.0 | | |
| 128.232.69.0 | | |
| 128.232.70.0 | | |
| 128.232.71.0 | | |
| 128.232.72.0 | | |
| 128.232.73.0 | | |
| 128.232.74.0 | | |
| 128.232.75.0 | | |
| 128.232.76.0 | 255.255.255.0 | CL CFR (CL assigned) |
| 128.232.77.0 | 255.255.255.0 | CBN |
| 128.232.78.0 | 255.255.255.0 | ORL CFR (ORL derived) |
| 128.232.79.0 | 255.255.255.0 | ORL ether (ORL derived) |

Figure 3: Interim subnets for IP/MSNL

Addresses in the subnets marked "CL derived" are derived from the host's IP/ethernet address. 128.232.$x$.$y$ ($x < 16$) uses 128.232.64 + $x$.$y$ for IP/MSNL/ethernet.

Addresses in the subnet marked "CL assigned" are assigned on an ad-hoc basis.

Addresses in the subnet marked "ORL derived" are derived from the host's IP/ethernet address. 192.5.239.$x$ uses 128.232.78.$x$ for IP/MSNL/CFR and 128.232.79.$x$ for IP/MSNL/ethernet.

## Performance Measurements

The round-trip time for an ICMP echo request ("ping") was measured for various packet sizes using IP over ethernet, IP/MSNL over ethernet, and IP/MSNL over a multi-hop circuit (two ethernets and the Cambridge Backbone Network).

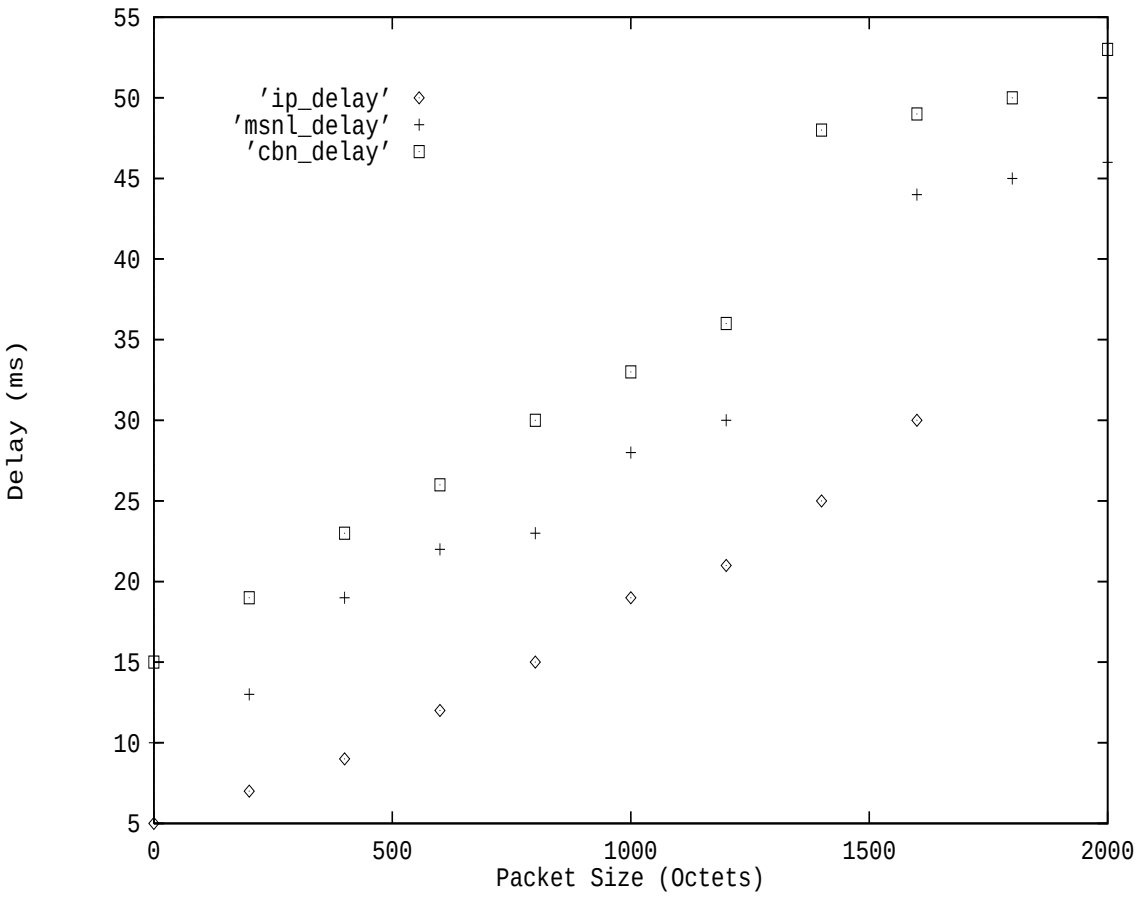


Figure 4: ICMP Round Trip Time

- The delay increases linearly with packet size, with about the same slope in all three cases. This slope represents the overhead in dealing with the user data (reading data in from network interface, calculating a checksum) and this overhead is not significantly increased by using MSNL IP.
- There is a discontinuity in the graph at about 1400 octets, as datagrams above that size will not fit into a single ethernet frame. The size of the jump in the graph represents the additional overhead of transmitting and receiving an additional ethernet frame. It was not possible to measure the round trip time with ethernet IP for ICMP datagrams of this size, as the Unix implementation is incapable of generating them.

- The 3 hop route with IP/MSNL takes about 4ms longer than the one hop route, regardless of the size of the datagram. It was not possible to generate truly comparable figures for normal IP over a three hop route, but it is likely that this would take about 3 times as long as the one hop route, resulting in an additional delay of between 10 and 60ms depending on the packet size. Thus, although IP/MSNL is slower for one hop routes, it can be significantly faster when there are many hops.

## Future Directions

It should be possible to use a separate MSNL circuit for each (host, dest, protocol, src port, dst port) tuple rather than each (host, dest) pair as at present. This would involve violating layering to a certain extent; the ms0 interface (notionally a link layer entity) would have to access transport layer information in order to decide which circuit to use. Despite this, it has several advantages:

- It would be possible to allocate different quality of service parameters to the different virtual circuits. For example, NTP and telnet traffic could be treated differently by MSNL routers. (Presumably NTP requires low jitter while telnet requires a low packet loss rate).
- With the Wanda operating system, it would be possible to have different priority threads handling different protocols. Thus, time-critical applications such as NTP could have their packets given higher priority. At present, all IP level processing must be performed at the same priority, as the system does not know which application the packet is destined for until both the IP and UDP (or TCP) headers have been decoded.
- With Wanda, it would be possible to send IP/MSNL traffic directly to the application that wanted it, without either the kernel or any other process needing to look at the IP header. This has advantages for real-time scheduling, as more of the processing is done at the priority of the application, rather than the priority of the kernel or an IP demultiplexing daemon. It would also improve the performance of Wanda IP, as there would be no need to context switch between the application and the IP daemon.

To do this, it will longer be possible to use a single statically defined MSNL port. An address/port mapping protocol will be needed to translate (IP address, protocol, destination port) into the MSNL address and MSNL port number of the destination entity.