Computing Laboratory Technical Report No. 9/94

IPP Routing Architecture

Damiano Bolla
University of Kent at Canterbury, England
May, 1994

Abstract

This paper describes the structure of the IPP routing architecture. IPP is an evolution of the TCP-IP protocol. The main advantage of IPP is the variable length addressing scheme. An IPP address can be fifteen bytes long and is optimised for local area networks. The logical structure of the address is very similar to IP, with the main difference, that each byte specifies a subnet, apart from the last byte that indicates a host.
Routing speed is maximised by the fact that all routing tables are accessed using a direct lookup method. The size of the routing tables within a router is fixed and small. The above two points allow the construction of very cheap and fast routers.
This routing architecture supports broadcast, multicast and real time data. It uses different routing priorities for each type of service. This results in a better management of network links. For normal network traffic the link usage is maximised by automatically load balancing the usage of available links. A working router can be found in the part of this document describing the tests done on this architecture.
IPP aims to keep all the other qualities of IP Eg: the method used to manage flow control, resequencing, etc.. The only things that changes are the structure of an address, the routing table and routing functions.

Table of Contents

1. Introduction

This paper is about a computer Network Routing Architecture. The need for routing arises when two computers need to exchange data and there is a choice of which links to use to transfer the data across the network. In this paper I assume that a network can be composed of many millions of hosts and that they can be distributed across the world. In this situation deciding what is the next link that has to be used to reach the desired destination can be a complex task.

In this paper I will not discuss the problems associated with providing a reliable connection on top of an unreliable one or how to provide the physical transport of the data over the network. I am only concerned with the Level Three of the OSI model.

OSI
Reference
Model

Application	Level 7
Presentation	Level 6
Session	Level 5
Transport	Level 4
Network	**Level 3**
Data Link	Level 2
Phisical	Level 1

This paper makes reference to TCP-IP. TCP-IP is a set of protocols providing a means to transfer data within a network with an address length of four bytes. The basic building block of TCP-IP is an IP packet that provides unreliable transfer of data from one end to another. On top of IP is it possible to build a reliable connection (TCP-IP) or have something that is more reliable then IP but less than TCP-IP (UDP). For further reference see [Davi-88].

IPP is shorthand for **IP P**lus, an improved (from the routing point of view) IP protocol. It is desirable to increase the limited addressing space of IP, to provide better routing management and to support advanced services like multicast and real time data. This paper addresses the routing aspect keeping all the other qualities the same. In this document there is a continuous reference to IP and TCP and it is therefore assumed that the reader is familiar with this subject.

This report presents an almost completed version of the IPP structure. At this point I am more concerned about explaining the structure of the router and how the address space is managed than talking about theoretical aspects. However, chapter two is a reminder of what I am trying to achieve.

- Section two outlines the desirable qualities of a routing architecture. This list is the result of reading and experience and it is stated in general terms.
- Section three describes the structure of IPP. It is possible to experiment with the routing aspects of IPP using a tool kit that has been written for the purpose. The tool kit simulates a real router and hosts in a network and has been used to test the IPP solutions proposed.
- Section four shows possible examples that can be tested using the IPP tool kit. The tool kit allows a test network to be built and checks how the routing algorithms perform.
- Section five discusses issues that are related to routing but in an indirect way. One of this issues is how to migrate from an IP addressing scheme to IPP.
- Section six concludes this report with a very brief outline of the achieved objectives.

2. Desirable qualities of a routing architecture

This part will attempt to specify what can be considered important from the routing point of view. Note that the order of importance is not well defined and depends on the particular situation being dealt with.

A routing architecture should provide the router with a simple and fast way to decide where to send the incoming data. A simple example of a "router" is an employee of a Post Office. He is in charge of deciding where to send the incoming letter given the recipient's address.

2.1. Address aggregation

Address aggregation is the ability to hide routing information for a set of addresses so the network community does not need to know how to reach each single address. A very common example of address aggregation can be made using a mail address. Eg:

John Smith
University of Kent
Canterbury
England

From the address we understand that "England" hides all the remaining parts of an address. "Canterbury" hides 'University of Kent, John Smith'. "University of Kent" hides the final destination that is 'John Smith'

Without address aggregation all hosts would need a table that maps all possible destinations into a router that is able to route to that host. This is obviously not reasonable for Wide Area Networks. Address aggregation is therefore the first quality that is needed in a routing architecture. See [Estr-92 Par 2.2]

2.2. Distributed routing tables

This property not only states that the routing tables should be distributed across the network but it also assumes that the total size of **all** routing tables grows linearly with the number of hosts/Subnets present in the network.

- If a new Subnet/Host is added the total size of routing information distributed across the network is increased by one unit only.
- The routing information is distributed evenly across all routers in the network. This is of course a desirable property since it is not economical to build routers with huge amounts of memory and it is also difficult to maintain large routing tables.
- Changes in routing tables should remain as local as possible. It would be very inconvenient if a change in a remote part of the network needed to be propagated throughout the whole network.

2.3. Load balancing and redundant routes

The routing architecture should allow for more than one link to a given destination and this should provide load balancing (if desired) and a redundant route to the given destination in case one of the links fails. The load balancing is especially desirable to control network congestion since it allows bandwidth to be created on demand and makes it available to everybody who requests it.

2.4. Expandable addressing space

The current problem of IP is the lack of address space. It should be noted that it is not reasonable to have an unlimited addressing space since this will pose a great burden on the routing algorithms. It is a more reasonable requirement to have a variable length address but to impose an overall limit on the size of this field.

It is not acceptable to overload a packet with excessive address information when it is not needed. There should be a mechanism that allows the redundant part of an address to be stripped off when it is not needed. This process of address abbreviation should be transparent to the user since the user is not concerned with how the network transports the data from one side to the other.

2.5. Efficient Multicast

First of all I have to specify what it is meant by multicast. It seems to be widely accepted that multicast is a way to distribute packets of information to a *list* of recipient hosts. The hosts may be anywhere in the network. In this situation there are two main problems to solve to provide the above service. The first is managing the multicast list of hosts that are part of the same multicast group. The second is finding the best tree of network links that connects all the hosts in the best bandwidth efficient way. This process is called finding the best spanning tree for a multicast list. For further Reference see [Raj-92]

2.6. Support for real time data

The normal behaviour of IPP is to optimise network bandwidth usage and redundancy. To be able to do this, no guarantee is made on the order on which the packets will arrive or the time they will take to travel across the network. For some applications this is not acceptable. There should be a way by which the routing algorithm detects that a packet should meet some special connection constraints in terms of timing and other parameters.

3. IPP Solution

In this part it is explained how IPP addresses the above requirements. This is done by first presenting the general ideas and then by going into the detailed description of the routing algorithms.

It is worth remembering that I am only concerned with the routing of a packet from one point to another. It is not guaranteed that the packet will arrive at the destination and like IP it is possible to build a reliable protocol (TCP) on top of an unreliable one (IP). There is however provision for special classes of packets that have different privileges. You can consider Multicast and Real Time data as a separate router scheduling classes. [Shenker-93]

3.1. Description of the IPP architecture

This part describes the structure of IPP in simple terms. There are various aspects that need to be explained.

- The relationship between the network structure (topology) and a IPP address.
- The structure of a broadcast address.
- The structure of a Multicast address.
- The structure of a Real Time address.

3.1.1. IPP Network Topology and Addressing Scheme

IPP is a variable length, limited, addressing scheme architecture. The address format is very much like IP, i.e. addresses are in the form 148.162.2.67 but addresses can be of variable length. The maximum IPP address length is 15 bytes. What follows is a series of examples explaining the various address parts.

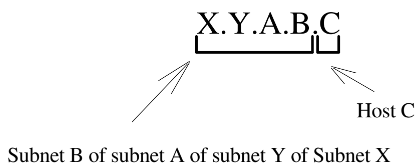
Note:

- An address cannot contain a zero.
- Address **255** is reserved for broadcast.
- Address **254** is reserved for Real Time data.
- Address **253 252 251** are reserved for future use.

Example:

1	This identifies a host. Host 1 at the topmost level
2.4	This is host 4 but on Subnet 2
1.2.4	Host 4 again but on Subnet 2 of Subnet 1
3	Host 3 at topmost level
147.162.2	Host 2 on Subnet 162 of Subnet 147
148.162.2.30	Host 30 of Subnet 2 of Subnet 162 of Subnet 148
6.8.9.3.5	Host 5 of Subnet 3 of Subnet 9 of Subnet 8 of Subnet 6

The above examples show how the rightmost number always identifies a host in the given hierarchy of networks.

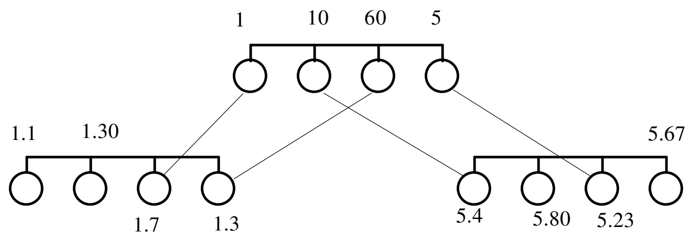


At the topmost level of the tree there can be 1 to 250 hosts or routers that are able to route to a Subnet. The topmost level can have 250 Subnets each having 250 hosts or routers. A **domain** is formed by the 250 hosts that are part of the same Subnet.

The network topology of an IPP network is closely related to the address given to any host or router. There are two rules.

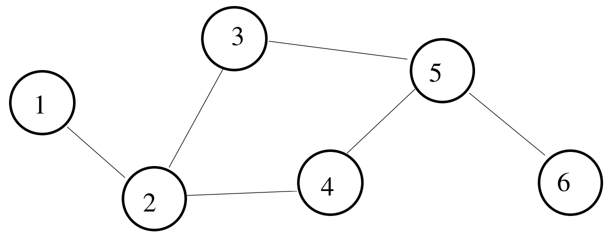
- Any router of a domain must be able to reach any other router of the same domain **without** using a parent network or subnet. This is equivalent of IP where a member of an IP subnet cannot send one packet to another member of the same subnet using another subnet.
- The address of a host/router **must** be either a parent of the domain where it is directly connected or a member of the same domain or a subnet. It is not possible to have a direct link to a domain that is more than one subnet above or below a given domain.

At this point clearly such a routing scheme is too weak if multiple routes for a host or Subnet are not provided. The desired structure is something like this.



From the diagram it can be seen that there are two Subnets, Subnet 1 and Subnet 5. Each one is connected to the parent by two links. A packet for Subnet 5 can go via router 10 or via router 5. This therefore achieves what is load balancing and redundancy at the Subnet level. (This is an extension of the Fat Tree concept [leis-92]). There can of course be more than two links to a Subnet.

It is also possible to achieve load balancing in the same domain. This means that for the network structure as follows.

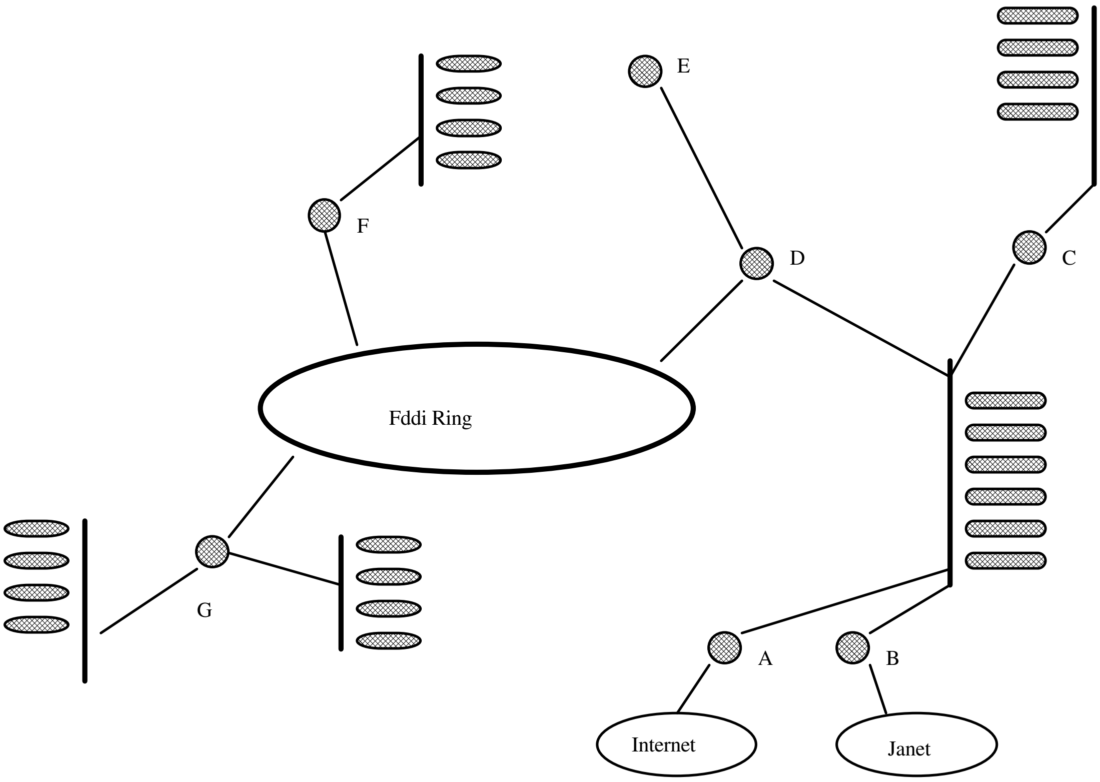


It is possible to set the routing tables so there is load balancing on routers 3 and 4 when packets are from host 1 and 6 (Or 2 and 5). More in section 4.5.

It is important to note that the routing tables for each router are of fixed size. If a new Subnet or host is added the changes are local to the domain and do not change the parameters of other routers.

One of the main objections of having a close relation between the address and the network topology is that it is not flexible enough for today's networks. I will now try to show how the required constraints are not as strong as they seem.

A real example can be the network topology of a university. This is a simplified version but the concepts still apply. The idea is to show that you can arrange your subnet in a meaningful way even with current topology.



I have to assume that a subnet identifier has been given to this organisation, this is like IP. E.g.: 1.2.3 This university can then decide how to manage the remaining addressing space below 1.2.3. One possibility is to put routers A,B,D,G, on the same domain and C as a sub domain.

Alternatively you can have only A,B in the same domain and all the other routers as a sub domain.

The question is then how to decide if to put a router on the same domain or in a sub domain? The answer is "depends on the traffic locality". That is a sub domain should be created if there is a forecast that the machines that are part of that domain will have local traffic.

3.1.2. Broadcast address Structure

A broadcast address uses the special octet 255 to identify all hosts of a given domain. Examples of broadcast addresses are:

255	Broadcast to all hosts at the top level domain
1.2.255	Broadcast to all hosts of the subnet 1.2
1.255.255	Broadcast to all hosts of subnet 1 **plus** the broadcast is sent to all subnet where it is further expanded.

3.1.3. Multicast address Structure

The adopted definition of Multicast is as follows: a multicast address represents a list of recipients to which the given packet has to be delivered. No assumptions are made about particular address properties of the recipients.

There is the need for somebody to maintain this list of hosts that belong to the same multicast list and there is the need to uniquely identify a multicast list.

A multicast identifier can be called an "addressed broadcast" since it is an address that starts with 255 and it is followed by the real local multicast id for the list.

255.3	Identifies the local multicast list number 3
255.3.5	Identifies the local multicast list 3.5

The detailed structure of multicast routing will be explained in detail later. To uniquely identify a multicast list, I use the address of the managing host followed by the local multicast identifier for that list. If the managing host is 3.4 and the multicast id for the list in the domain 3.x is 255.9 I can uniquely identify the multicast list with the following address.

3.4.255.9

3.1.4. Real Time address structure

Real time packets travelling on the network **must** be subjected to special routing algorithms. They may be guaranteed to be in sequence and to have a specified range of possible time delays.

To be able to enforce such special behaviour, the first thing that a router needs to do is to recognise such a special packet. Giving the requirement that an IPP network must be able to use a fast data link, it follows that the operation of recognising special packets should be simple.

For this reason there is a separate class of addresses, similar to multicast, for real time data. The set-up of a real time connection between two arbitrary hosts will be done by the calling host. The calling host will take care of asking all the domains where the packet will travel for a *Real Time Id* and for the minimum connection requirements in terms of delay and data throughput. This is called admission control and is one of the steps required to be able to guarantee the specified requirements.

Once all the domains grant the required bandwidth and time delay the calling host can ask for the route to be set-up. The calling host will then send packets with the special address prefix 254.xxx The router identifies this special packet and deals with it accordingly. The router is therefore bound to deliver the **Q**uality **O**f **S**ervice that was requested and granted at connection establishment. To avoid possible deadlocks the granted real time resource will be released if there is no usage for a specified amount of time.

The structure of a real time address is therefore 254.xxx...

3.2. Packet Routing

Having in mind the structure of the network it is possible to understand how routing a packet is performed. This part is divided in sections with each one dealing with a particular aspect of routing. The first section will explain how sending packets within the same domain is performed. Then the subnet routing is explained and finally the routing to a parent domain is explained.

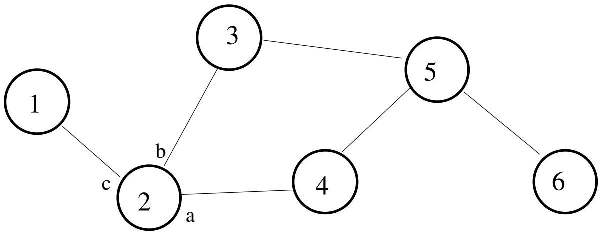
3.2.1. Same Domain Routing

IPP does not assume a network topology at domain level, i.e. hosts that belong to the same domain can be variously connected (See Section 4.7) but all hosts of the same domain **must** know how to reach each other **without** using another domain. It is known that there are a maximum of 250 hosts in the same domain. I can set-up a table that lists the interface to be used and who will be the next host to use to reach the desired destination.

Destination	Next Hop	Interface
1	23	1
2	42	2
3	2	1
4	33	3
.....	.....	.....

Each time I want to reach a host in the same domain I read the data in the *destination* field of the routing table and send the packet using the given Interface to the given NextHop.

To allow redundant routes to the same host I have to list more than one choice for each destination. For example if I use the following network topology.



The routing table will be different for each host. Host 1 has to use host 2 to route to any other destination. Host 2 can use two different links to reach host 5 therefore the routing table for host 2 will be:

Destination	Next Hop	Interface
1	1	c
5	3	b
	4	a
....	....	....

The first entry says that the Next Hop to reach destination 1 is 1 itself on interface c
There are two choices to reach host 5. One is using host 3 and the other is by using host 4.

To be able to do load balancing on the two or more routes I need a way to give a price to a route. This is done by adding a cost for every packet and by keeping track of how much has been spent on that route. The final route data structure is like this.

```
struct Route
        u_char    Next
        u_char    Interface
        u_char    Cost
        u_int     Cumulated
```

The meaning of the above fields is as follows.

- Next is the address in this domain or in a directly connected domain of the router/host that will be receiving the packet. If this value is Zero it means that this entry is not valid.
- Interface is the interface to use in this router to send the packet. In the case of a route to a Subnet or parent if this is zero it means that the Next is a distant router in this domain for the given Subnet.

- Cost is the cost of this route and is a means of determining how many packets will be sent using this particular route. In the case that a link is too expensive compared to others that are available, the Domain Admin may decide to completely remove that link from the routing table.
- Cumulated is the total cumulated cost of this route and is used to decide what route will be selected next by the route manager.

Note that the above description applies for both Same-Domain routings, Sub domain routing and Parent routing.

The complete routing table for the same domain routing is a set of possible routes for each destination. That is for **each** destination there will be a certain number (ROUTE_MAX) of possible routes that can be used.

Given that a network structure may contain routing loops there must be a way to avoid them. The first step in avoiding a network loop is to detect that the packet that is received is the same packet that was sent some time ago. Once this is done there should be a way to try to deliver the packet if it is possible.

For this reason a host entry will have an entry (Another) that is used for loop avoidance.

```
struct HostEntry
        u_char     Direct
        u_char     Another
        struct     Route Link[ROUTE_MAX+1]
```

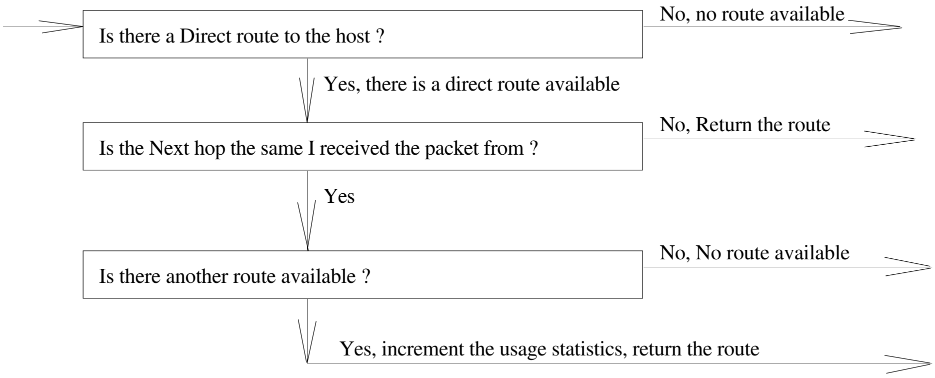
The array of Route contains the various possible routes to the desired host.

- Direct is an index to a direct route to the given host. It is used when a loop is detected and the router must make sure that the packet will arrive to destination.
- Another is also an index into the array of possible routes and is used to try to avoid network loops. This is used when a lookup in the routing table has as next hop the same host from which the packet is received. To avoid sending the same packet back to the router from which the packet just arrived, and thus creating a loop, the **Another** route is used.

Direct points to what is the cheapest route at the moment. **Another** points at the second cheapest route to the desired destination. Note that a route to a host must have a valid interface to use to reach the next hop in the same domain. The routing table for all possible hosts in the same domain is therefore an array of the above structures.

```
typedef struct HostEntry HostData[ADDR_NUM]
```

The function that finds a route to a host will then use the above data structure in the following way. For further details see the function **FindHostRoute**.



Since the routing tables are all direct tables determining the availability of a route is a very fast process. The search speed depends only on the speed of the processor. It does not vary with routing table size or other factors.

3.2.2. Subnet Routing

The duty of this table is to hold information on how to reach each of the Subnets of this domain. Again this table can have multiple routes for a given Subnet and as in the host table the routing algorithm just picks up the route that is prepared by the router manager.

The structure of a Subnet table is as follows:

```
struct NetEntry
        u_char    Try
        u_char    Direct
        struct    Route Link[ROUTE_MAX+1]
```

A routing entry for a subnet is similar to the routing entry for a host. Each entry has a set of routes available. In this case, however, the next hop can assume two different meanings depending if the route has a valid interface or not.

- If the route has a valid interface then NextHop is a directly connected router accepting packets in the subnet that we want to reach.
- If the route does not have a valid interface then NextHop is **not** a directly connected router to the desired subnet. It is a *distant* router for the given subnet. The address of a distant router is a single byte that indicates what router **in this domain** is able to directly send packets to the desired subnet.

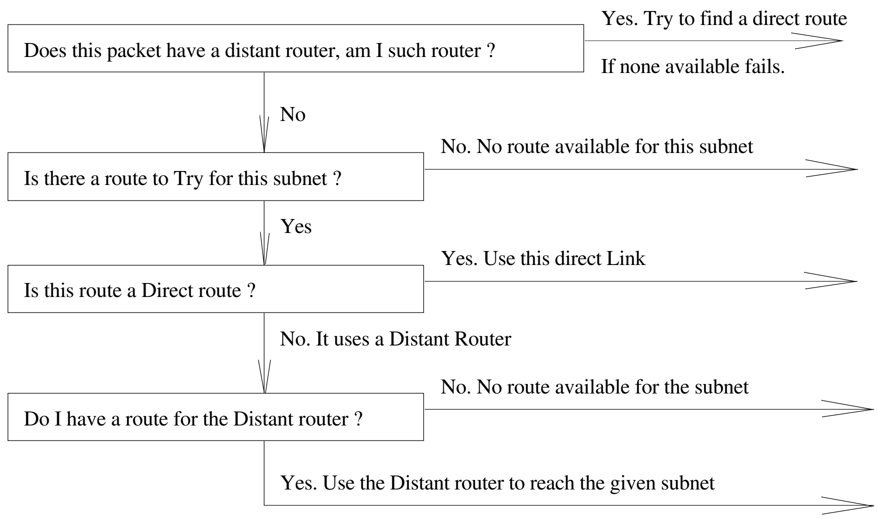
The meaning of Try and Direct is the following

- Try contains the index of the Route to use for this subnet. This index is updated to point of the cheapest route so far.
- Direct contains the index of a **direct** route to the given subnet. This field can be empty indicating that this router does not have a direct route to the subnet.

The table is an array of the above entries as follows.

typedef struct NetEntry NetData[ADDR_NUM];

The algorithm that finds a route to a Subnet is as follows. For further details see the function **FindSubnetRoute**.



3.2.3. Parent Routing

Since there is only one parent of a given Subnet this table contains only a series of multiple routes that can be used to reach the parent domain.

Finding a route for a parent domain is very similar to finding a route for a subnet. The structure of the algorithm is the same the only difference is the names of the variables. For further details see the function **FindParentRoute**.

3.2.4. Loop table

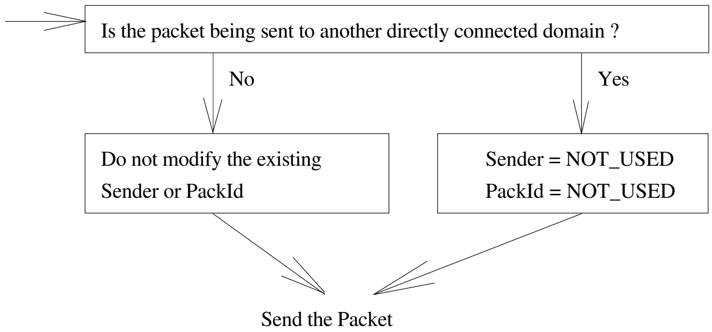
This table is needed to handle the possible loops that are present in a domain when there are multiple routes to a host. The principle is that a packet coming from one host with a given PacketId should not appear again within a certain amount of time.

Since there are 250 possible "hosts" in a domain and each one can generate 255 different packet ids, it follows that this table is a direct table holding 250*255 entries each one holding the last time a packet coming from the given sender with the given Id was seen.

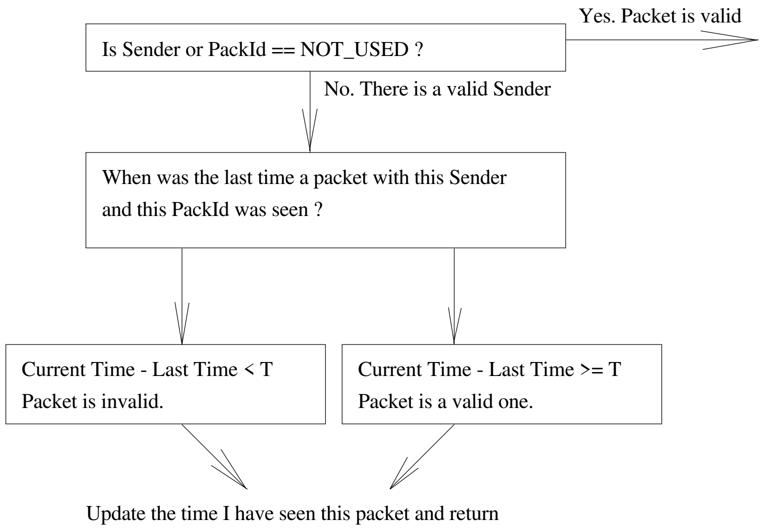
It is possible to set up the domain routing tables so loops are impossible. If I set up a routing table without loops I will lose some redundancy. Experiments done with the tool kit (See Section 4.5) shows that the number of packets dropped due to loop detection (In the case of routing tables with loops) is of the order of 5%. However this is not a fixed figure and will vary depending on the network topology, the routes cost and the traffic type.

The loop detection algorithm is closely coupled with the routing algorithm when a packet is sent down to a subnet or up to a parent. In the above two cases the packet id and the Sender are set to Zero (Indicating field not used) and the receiving router will assign a new packet id. In respect of loop avoidance, every time a packet crosses a domain the receiving router will appear to be the "creator" of the given packet.

The algorithm that creates a new packet id can be described as follows. See the function **FindRoute** for further reference.



When a packet is received from a router one of the validation checks that are performed is to see if the packet has already been seen. To do this Sender and PackId is used. For further details see the function **SeenAlready**.



A positive aspect of this method of detecting loops is that the time stored in the router table is not an absolute time. This means that the clocks in the routers of the same domain do not need to be synchronised.

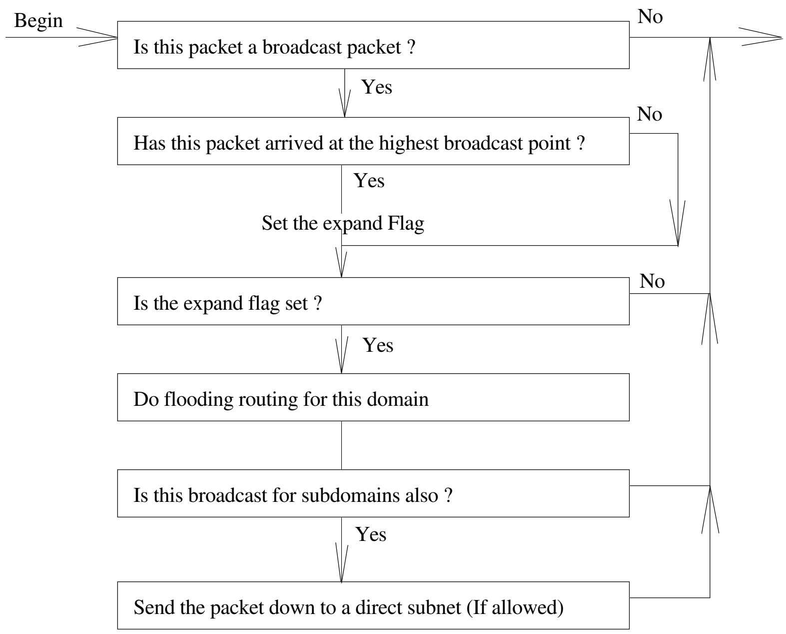
3.3. Broadcast Routing

This part describes the method used to implement broadcast routing. The strategy used is flooding the domain with the broadcast. One of the problems to solve is deciding when to start to "explode" a broadcast while trying to avoid creation of duplicate packets that cannot be recognised as such.

As an example, consider the following broadcast 1.255.255. This broadcast should reach all hosts in subnet 1 and all hosts in all the Subnets of subnet 1. To do this the expansion of a broadcast will always begin from the top of the network address tree even if the packet comes from a sub domain of the broadcast destination.

The routing algorithm for a broadcast can be outlined as follows. The interesting part is the fact that the normal routing algorithm will take care of sending the packet to a parent if the DoBroadcast function does not recognise the need to expand this broadcast.

When the packet is sent down to a subnet the router performs two checks. The first one is that the subnet must be directly connected to avoid unnecessary overhead. The second test is to see if it is the designated router for the given subnet. This avoids sending the same copy of a broadcast packet to the same subnet from two directly connected parents.



For further details see the function **DoBroadcast**.

3.4. Multicast Routing

This part will describe how multicast set-up and routing are implemented. One aspect that should be kept in mind is that a multicast channel has different requirements than a normal IPP routing of a packet and therefore it needs to be implemented in a different way.

The more general definition of multicast is used. This assumes nothing about what and how many hosts are part of the multicast list of recipients. What multicast requires is to be able to distribute a single packet to multiple recipients in a network bandwidth efficient manner. It is a reasonable assumption that only one host is in charge of adding or deleting other recipients from the multicast list.

A multicast packet is a special packet since it follows different routing strategies than normal IPP packets. It will have its own addressing scheme that is similar but different to IPP. A special multicast addressing scheme results in a simple and fast algorithm that detects if a packet is a multicast packet. Once a packet is recognised as a multicast packet a special routing algorithm can be performed to satisfy the multicast requirements.

A multicast channel is identified by one or more bytes. These bytes do not have any inherent meaning, that is, they are chosen by the system to denote one particular multicast session and they may be reused as soon as the given ID is not in use. A multicast distribution list has therefore two types of addresses.

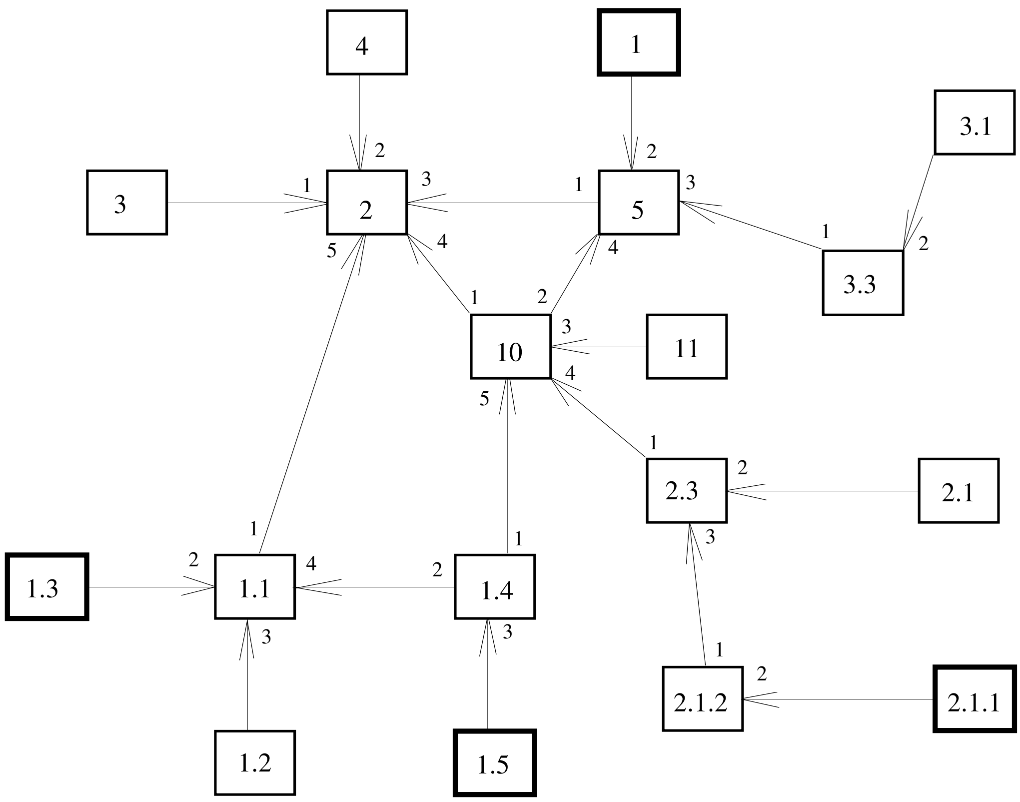
- An address, local to the current domain, that identifies a specific multicast list. This address is the one used by the network and is different for each domain. The mapping between different Multicast addresses in each domain is done at connection set-up. The format of this multicast address is 255.x

- The global address of the multicast list. This address is the one used by the hosts in the network to join or leave the multicast list. It is a symbolic address that is used by the administration programs. This address results from joining the IPP address of the host that maintains the multicast list with the Multicast address of that list in the domain where the multicast list Admin resides. Eg: If the administrator for the multicast list has IPP address 1.2.3 and the multicast identifier in domain 1.2.x for the given multicast list is 255.9 then you can uniquely identify the given multicast list using the address 1.2.3.255.9.

Before going into the details of how a multicast list is set up it is important to introduce the figure of the Domain Admin. The Domain Admin is a host in each domain that is in charge of managing the routing tables and decides policies on the usage of the given domain. It is assumed that the set-up of the multicast routing tables for a given domain would be driven by the creator of the multicast list but managed by the Domain Admin.

3.4.1. Multicast List Set-up

It is useful to see how a multicast set-up is done to have a better understanding of the routing algorithm. Given the following network topology and addresses.



Host 2.1.1 is the creator of the multicast list that includes hosts 2.1.1, 1, 1.3, 1.5. Since 2.1.1 is the creator of the list it is his duty to set-up the multicast routing. The first thing that 2.1.1 does is to request a multicast id for all domains that the multicast packet will travel to. It is possible to do this by looking at the address of the hosts in the list. In this case host 2.1.1 should get a multicast id from the following domains.

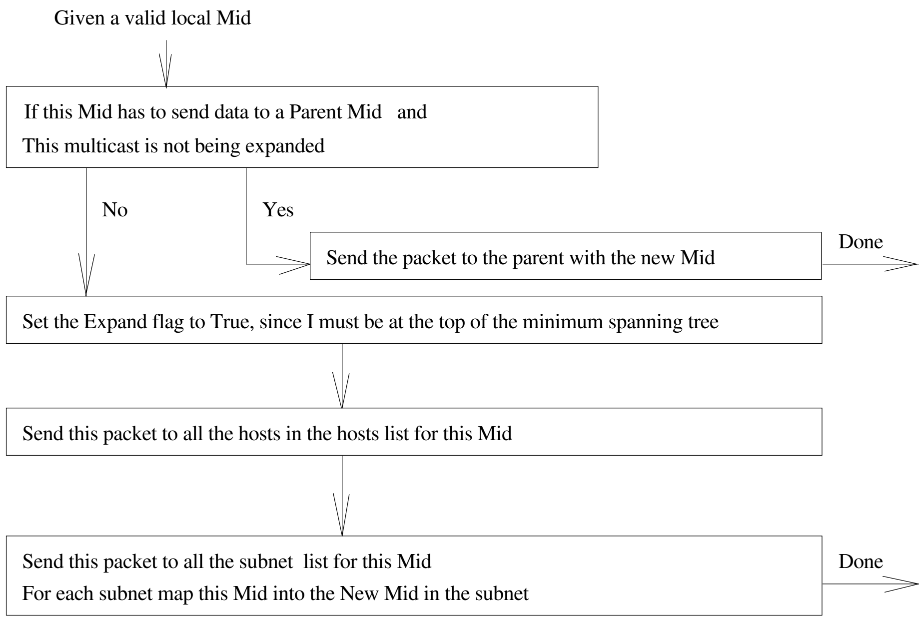
Domain	Obtained Mid
2.1.x	255.1
2.x	255.3
x	255.1
1.x	255.2

Once this information is obtained, the managing host can set the routing table of each domain to reach the desired hosts. This operation will define the actions to do for the given multicast id in the local domain.

What happens is that the Multicast creator sends a request to the domain Admin to route the given multicast id to the specified hosts or subnet. In the case of the above example it is possible to see how in domain 2.1.x we need to reach host 2.1.1 and therefore the Mid 255.1 in domain 2.1.x will include host 2.1.1 as recipient of the multicast. A multicast list that is not used will be cleared by the router administration software. This solves possible deadlocks that can happen if the administrator of a multicast list does not release the allocated multicast identifiers due to exceptional circumstances.

3.4.2. Multicast Routing Algorithm

This part will describe how the routing for multicast packets is performed. It is essential to remember that for every domain where the multicast packet travels there is a possibly **different** multicast id. Each time a multicast packet crosses a domain it has to map its multicast id into the new multicast id in the new domain.



Since the multicast routing table of each router is different, the routing decisions are different for each router.

The routing algorithm does **not** need to search the routing table. It performs the operations previously decided by the Domain Admin and the Multicast List manager.

For further details see the function **DoMulticast**

3.5. Routing of Real Time Packets

This part describes how real time data can be supported by an IPP network. The first thing to recognise is that real time data has a separate class of router scheduling requirements. Real time data requires the travelling time of the packet to be within given boundaries and may require packets to be kept in sequence.

To be able to satisfy these special requirements a domain must be able to decide if it can carry the given real time channel or not. A router must be able to quickly detect this special class of packets and apply the given operations to them.

3.5.1. Real Time connection Set-up

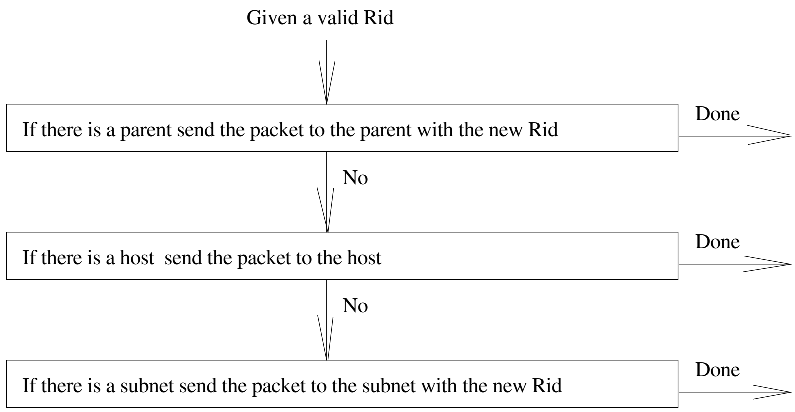
The set-up of a real time connection is driven by the caller. For each domain where the packet will travel the caller will need to ask if the domain can satisfy the minimum acceptable conditions. This is the first step to be able to guarantee prefixed constraints and it is called Admission Control.

If a domain accepts the given real time connection it will return a Real Time Id. A Real Time Id (Rid) is similar to a Mid but uses a different root identifier (254 instead of 255).

Once all domains have granted the permission to carry the given real time connection the caller is in charge of actually asking the Domain Admin to route the given Rid to another Rid in another Domain. Since a real time connection reserves valuable resources, a connection or real time identifier that is not used for a specified amount of time will be released.

3.5.2. Real Time routing Algorithm

The routing algorithm is in charge of detecting a valid Rid and performing the requested operation on it. A diagram of the algorithm is the following.



It can be seen that the algorithm is simple. No time is wasted in searching for what to do. All the decisions on where the data should go and the connection requirements are done at connection establishment.

The router has the ability to put a real time packet in front of the output queue of the desired link if this needs to be done to satisfy the timing constraints. Note that a router has no input queue. This can be done since each interface can be served by a dedicate processor and results in a simpler routing algorithm than if an input queue were provided.

If an input queue was present, the router would need to do a lookup in the input queue looking for real time packets, by having no input queue this lookup is avoided and all timing is done by deciding where to put the real time packet in the output queue.

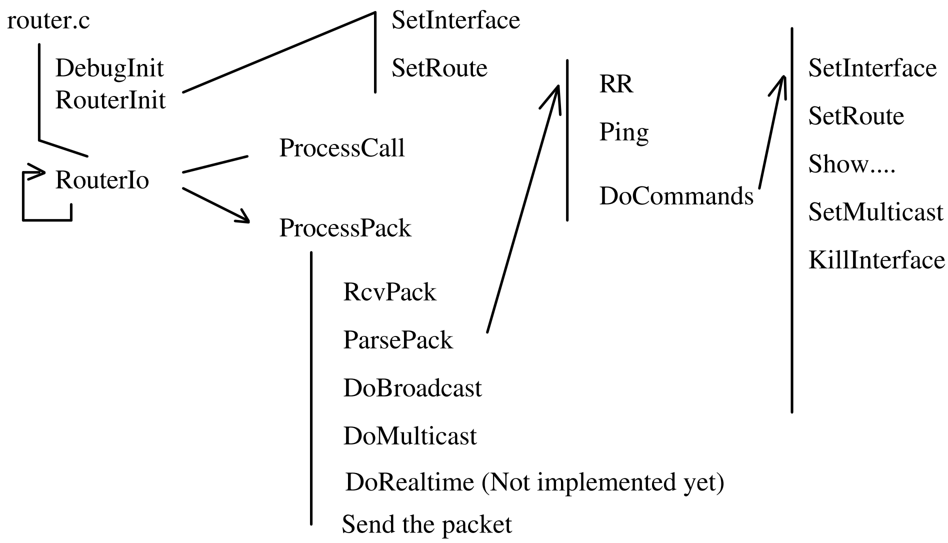
4. Example programs

This section describes the example programs that come with the experimental tool kit. The tool kit is a series of programs that allows the simulation of the behaviour of an IPP network. This simulation is from the routing point of view only. It is assumed that a fully implemented IPP network will implement all features of IP plus the changed addressing and routing scheme.

Note that even if this test implementation does not permit two routers to share one physical line it is obviously possible to do so in a real implementation.

4.1. Router Structure

Before entering into the details of the routing algorithms I will describe the general structure of the router as implemented in the toolkit. The router work is completely driven by network requests. The main loop waits for packets and processes them as soon as they arrive.



The program starts at the **main** function in the file *router.c* it then calls DebugInit, RouterInit and then loops waiting for packets. Packets are received by RouterIo and processed by ProcessPack.

ProcessPack is in charge of detecting what to do with the given packet. In the case when the packet is directed at the router itself the ParsePack will try to do what the packet requests.

4.2. Auxiliary programs

This part will describe the other programs that are distributed with the tool kit and are used to set-up an experimental network structure. All of them are very similar to each other. The differences are in what they allow you to do.

4.2.1. Host

This program provides a very crude simulation of a host. It requires a configuration file that specifies where this host should send packets. What the program does is to send a packet to each of the destinations. It also shows the packets that it receives.

4.2.2. User

It is necessary to provide a way to interact with the network in a non automated way. The **user** program provides this. It is a command line interface that allows you to send different types of packets to the network.

4.2.3. NetAdm

An IPP network requires a Network Administrator program for each domain. This program is in charge of administering the routing tables for the given domain. The **NetAdm** program is just a stub for a *Network Administrator*. The duties of the **NetAdm** are:

- It knows about the topology and link capacity of the domain. It also knows the topology and link capacity toward the parent network and all the Subnets.
- It is in charge of setting up the routing tables for the domain given the above knowledge of the domain topology.
- It is in charge of allocating valid Multicast Id addresses for the given domain. It also handles the set-up of the multicast routing tables after the Multicast List Administrator has given the necessary information to this Domain Administrator.
- It is in charge of releasing Real Time addresses after it has acknowledged that the domain is able to withstand the required data rate with the specified constraints.

It is important to note the difference between the framework of this Domain Administrator compared to the framework of administering the Internet routing table. The amount of information that this Domain Admin needs to handle is limited in size and is localised. This allows for a far better knowledge of the network topology and link type than if this amount of information was not known in advance and spread across various networks.

To avoid wasted resources and possible deadlocks a Multicast Id or a Real Time Id that is not used for a preset amount of time will be released.

4.3. How to build the tool kit

The tool kit is delivered as a tarred and gzipped file. Installation is done by creating a working directory and then extracting the material into that directory. Once this is done the **readme** file in the main tool kit directory can be read and the instructions inside should be followed.

The tool kit and source of this paper can be obtained via anonymous ftp from: unix.hensa.ac.uk
The postscript can be found in: /pub/misc/ukc.reports/comp.sci/reports/9-94.Z
The tool kit can be found in: /pub/misc/ukc.reports/comp.sci/reports/9-94.app.tar.Z

4.4. How to use the programs

This section explains the basic structure of the tool kit and shows how it can be used. The tool kit parts are divided in different sub directories each one holding part of the system. The most useful part is probably the **logd** demon. This program is under the logd directory listens for packets on udp port 1234 of the local machine. The other parts of the tool kit will then send data to this daemon when they need to display debugging information.

The second most important program is the router. The router requires the configuration file that specifies its address, the ports to connect to, the interfaces to create and the routing table specification. This file must be given on the router stdin. Once started a router will output on stdout the "interfaces" it has created and then starts accepting packets from the network.

Connection to a router can be done by using the program **mon** under the monitor directory. This is an interactive program. It will ask for the IP port to connect to. This is like a telephone number to dial to connect to the router. This information can be found by looking at the router output after it has been started. Mon will then ask for its own address and for the address of the router in this domain. This is the single trailing byte of a router address.

Once this information is given the mon will wait for packets to display from the router or for user commands. A typical command is to ask to the router to show its routing table. Using the topology shown in example two the monitor has address 1 and the router has address 2. To show router 2 routing table type:

Destination> 2
packet Type (Data, RecordRoute Ping) > d
data> show route

If something goes wrong the logd should explain why.

The other program that is used in this version of the tool kit is the **host**. A host is a simulation of a host behaviour. It is configured by a config file passed in stdin and what it does is to send data to the specified destinations.

4.5. Available commands on the Router

A router will respond to commands that are directed to it. What follows is the list of the available commands and a short summary of the effects. Some commands are self explanatory and are therefore not described. It must be noted that words that are written in Italic should be replaced with an appropriate value, words that are written in bold should be written as they are. An example of usage of the following commands can be found in the initialization scripts for the routers in the various examples.

The command **set route parent** has two possible syntax's. The first one is used to specify a direct route to a parent. The second one is used to specify a distant route for the parent domain. Note the similarity to the **set route net** command.

show	**route**					
show	**multicast**					
show	**interface**					
show	**stat**					
set	**route**	**parent**	*NextHop*	*Interface*	*Cost*	
			NextHop	**Remote**	*Cost*	
set	**route**	**host**	*HostId*	*NextHop*	*Interface*	*Cost*
set	**route**	**net**	*NetId*	*NextHop*	*Interface*	*Cost*
			NetId	*NextHop*	**Remote**	*Cost*
set	**multicast**	**mid**	*Maddr*	*AdminHost*		
set	**multicast**	**expire**	*Maddr*	*Seconds*		
set	**multicast**	**parent**	*Maddr*	*ParentMaddr*		
set	**multicast**	**host**	*Maddr*	*HostId*		
set	**multicast**	**net**	*Maddr*	*NetId*	*NetMaddr*	*RouterId*
set	**interface**	*IfNum*	*Tech*	**Listen**		
set	**interface**	*IfNum*	*Tech*	**Connect**	*ToLevel*	*Parameters*

set	**brdhost**	*HostId*
set	**debug**	*Debug-Level*
kill	**interface**	*IfNum*
kill	**router**	
get	**mid**	

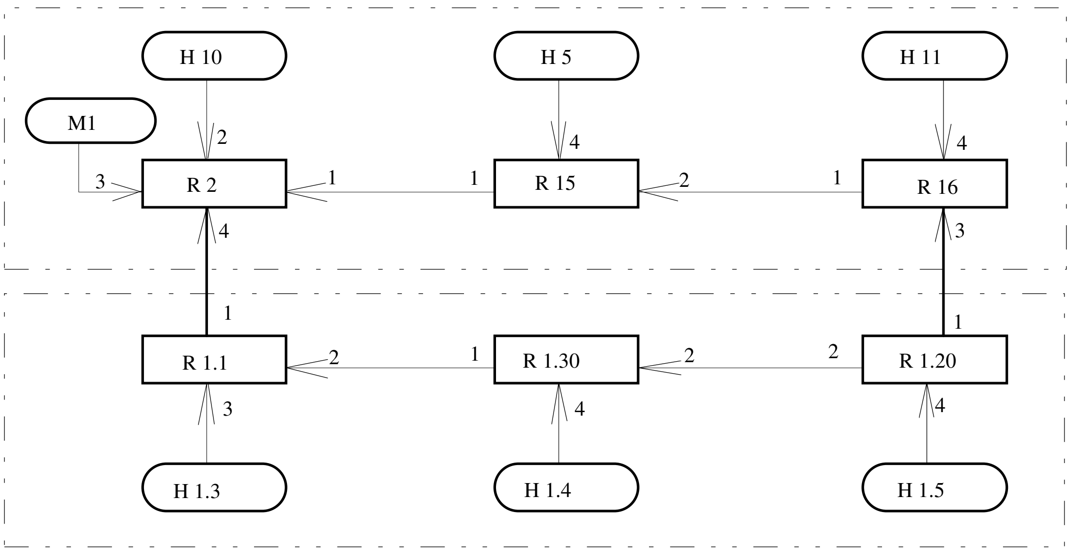
4.6. Example One

This example shows how a Subnet can be connected to a parent by two routes and how load balancing can be performed on the two links.

There is a parent network composed of three routers, three hosts and a Subnet. The Subnet is also composed of three routers and three hosts. The hosts send data to each other and the option Record Route will show the path taken by the packets.

The numbers on the links indicate the interface that the router is using to connect to the other end. The arrow indicates what router is the caller and what is the listener in the action of setting up a link.

The routers are prefixed by the letter **R,** the hosts are prefixed by **H** and the monitor is prefixed by the letter **M**.

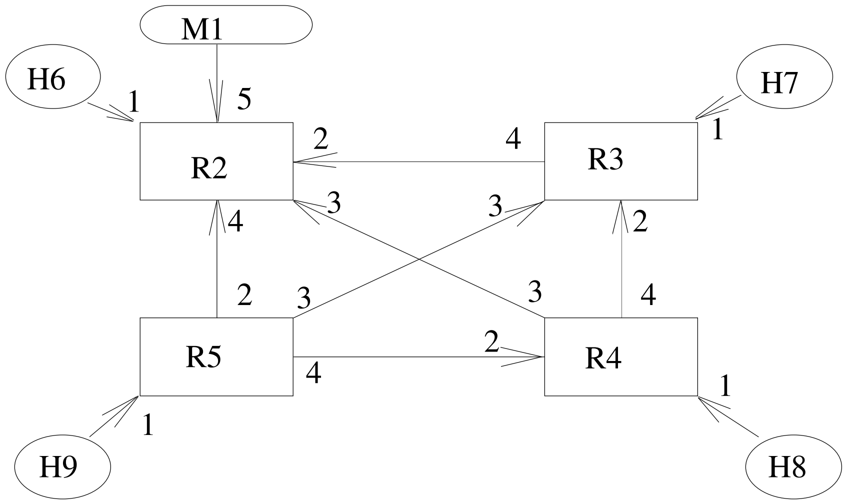


This example can be run using the **run** command in the config/ex1 directory. Note that if something goes wrong, possible causes may be the fact that the X screen is too small or that the PATH is not correct. In this case the system can be built by hand by starting one router at a time and understanding what the system is doing.

The **logd** daemon should always be started first since this will probably show what went wrong.

4.7. Example Two

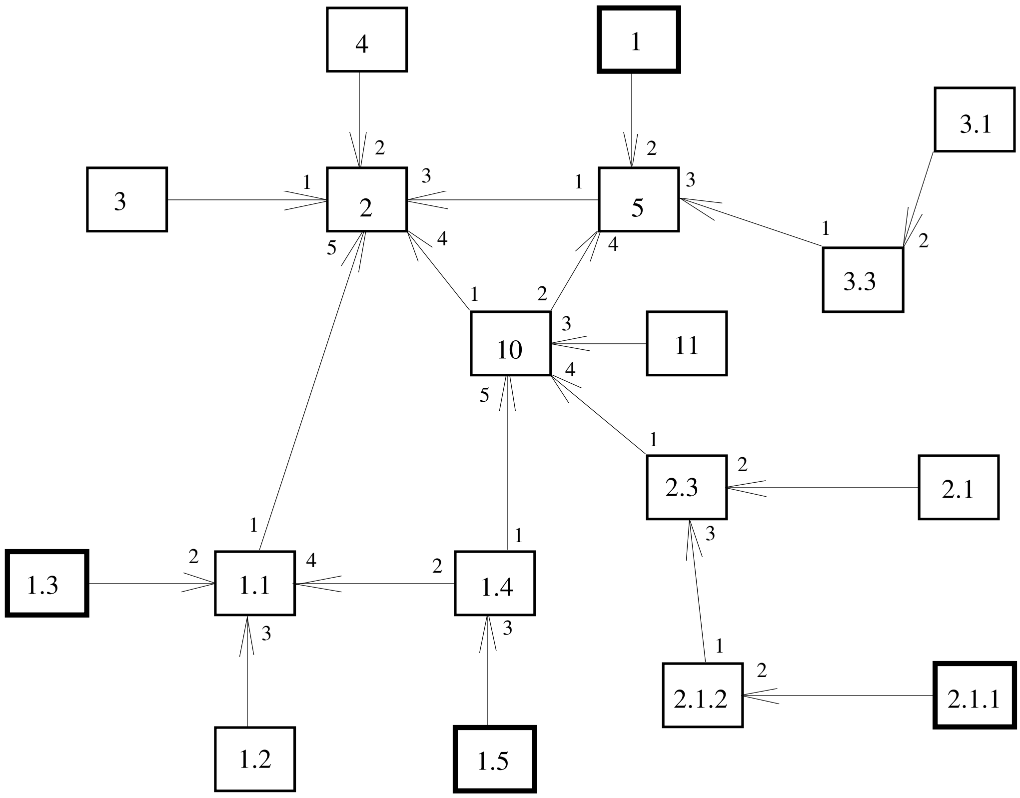
This example shows how it is possible to achieve load balancing at the same domain level. It also shows how this network configuration will lead to routing loops only in certain cases. In the case of routing loops it can be seen how many packets actually get discarded due to a real loop and how many still get to the destination.



The arrows indicate if the destination interface is a listening interface or not. Eg: Host 9 connects to Router 5 at interface One of the router that is a Listening interface.

4.8. Example Three

This example shows a network with a multicast list already set-up. The multicast host list is composed of hosts 1 1.3 1.5 2.1.1



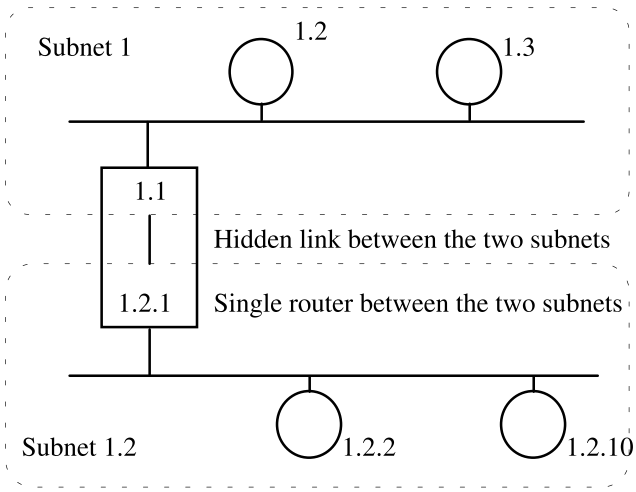
You can run this example by using the run command inside the config/ex3 directory.

5. Side Issues

This section describes the parts of the project that are not yet implemented but that are planned.

5.1. Two routers in one

It can be seen that there is always a link between a Parent and a Child domain. This link is in most cases real but in other cases can be imaginary. A typical case is when two Subnets are connected with only one box between them. In this case the box would contain two routers with a hidden link. It will appear as two routers and one link from the network point of view but it will be one single box and no wires in practice.



5.2. Migration Path

The huge address space of IPP should not be wasted in a very unbalanced tree. It is therefore advisable to specify different Subnets depending on regional areas and other Subnets for organisations that would like to keep their own network separate.

A possible structure would be

Subnet	Subnet	
1	USA	
	1	Texas
	2	Minnesota
	3	North Dakota
	....	
2	Europe	
	1	England
	2	Italy
	3	Germany
	....	
3	Australia	
4	Russia	
.....		

Subnets can be allocated to companies. Something like.

5	Company	
	1	Motorola
	2	IBM
	3	HP
	....	
6	Packet Radio	

It is possible to see that the top level remains quite sparse. Again this choice is debatable and can be changed.

To migrate from IP to IPP a means must be provided to transparently obtain the IPP addresses of a host. It is possible to modify the domain name server bind to return IPP addresses as well as IP if queried for. In the transition period the same physical links could be serving both IP and IPP depending on the address supplied.

It is possible to say that the address of something is determined by where it is logically connected. If for example a Host or Router connects to a router that has address 1.2.3 the address can only be either a parent of 1.2.3 or in the same domain as 1.2.3 or a Subnet of 1.2. This scheme is like the road system where your home address is determined by where your house is in relation to a road.

6. Conclusion

A connectionless approach to network routing gives advantages in terms of load balancing and reliability of a network connection. It is also possible to make the choice of a route an extremely quick process if the tables that are being "searched" are direct tables. Having a fixed size, direct lookup, routing table allows the construction of very high speed routers. This is possible since most of the routing functions can be built in hardware.

The fact that the network administrator of a domain only sees and cares for a limited part of the network should make network administration easier and more efficient possibly reducing the downtime of routes.

A variable length, limited, address space gives greater possibility to organise the structure of the network in a logical way and this will result in less illogical routes taken by packets when trying to reach the desired destination.

7. Glossary

It is useful to clearly specify words that have different meaning in different context.

- **Domain**: Defines the group of hosts and/or routers that are within the same subnet. Note that this does **not** include the hosts and routers that are below any of the subnet of the given domain. Given the structure of IPP it follows that a domain can have a maximum of 250 hosts and/or routers.

8. References

[leis-92] Charles E. Leiserson, 1992: "The Network Architecture of the Connection Machine CM-5," *Thinking Machines Corporation Cambridge, Massachusetts* April 27, 1992

[Scott-94] Scott Shenker, David D. Clarck, Lixia Zhang, 1994 "A Scheduling Service Model and a Scheduling Architecture for an Integrated Services Packet Network", MIT Electronic Paper.

[Davi-88] John Davison, 1988 "An Introduction to TCP-IP", *Springer-Verlag* ISBN 3-540-96651-X

[Raj-92] Bala Rajagopalan, 1992 "Reliability and Scaling Issues in Multicast Communication", *Computer Communication review*, Vol 22, No. 4 pp.188-198.

9. Bibliography

[zaum-91] William T. Zaumen, J.J Garcia-Luna Aceves, 1991: "Dynamics of distributed shortest-path routing algorithms," *Computer Communication review*, Vol 21, No. 6, pp.31-42

[tsu-91] Paul F. Tsuchiya, 1991: "Efficient and Robust Policy Routing Using Multiple Hierarchical Addresses," *Computer Communication review*, Vol 21, No. 6, pp. 53-65

[est-91] Deborah Estrin and Martha Steenstrup, 1991: "Inter Domain Policy routing: Overview of architecture and Protocols" *Computer Communication review*, Vol 21, No. 1, pp.71-78

[wang-92] Zheng Wang, 1992: "Analysis of Shortest-Path Algorithms in a Dynaming Network environment," *Computer Communication review*, Vol 22, No. 2, pp. 63-71

[Estr-92] Deborah Estrin, Yakov Rekheter, Steven Hotz, 1992: "Scalable Inter-Domain Routing Architecture," *Computer Communication review*, Vol 22, No. 4, pp.40-52

[bahk-92] Saewoong Bahk, Magda El Zarki, 1992: "Dynamic Multi-path Routing and how it compares with other Dynamic Routing Algorithms for High Speed Wide Area Networks," *Computer Communication review*, Vol 22, No. 4, pp.53-64

[tsu-92] Paul F. Tsuchia, 1992: "Internet Routing over Large Public Data Networks using Shortcuts," *Computer Communication review*, Vol 22, No. 4, pp.65-75

[bil-91] Paul Bay and Gianfranco Bilardi, 1991: "Deterministic On-Line Routing on Area-Universal Networks," *Proceedings of the 31 Annual Symposium on Foundations of computer Science (St. Louis, MO, Oct 22-24)* pp. 297-306

[gaug-93] Patrick T. Gaughan and Sudhakar Yalamanchili, 1993: "Adaptive Routing Protocols for Hypercube Interconnection Networks," *Computer*, May 1993, pp. 12-23