

# Internet Infrastructure

K. U. Leuven 2009-2010

## Part 1: Networking

Prof. dr. ir. A. Mariën

## Table of Contents

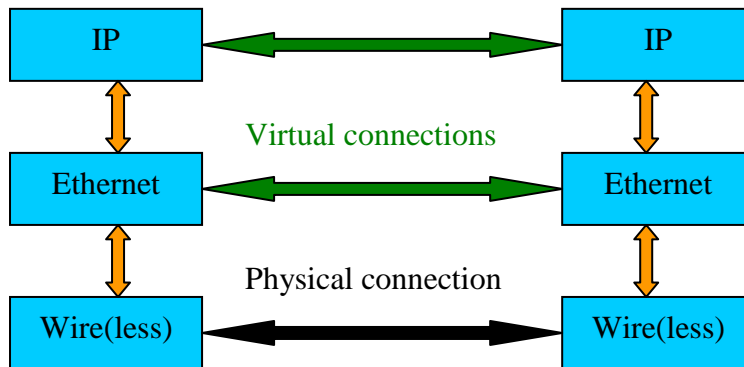
Table of Contents .....	2
Introduction .....	4
Network configurations .....	4
Components – physical layer .....	5
Repeater .....	5
Hub .....	5
Components: Layer 2 LAN standards .....	7
Standards .....	7
Bridge .....	7
How to deal with mixed environments? .....	7
Layer 2: Ethernet .....	8
Layer 2: Switch .....	8
Introduction .....	8
Switch properties .....	9
Store and Forward .....	9
Learning cache .....	10
Problem: Two identical MAC addresses .....	11
Problem: Packet storms .....	11
Questions .....	11
Spanning Tree Protocol .....	11
Virtual LAN .....	15
Network traffic inspection – Ethernet .....	17
Pcap – packet capturing API .....	17
ethereal .....	17
Layer 3: IP .....	22
IP essentials .....	22
IP addressing .....	22
Multiple LANs .....	22
Multiple interfaces .....	23
Note .....	24
Node – interface – LAN – IP address .....	24
Inter-LAN communication .....	24
Communication examples .....	25
Communication on the same LAN .....	25
Communication with the router .....	25
Communication across the router .....	25
Exercise .....	27
IP – ICMP - UDP – TCP .....	28
Internet Control Message Protocol: ICMP .....	28
TCP and UDP .....	30
User Datagram Protocol: UDP .....	30
Transmission Control Protocol: TCP .....	30
Fragmentation .....	31
Windows .....	31
Duplicate packets .....	31
Routing .....	31
Static (Configured) routing .....	32
Routing protocols .....	33

IP – Ethernet.....	33
Address Resolution Protocol: ARP protocol.....	35
Network communication: IP and Ethernet interactions .....	36
Network Address Translation.....	36
NAT intern to extern .....	37
NAT: extern to intern .....	37
Automatic Configuration.....	39
Automatic Configuration: What is the problem? .....	39
Parameters .....	39
Link-layer .....	39
IP-layer .....	39
Routing .....	39
TCP.....	40
Per host:.....	40
Auto Configuration: alternatives .....	40
RARP and DRARP .....	41
BOOTP.....	41
DHCP .....	41
DHCP: IP to host mapping.....	41
DHCP - BOOTP: relations .....	42
DHCP - BOOTP: primary differences .....	42
DHCP: the basics .....	42
DHCP: Client-Server Protocol .....	42
Message format .....	42
Chicken&Egg problem.....	43
Chicken&Egg solution .....	43
Security of DHCP.....	43
DHCP relay .....	44
RFC 1542: Clarifications and Extensions for the Bootstrap Protocol .....	44
Fields in packages .....	45
Use of fields .....	45
Gateway address usage in request.....	45
The server reply handling.....	45
The actual scheme: .....	46
IP Multicast .....	46
Motivation .....	46
Definition .....	46
Sending.....	46
Receiving.....	47
Ethernet link .....	47
Routing .....	47
References .....	47

## Introduction

The network architecture is based on a layered approach. In network literature it is called a network stack. There are two models. The first is the formal, complete ISO model with its seven layers. The second is the model underpinning most implementations today, the TCP/IP and Ethernet model. For the purposes of this course we will use a simplified model that is directly related to the latter.

For the basic layers we restrict ourselves to the physical layer, on top of which we run an Ethernet layer, on top of which we run the IP stack:



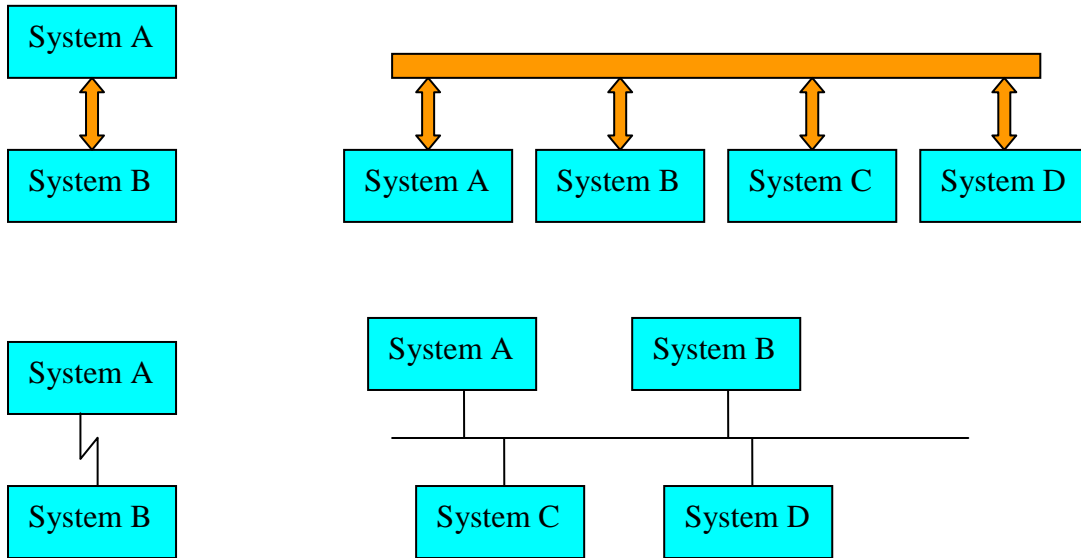
## Network configurations

The simplest network is one with two components, a peer-to-peer connection. In such connections there is no need to include any form of addresses: both parties now there can only be one other party at the other end.

The extension to a “bus” network with multiple components is the next step. In principle, every system on the bus can read any communication on the bus. This can be advantageous when the inherent parallelism can be used to advantage, like broadcasting. It does require the introduction of addresses to deliver directed messages.

When drawing network diagrams, the level of detail varies depending on the purpose of the drawing.

The diagrams may look like these:



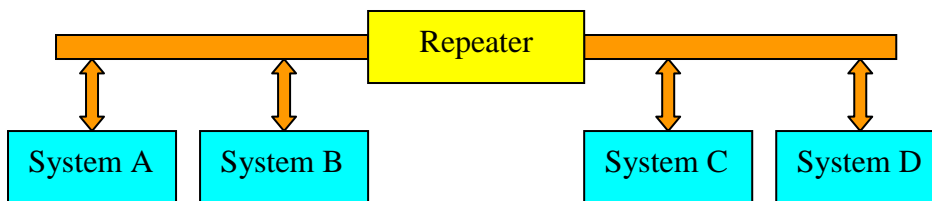
## Components – physical layer

### *Repeater*

Some components are only concerned with the lowest layer, the “wire” level.

Wires have restrictions. If too many systems are connected, problems like collisions, lost packets, monopolizing etc. may occur.

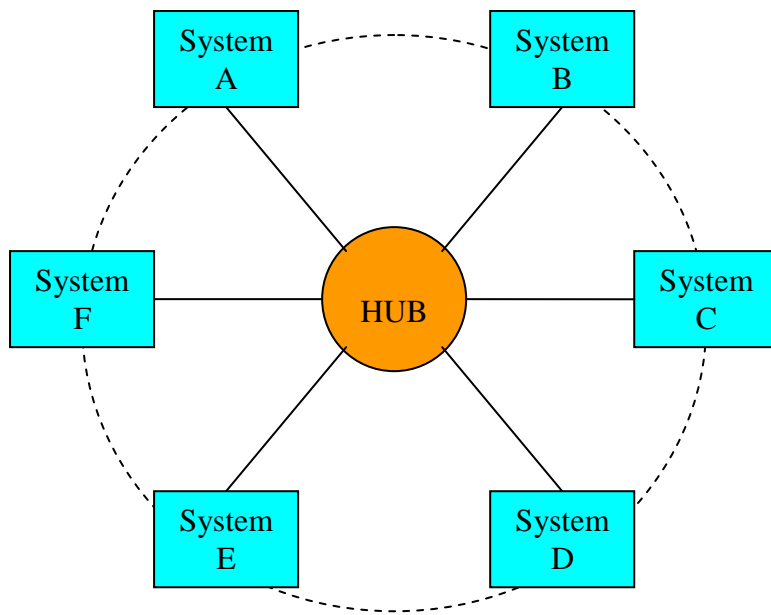
If the distance between the components becomes overly long signal strength, noise, etc. become important disturbing factors. A repeater is put in the ‘middle’ of a wire to address these problems.



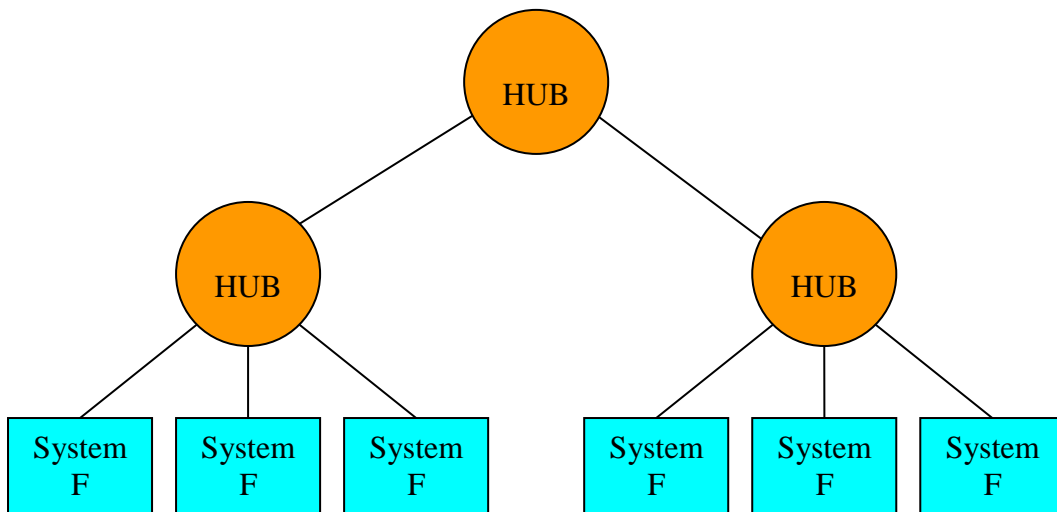
Repeaters are not very common in today’s networks.

### *Hub*

In many logical diagrams, three or more components must be interconnected. Cables, however, are usually point-to-point. To add components, one could use ‘T’ pieces, creating drop points or split points. A more general solution is to link all components to a ‘hub’, using point-to-point cabling. The following picture shows why it is called a hub: the wires are the spokes of the wheel; the Hub is, well, the hub.



Hubs can be chained in a tree-like structure:



Hubs were very common, and can still be found in many places, because they allow for easy interconnection and are relatively cheap. Nowadays, they are getting replaced by more intelligent devices, as the prices of such components have dropped significantly.

Don't make loops with HUBS. Guess what happens.

## Components: Layer 2 LAN standards

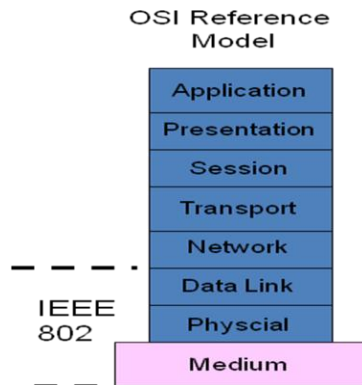
### Standards

The scope of the main standardization body, the IEEE 802 committee, is concerned with OSI layers physical and data link, and media.

(<http://www.ieee802.org/>)

The following diagram is taken from:

([http://www.ieee802.org/newcomer\\_orientation-v3.ppt](http://www.ieee802.org/newcomer_orientation-v3.ppt))



IEEE 802 standards

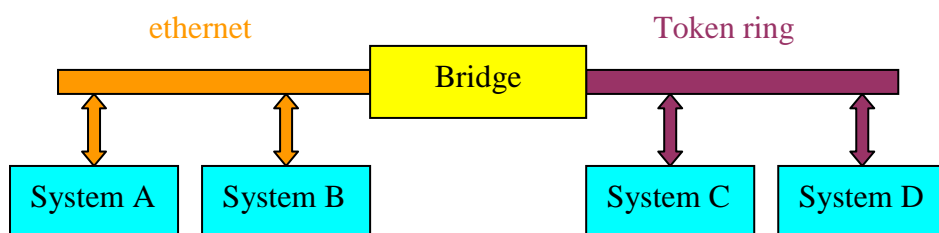
- 802.1: General issues
- 802.3: Ethernet
- 802.5: Token ring
- 802.11: Wireless LAN
- 802.15: wireless PAN
- 802.146: WiMAX

Ethernet: CSMA/CD: Carrier Sense, Multiple Access, Collision Detection  
Ethernet and Token ring are competing technologies (sometime, somewhere).  
IP can run on top of both.

### Bridge

#### How to deal with mixed environments?

A bridge solves the question on how to deal with mixed environments at the Ethernet-token ring layer. The technologies must be similar enough to allow more or less transparent cooperation..



## Layer 2: Ethernet

Ethernet and variations are becoming the de-facto standard. Other technologies have become nearly extinct. As a consequence, more and more functionality shifts to the Ethernet layer, which enforces the push to standardize on Ethernet even more.

Ethernet packets contain two main elements: first, address information, namely a 'to' address, and a 'from' address, and second, the data that is sent.

The Ethernet addresses are 48 bits long. The Ethernet addresses are assigned to Ethernet devices by the manufacturer. The manufacturer gets a block (or more) of MAC addresses he can use. Therefore there should never be a conflict of addresses.

It is possible to change the MAC address, either inside the hardware or in the software. It also depends on the system how easy it is to reconfigure the MAC address. Whether this is a good idea is another question.

An Ethernet packet has a maximum payload of 1500 bytes.

There can be a maximum of 1024 interfaces on one Ethernet segment.

Ethernet supports three communication methods

- Directed: a packet is addressed to one specific host
- Broadcast: the packet is intended for all hosts in the broadcast domain
- Multicast: the packet is intended for all host in the multicast domain

Note that the technology itself assumes a form a broadcasting, so the other communication methods are logical only.

The broadcast domain includes all systems on the network, as seen by the Ethernet layer.

Multicast communication is to multicast domains. These are possibly overlapping subsets of the systems in the broadcast domain. The model behind it is a form of publish-subscribe:

systems register for multicast communication, and publishers publish to any registered

listeners. Directed is mimicking point-to-point communication over a broadcast medium.

The three types of communication are distinguished by the addresses. A part of the address space is reserved for multicast and broadcast.

(see <http://www.iana.org/assignments/ethernet-numbers>)

Ethernet cards typically filter the traffic: only broadcast, multicasts for which there is a subscription, and packets with the right MAC address are further processed. In some cases it is necessary for a machine to listen to all traffic. The cards support this mode of operation.

The card is said to be in 'promiscuous' mode. This mode can also be abused to listen in to other peoples communication.

## Layer 2: Switch

### *Introduction*

When adding more and more components to a network, problems start to occur. For token ring, the number of members of the ring is limited. For Ethernet, more and more collisions occur, leading to retransmission, leading to network congestion.

There are other problems as well. When two systems on the network produce heavy communication between them, all other systems that try to use the network suffer from their traffic. The confidentiality problem of the communication may also have to be solved.

A hub cannot solve the above problems. More understanding of the higher layer, Ethernet, is required to provide solutions. The hub connection diagram on the other hand, is well suited: all systems have a private connection to the Hub. That allows in principle to limit traffic on the connections starting from, or with destination to, the connected system.

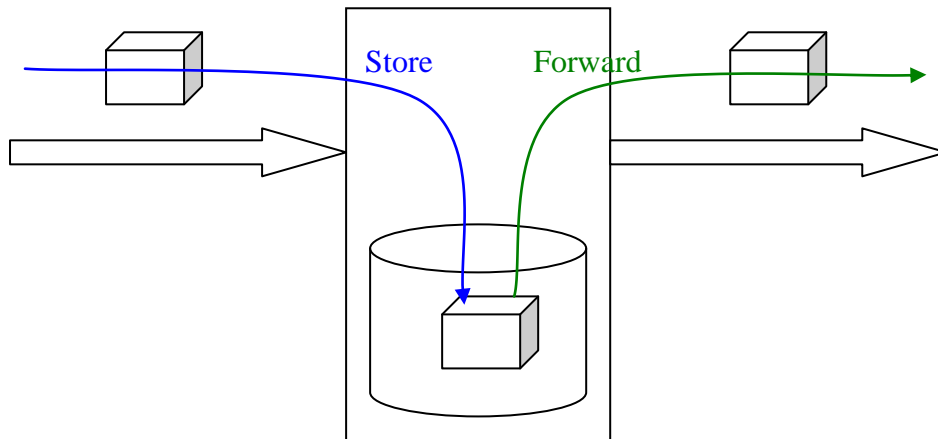
A new device is required. It should know about which interface is where, so that directed communication can then use only the wire to that device. It should learn about the devices without having to resort to configuration. The device will have to understand Ethernet so that it can use Ethernet addresses inside Ethernet packets to guide its operation.

### ***Switch properties***

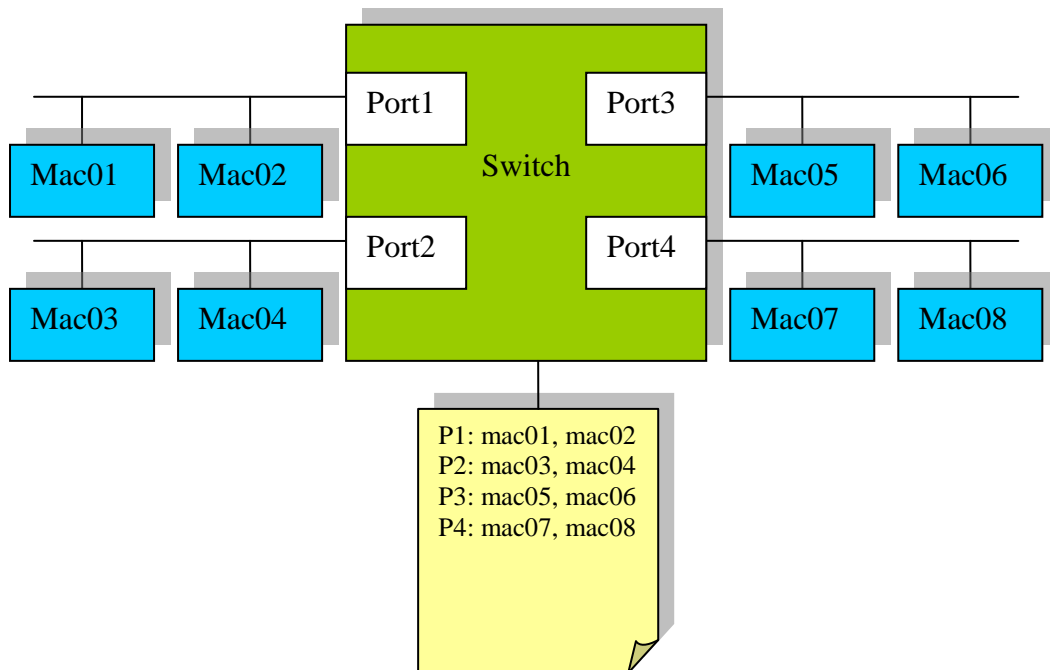
IEEE standard 802.1d defines the properties of a switch.

- Store and forward
- Implement a learning cache of addresses per port
- Avoid packet loops via spanning tree computation

### **Store and Forward**



## Learning cache



The switch learns on which port which MAC addresses are. It builds up a cache with these entries. Like with all caches, the data in it may expire, the data may be replaced with newer entries, or more frequently used entries.

The switch learns the MAC addresses by inspecting all packets and looking at the 'from' addresses.

If a switch receives a packet with source 'mac<sub>source</sub>' and destination 'mac<sub>destination</sub>' on port 'p':

- It updates the cache with the entry: 'mac<sub>source</sub>' is on port 'p'
- It searches the cache for the port number 'q' associated with 'mac<sub>destination</sub>'
- If no port 'q' is found
- Then
  - It sends the packet on all ports but 'p'
- Else
  - It sends the packet only on port 'q'

Some consequences:

Initially, the switch acts like a hub. It has no knowledge of any locations of addresses, so it will have to forward packets to all but the incoming port, which is the same as a hub would do.

A switch may return to the hub behavior. A switch has a limited number of storage. At some point, it will have to purge information from the store to make room for new data. In such a case, when the address-port mapping cache is full, the switch broadcasts to all addresses not in its cache.

The switch may allow multiple parties in the same LAN to communicate at full speed, as if they were the only parties on it. Even if two parties communicate a lot, their traffic is limited

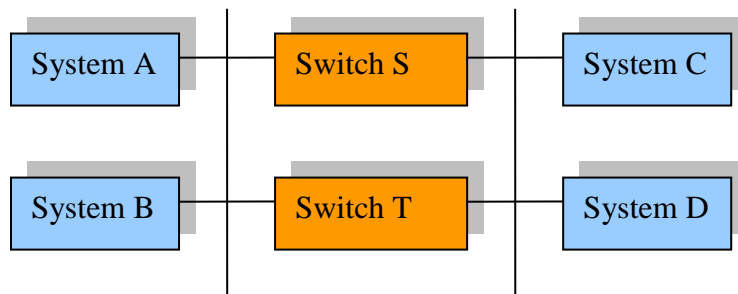
to the source and destination ports and the other ports do not see this traffic. This effectively increases the bandwidth of the network.

### Problem: Two identical MAC addresses

If system A has address MAC, and system B has the same address MAC, but on different ports, the switch will very likely get confused.

When it receives a packet from A, it decides MAC is on port p, when it receives a packet from B, it decides MAC is on port q. It will only send packets to the port on which it last received a package, so system C may not be able to reach B if A was the last to send a packet.

Another way that this problem may occur, is a typical HA set-up. Switches can connect to other switches. In fact, such set-ups are rather common to avoid a single point of failure. In so called redundant set-ups, each component is doubled. If both somehow become active, it goes wrong.



System A sends a packet to C.

Both switches S and T pick it up on their left port, update their caches, and send it on their right port.

Both switches now pick up a packet from the right port, coming from A. They both update their caches and decide A is now on their right port. When C replies, the switches do not react: A is on the right port, and the packet is on the right port, so A will have seen it.

### Problem: Packet storms

In contrast with other technologies, switches duplicate packets: one incoming packet can be sent to multiple ports. If such a packet reaches another switch, the same duplication may happen. If one of the duplicated packages of the second switch reaches the first again, another duplication round is started, leading to ever more and more copies of the package.

Any loop in switches may cause this behavior. One solution is to make sure there are no such loops. That can be hard if you want to achieve high availability.

The other solution is to reconfigure the switches to reduce the graph to a tree.

### Questions

Can hubs also produce packet storms? Compare how Hubs and Switches are similar/different with respect to this problem. The solution for switches (STP) is next. Is something similar conceivable for hubs?

### Spanning Tree Protocol

The main goal is to transform a graph to a tree. That way, all LANs are still connected and there are no loops. To create a tree from a graph with loops, some ports on some switches are

disabled. On each LAN there will be one active switch, which is the designated switch on that LAN. The ports on the other switches connected to this LAN are deactivated. On each switch, there is a port with the best path to the root switch.

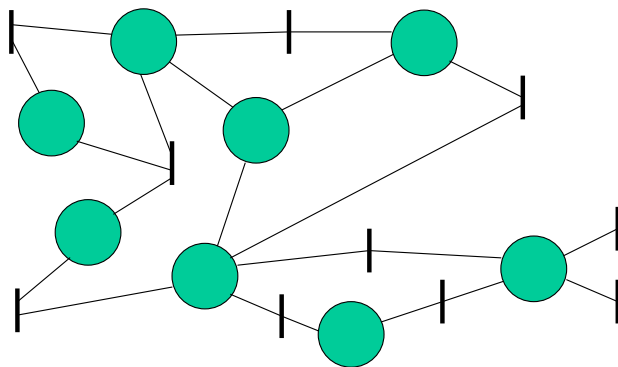
The algorithm decides on the root switch, and determines for each switch the cost of getting to the root bridge.

All switches are numbered. The switch with the lowest number becomes the root bridge. Each switch considers itself the root bridge, cost 0. They broadcast that message. When they learn there is a better choice for the root switch, or that is a lower cost path to the actual root switch, they update their information and broadcast it.

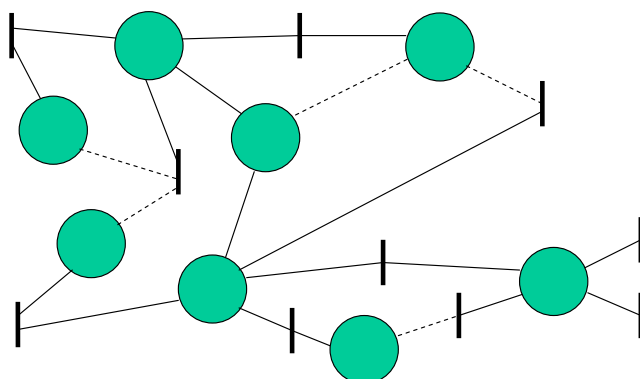
The direct neighbors of the root switch learn the correct root switch after step one, those connected to them after step two, etc. They also learn the cost associated with that path. Longer paths with lower cost may occur in later steps, but the root switch is stable. The number of steps need is the depth of the tree.

The switches remember the best path per port. If some other switch has a better path to the root on the port, the port is disabled. There is one switch that is closest to the root switch; all others on that LAN disable their ports. That switch becomes the designated switch on that LAN.

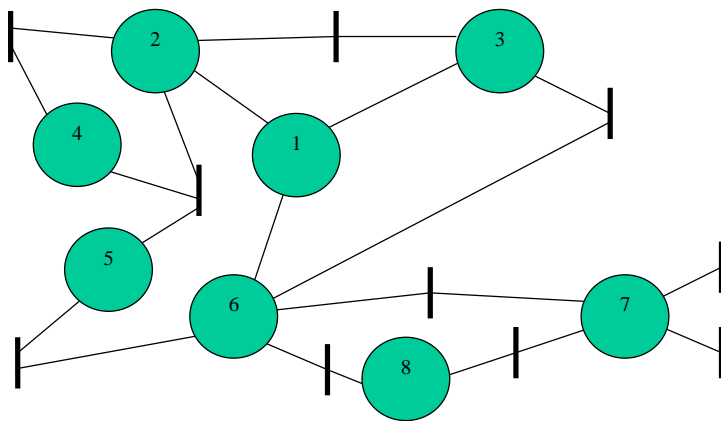
Sample network (taken from the book: interconnections):



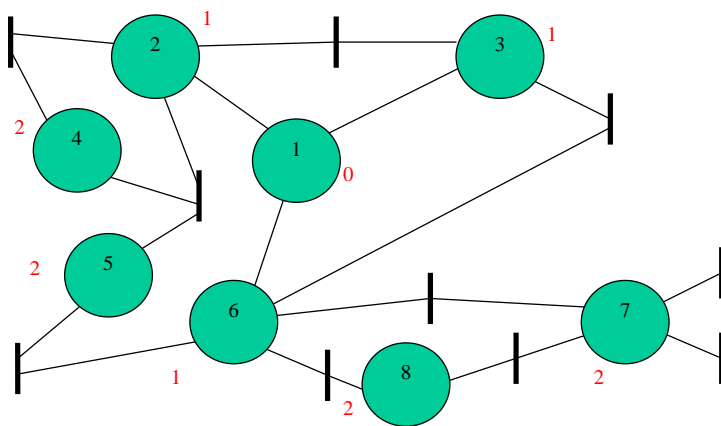
Loop free:



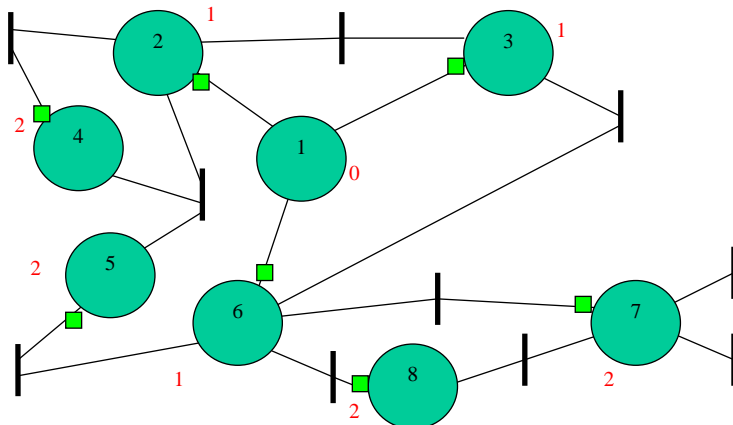
Labelled switches:



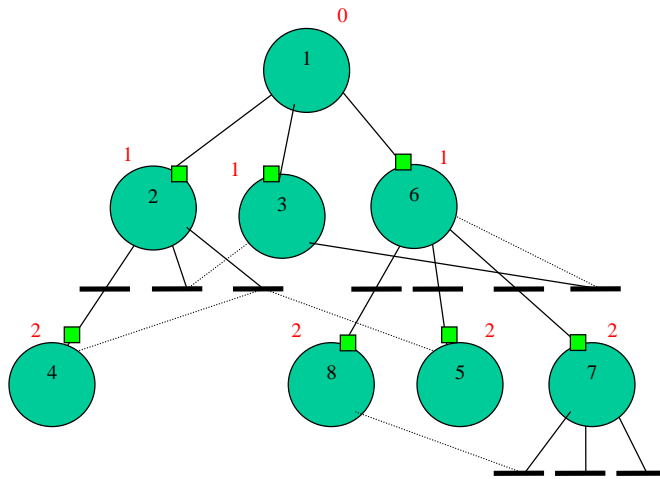
Costs to root bridge:



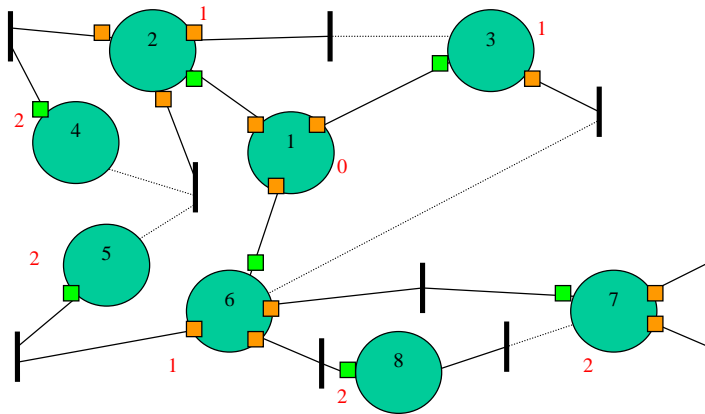
Path to root bridge:



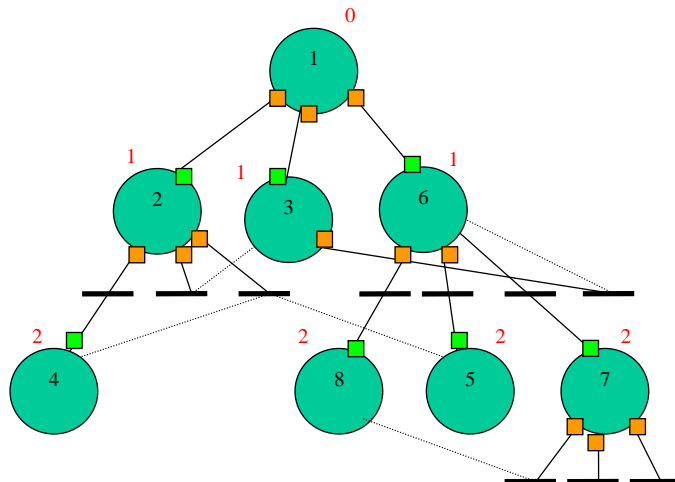
The path to the root bridge is the shortest path. Each step decreases the cost to get to the root bridge. In the simple cost scheme, it goes from 2 to 1 to zero. Looking at it alternatively, the costs in this simple scheme reflect the depth in the tree.



Designated bridge:



Again, the tree view makes it more clear how the system is working.



Packets are propagated upwards via the path to the root bridge, and are distributed down the designated switch ports.

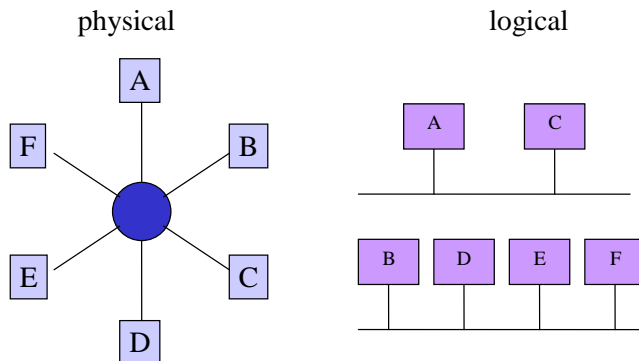
## Virtual LAN

A virtual LAN (VLAN) provides a high level of flexibility for the configuration of the network in LANs, with no direct correspondence with the physical location of the components. It is therefore also a solution for the mismatch between the physical location of components and the logical organization.

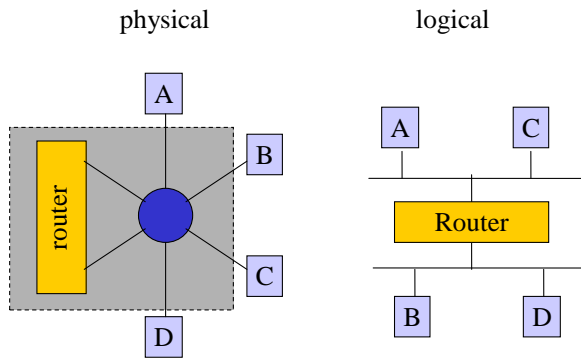
In an organization the logical way to assign LANs is based on the organizational unit. The requirements within Human Resources, finance and administration in general are different from the people in the front desk offices, or in the engineering department. These differences may found in the use of the same shared resources within the group but different from the others, in the need to have guaranteed bandwidth, and the shielding of sensitive data to a group. In larger organizations there are (also physical) reorganizations with a high frequency: offices get assigned to other groups, offices are merged or split, etc. In view of these changes it becomes hard to match the physical network topology to the logical requirements.

A part of the solution for the problems is the use of VLANs. Above layer 2, the network behaves as if the topology matches the requirements; However, this impression is created on level 2, as level 1 connections do not enforce it.

Another way to approach the need for VLANs is that when you have multiple LANs, one could use a switch for each. However, it is more economical and practical to merge the switches physically into one, while make them logically behave as if it are more. You gain a lot: easy to switch a system to another LAN, distribution of systems across switches can be very flexible, and cables go to the same component.



The consequence of the virtual solution is that a nice logical diagram can in fact, physically, look rather different. In the next logical picture, the router sits in between the two LANs. It looks impossible to pass from one LAN to the other, without passing the router. In the actual connection diagram, all devices are hooked to the same switch. Traffic from A to B does not have to pass via the router.



## VLAN

To create a VLAN, one must provide a single broadcast domain per VLAN. Any packets on the VLAN should mimic those inside a real LAN as far as any node in the VLAN is concerned.

### Packet sniffing

Packet sniffing is the technique with which one listens in on communications not intended for the listener. On layer two, devices should drop messages with a MAC that is not a broadcast or multicast and not matching their MAC address. However, it is sometimes necessary to listen to any communication on the line. Therefore, network cards foresee the option to put the card in promiscuous mode: listen to any traffic on this network.

In principle, packet sniffing will work within the VLAN only. After initial learning, a switch limits traffic on a port to the messages with a destination address on that port. This reduces the options to sniff other traffic.

The solution built into switches is a special port that will carry all the traffic of the switch, including all VLANs. A particular concern is that the total traffic of a switch with 100Mb ports can exceed this amount of data, so a 1Gb link for the SPAN port would be required.

### Configuration of VLANs

There are different solutions to decide which targets are on which VLANs.

#### Port based

The ports on the switch are partitioned in consecutive ranges, each defining a VLAN. For instance: 1-i, i+1-j, j+1-n: 4 VLANs.

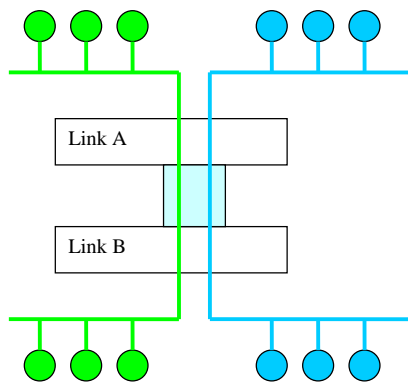
One can also define on a per port basis which VLAN it belongs to.

More complex configuration may be possible. One can define a list of MACs per VLAN. If a MAC is associated with a port, that port becomes part of the corresponding VLAN.

One can even go further: the IP address associated with a MAC that sits on a port is assigned a VLAN.

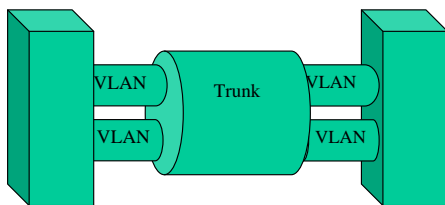
### Tagged VLAN

There are a few other situations where virtual LANs come in handy. It may be necessary to interconnect two different locations with each more than one LAN, where the two sites have stations that should be in the same LAN. A typical case is a high speed link (gigabit or fiber) between two sites, that each have a number of LANs, where it would be convenient to have some virtual LANs spanning the two sites.



The single link between the components carries the traffic for both the VLANs. To enable this, all packets on this linked are tagged with the VLAN they belong to. The Link A & B components do not have to be switches: any components that can handle the extra tagging can use this technology.

Often, the components will be switches. The end of a tagged link can be interpreted as a set of virtual ports.



If the system is a computer, the end points are also referred to as virtual interfaces.

## Network traffic inspection – Ethernet

### *Pcap – packet capturing API*

libpcap is an API for low-level packet capturing. There is a version for Windows, winPcap. Packets are read directly from the network adaptors. The network adaptor must be used in promiscuous mode, meaning that it will not use any filtering. Otherwise, packets for other destinations would not be visible.

The following links are the start points:

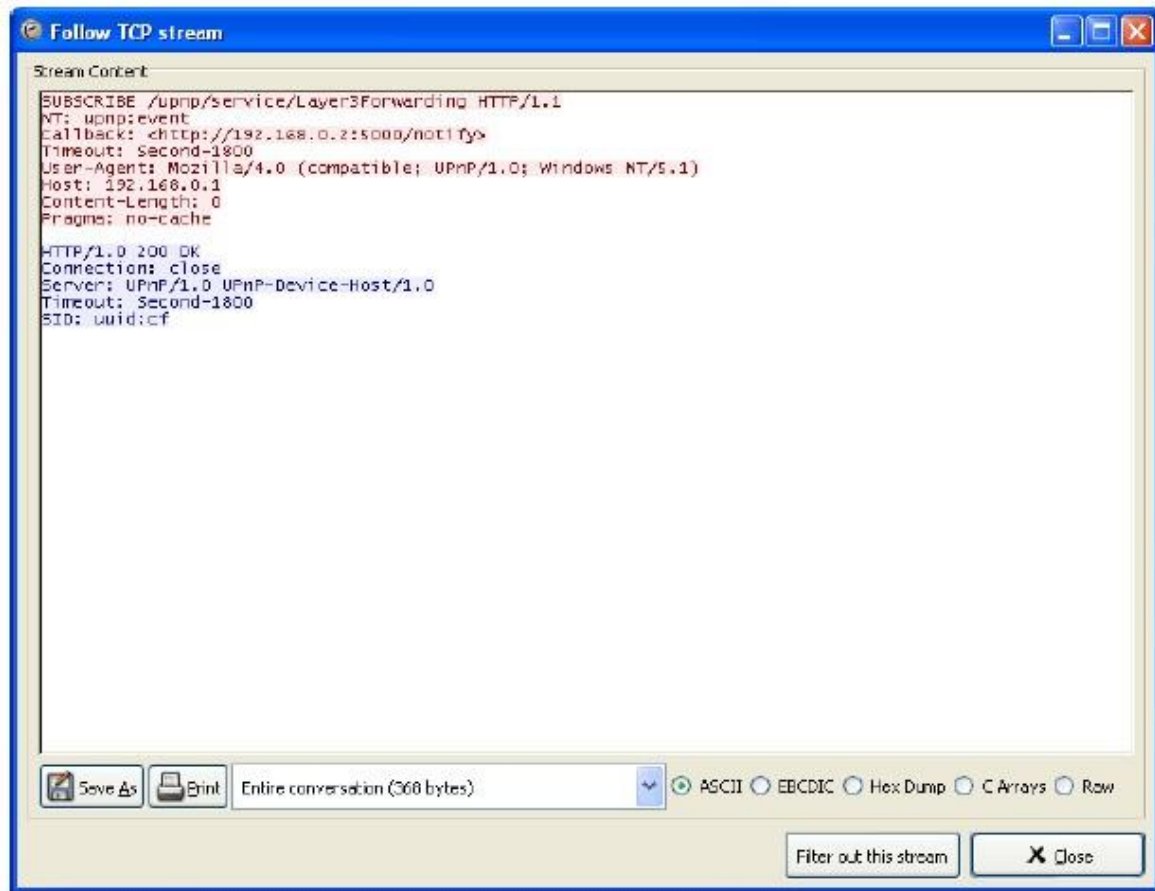
<http://www.tcpdump.org/>  
<http://www.winpcap.org/>

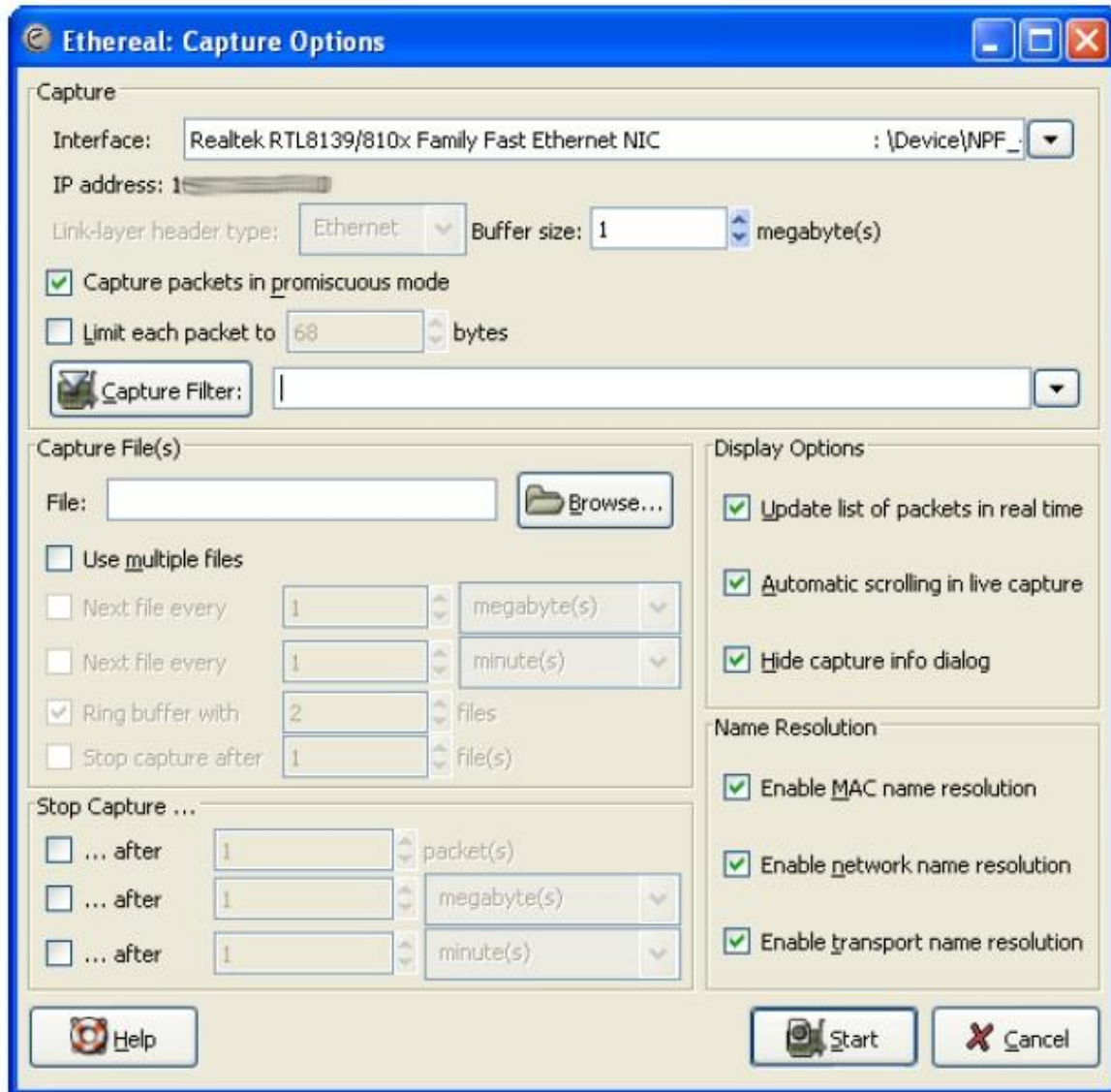
### *ethereal*

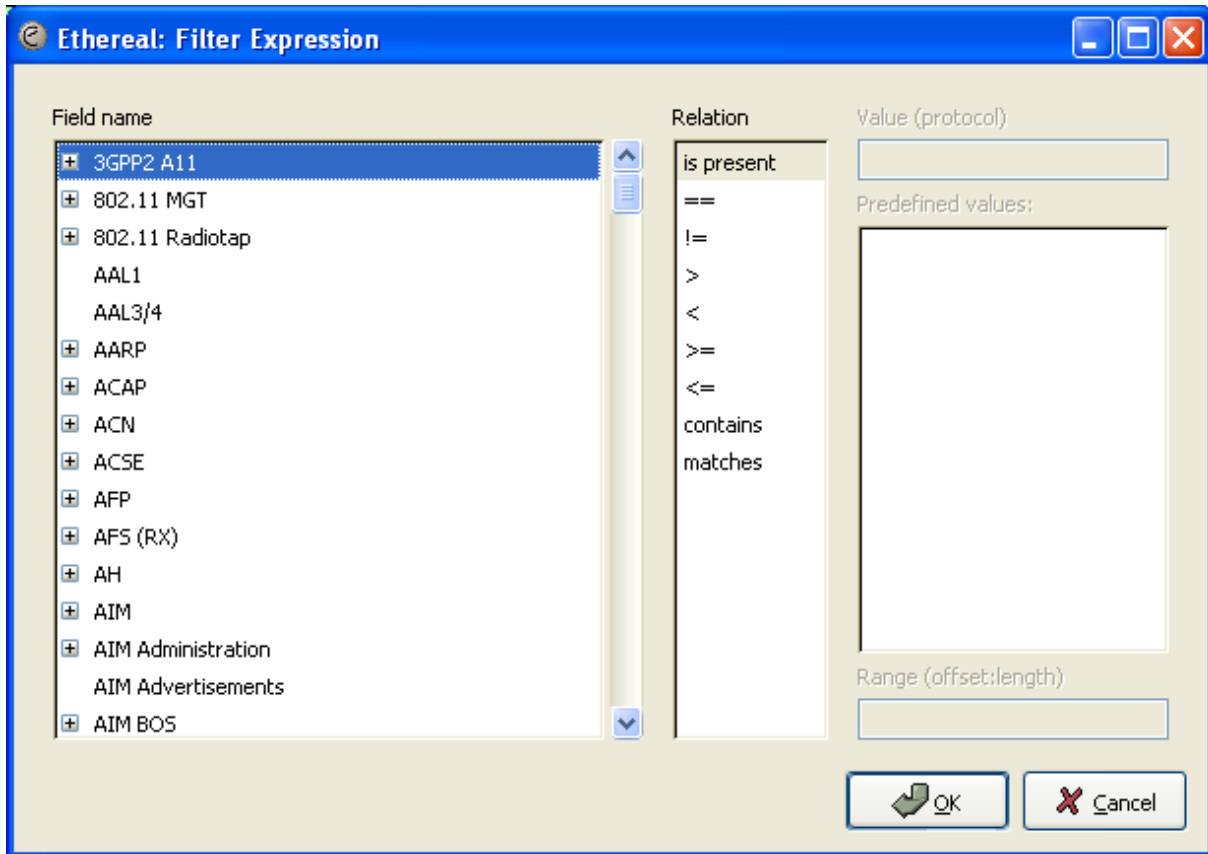
Ethereal is a protocol analyzer: it can capture real-time packets, or read historical traces, also many produced by other tools.



Figure 7.1. The "Follow TCP Stream" dialog box







## Layer 3: IP

The IP protocol forms the basic layer for TCP/IP networking. It is the carrier for the 'Internet'. It is independent of the physical networking. It is available on practically any system today.

The IP version that will be used in this course is IPv4. IPv6 has been defined, but it is still gaining momentum only slowly.

### *IP essentials*

An IP packet contains two essential elements: the addresses of the sending component and the receiving component, and the data to be transmitted:

(from address, to address, data)

IPv4 addresses are 32 bits in size. This was initially expected to be large enough. Then, with the Internet boom it was considered to be too short to last very long. IPv6 defines 64bit addresses. Meanwhile, there are a number of technologies that have relieved the pressure on the address space significantly. We will discuss this later.

Next to the essential data, there is additional information in the packets. The more important parts are: IP version (4 or 6, as said, we only talk about version 4), field lengths, checksum (integrity control), flags, fragmentation information, time-to-live (or hop count), and protocol (with ICMP, UDP, TCP the main protocol types)

### *IP addressing*

An IP address is a 32 bit number. It is written typically as 4 decimal numbers, each representing one byte of the address. This representation is the so-called "dotted decimal" format: d1.d2.d3.d4.

Example: 206.4.56.11, representing 0xCE,0x04,0x38,0x0B.

Each component that participates in an IP communication has a unique address. When all nodes that are connected are on the same LAN, that requirement is sufficient. For point to point connections, addresses are not necessary. Such a connection can be established with SLIP or PPP, for instance, and there address information can be suppressed.

In an isolated network any IP numbers can be used, as long as they are unique. In communication between multiple networks, there must be more structure in the addressing to allow communication. Instead of just node numbers, also network addresses (identifiers) must be used. On the Internet, with its millions of nodes, there must be rules to assign network and node addresses..

### *Multiple LANs*

The operation within each LAN is similar to the operation within a single LAN. The only requirement is that component addresses must be unique. When there is more than one sub-network, the requirements change. The component address is split in two parts: a sub-network identifier, and a component identifier. The address becomes a pair (LAN, component). In IP the 32 bits are split in two parts: the leftmost bits form the network identifier, the remaining bits are the component identifier.

On each LAN there are two special component addresses: the address with all bits '0', and the one with all bits '1'. The all '1' is the broadcast address: the destination are all the systems on

the LAN. Therefore, the number of actual real nodes possible is  $2^n - 2$ , where n is the number of bits for the node address.

Previously, the number of bits for the network was 8, 16 or 24. Nowadays, the number of bits can be any number. With 8, 16 and 24, the sub-networks that could be formed were divided into classes: very large, large, small. The new scheme is called CIDR: Classless Inter-Domain Routing.

All components in a sub-network have the same network identifier. The netmask is used to identify the sub-network identifier given an IP address: the 'and' of the IP address and the netmask gives the network identifier.

There are two ways to look at the network part of an address: it is 20 bits, or the netmask is 0xffff000. Bit naming using the "n.n.n.n/20" notation, whereas netmasks are expressed as n.n.n.n, with zero values for the LAN part, like 255.255.240.0.

Unfortunately, the dotted decimal notation does not make it easy to do this for most people. For a 20-bit network, the mask could be 00001010.00000101.0101xxxx.xxxxxxxx or 0x0a055000/20 or 10.5.80.0/20.

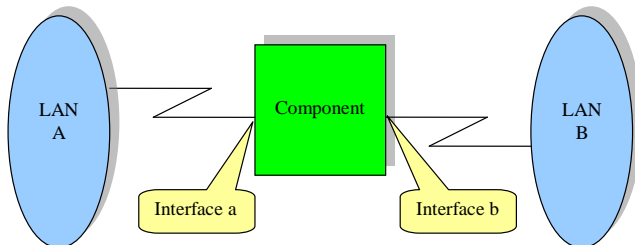
Suppose the IP address is 10.5.89.33, and the netmask is 255.255.240.0. What is the LAN?

```

10=00001010
5=00000101
89=01011001
33=00100001
255=11111111
240=11110000
00001010 00000101 01101001 00100001
11111111 11111111 11110000 00000000
-----
00001010 00000101 01100000 00000000
Or: 0x0a056000
Or: 10.5.80.0
    
```

### Multiple interfaces

If a system wants to communicate with another component, it will determine if the component is on the same LAN. It is on the same LAN if the network address of the component is the same as a network address of the sending component on one of its interfaces.



On each interface, the system has an IP address. On each interface there is a LAN, defined by the network address. Alternatively the IP address on that interface together with the netmask define the LAN address.

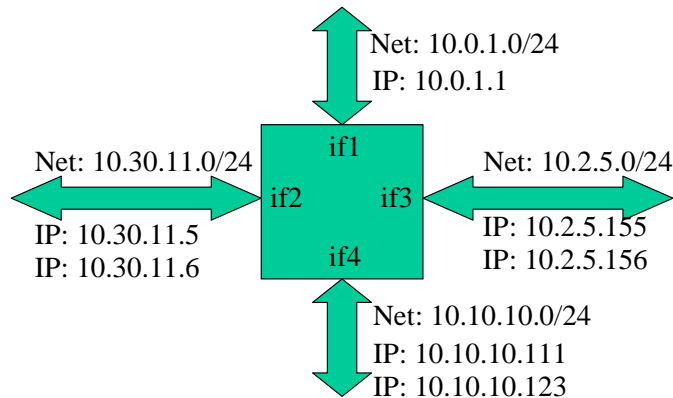
The view of the component on the addresses should be consistent: there should be no overlaps.

**Note**

Sometimes, you may find an interface without an IP address. As far as IP is concerned, it does not exist. It is used to listen to communication without being able to send any IP traffic. This is a stealth listening device.

**Node – interface – LAN – IP address**

A node can have multiple interfaces: if1, if2, if3, if4. Each interface is connected to an IP LAN, defined by its network mask. The component can have multiple IP addresses on one interface, all part of the same LAN.

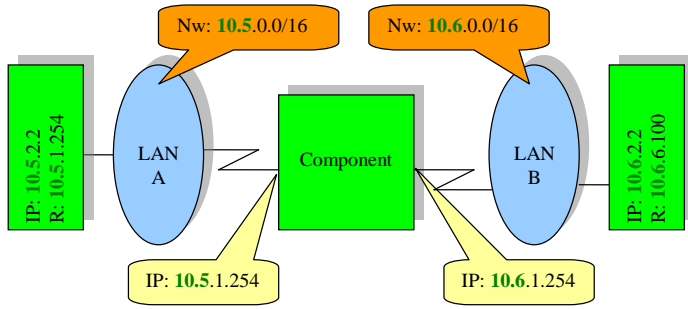


It is possible that you have virtual interfaces. A real interface is duplicated into multiple logical interfaces. This can make higher level communication configuration look normal, whereas under the hood a change has been made, like for trunking over one connection.

**Inter-LAN communication**

To allow communication between two components in different LANs there must be at least one component that is present on both LANs, and that will forward packets across. Components with just this purpose: forwarding packets to the right interface are called routers. A router has an IP address on all the LANs it is connected to. It knows where to forward packets to.

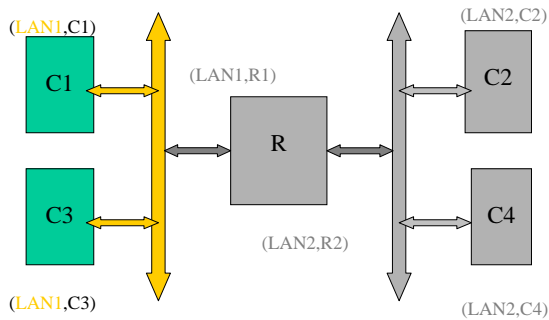
Each component can check if the destination is on the same LAN as itself. If there are multiple interfaces on the component, the check must include all networks as defined for all of the interfaces. These networks can have different sizes (number of bits used for the network). If the destination address is not part of any of the directly connected networks, the packet must be forwarded to a node 'closer' to the destination. Such a component is typically a router, a device that sits in between networks. The most simple configuration scheme is for every component to provide the address of that router. This is the same as saying that they all define a default route.



## Communication examples

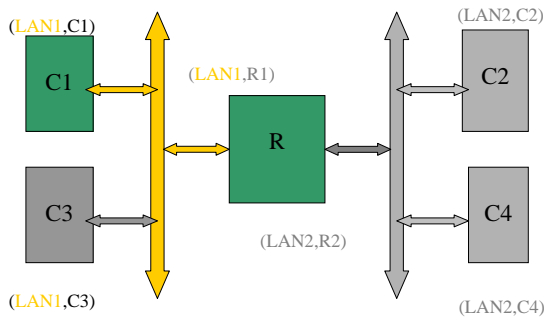
### Communication on the same LAN

In the first example we look at the communication on the same LAN: C1 takes two C3. Based on the IP address C1 knows C3 is on the same LAN, so it will try to communicate directly.



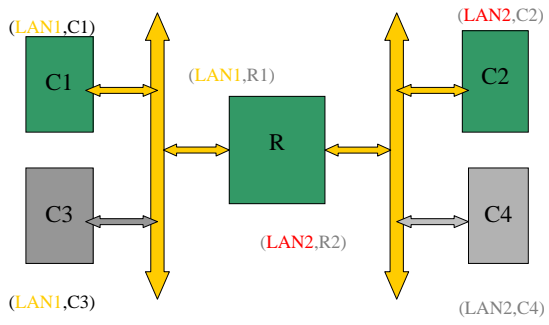
### Communication with the router

The router has an IP address on the LAN. Communication with the router happens directly.

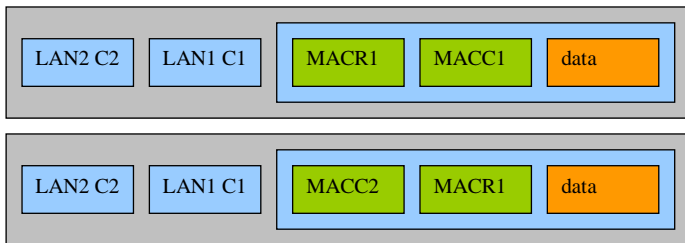


### Communication across the router

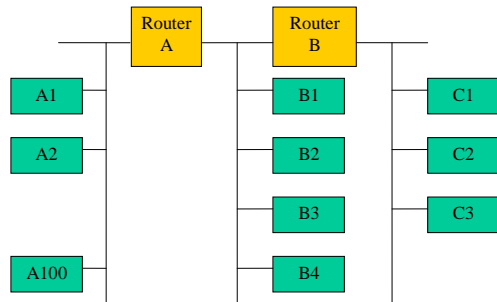
C1 wants to communicate with C2. Based on the IP and netmask, C1 knows C2 is not "local". C1 has R1 as default route. C1 sends the packet for C2 to R1. R1 receives the packet, and finds out C2 is local on LAN2. So it will set up a local communication on LAN2 with C2.



Packets:



## Exercise



For each LAN:

- What is the netmask?
- What is the network address?

For each component:

- What is the MAC address on each interface?
- What is the IP address on each interface?
- What is the default route?

For each router:

- What is the default route?
- What are the explicit routes?

Approach:

- identify the number of systems that would be present in the LANs:
- based on the number of components, determine the number of bits for components
- we propose to use 10.x.y.z style addresses (see RFC1918 and later)
- for LAN A: default route to router A; for LAN C default route to router B
- in LAN B, what is the default route?

## IP – ICMP - UDP – TCP

In this section we will discuss the basic protocols on top of IP. These protocols are ICMP, UDP and TCP.

- ICMP: Internet Control Message Protocol
- UDP: User Datagram Protocol
- TCP: Transmission Control Protocol

### ***Internet Control Message Protocol: ICMP***

The name ICMP indicates its purpose: Internet Control Messages. These messages are not intended to ship application data around, but are aimed at supporting the Internet infrastructure itself. They can be regarded as meta data. Rather than providing space for error conditions or diagnostics inside the core protocol, ICMP allows to add, extend and modify these facilities independently.

ICMP messages are addressed to the IP implementation itself, not to any application. This is worth noting, as the IP stack is the responsibility, most often, of the system, the OS. There is no application, no user interface, no user that knows or needs to about this, if all is well.

An ICMP packet contains two essential pieces of information, the ICMP type, and the ICMP code. The ICMP type is for example:

- ✓ destination unreachable
- ✓ time exceeded
- ✓ echo request and reply
- ✓ router advertisement and solicitation
- ✓ ...

The design of ICMP produces very simple communication packets, self sufficient. There is no state or connection set-up. The most complex communication is of the form request-reply, with two asynchronous messages, no delivery guarantee.

The messages that are most common are destination unreachable, time exceeded, and echo request-reply. Destination unreachable will be send back if a component has no way to deliver a packet to the destination address. In other words: when a routing component has no route to the destination, or knows there is no such destination.

Somewhat linked is the ICMP message: time exceeded. Packets contain a hop count, that is decreased in each hop. If it reaches zero, the packet is too tired to continue. The component notifies the sender it drops the packet and sends a time exceeded message back.

A specific use of this facility is the program traceroute/tracert. It sends packets with increasing time-to-live, and listens to time-exceeded messages. The first reply should come from the next hop, then the second hop, etc, effectively constructing the path a packet follows.

Echo request and reply are two sets of messages used for diagnostic purposes. The program 'ping' sends out request packages and listens for reply packages. One can measure round trip time, and indirectly detect destination unreachable.

ICMP messages are very basic, and that is exactly why they are so useful: they do not depend on complex state or availability of many services. If a "ping" works, you have a basic

assurance about what works in your system (IP stack, network connector, routing). If not you are unlucky, because any of the underlying systems may be malfunctioning, or the ICMP packets get blocked. That is way variations on ping using other protocols and ports may be handy as well.

## **TCP and UDP**

With IP, components send packets to components. But any component has typically many services running that want to communicate. To allow packets to be delivered to specific components another level of addressing is needed. Therefore, UDP and TCP introduce service identification, called port numbers. A service can listen to a specific port and IP address. Only packets for that port will be delivered to the service. If a service registers it is using a port, not other application should be allowed to use that port. This is specifically true for ports used by well-known services like email: if those ports were used by unauthorized processes, they could read email intended for other people, for instance.

Both TCP and UDP provide an additional addressing level:

- ✓ component address space: service identification
- ✓ machine address space: LAN + node identification

A TCP or UDP connection becomes a 6-tuple:

(FromLAN, FromComponent, FromPort, ToLAN, ToComponent, ToPort)

## **User Datagram Protocol: UDP**

UDP does not introduce a lot of new things on top of IP. It just sends a packet from one site to the other, there is no guaranteed delivery, only best effort, there is no connection set-up, etc. The packet contain the following essential information on top of the IP information: source port, destination port.

UDP is popular for services that need a no-frills connection: basic packet delivery. It may be because reliability is less important than communication speed, or that reliability is already part of the application functionality.

Just like with IP, addresses and port numbers are easy to fake: any rogue system can craft packets that seem to come from elsewhere.

Example services using UDP are DHCP, DNS, NETBIOS, and SNMP.

## **Transmission Control Protocol: TCP**

TCP introduces connection set-up between services on remote components and reliability.

A TCP packet contains the following essentials:

(source port, destination port, sequence number, acknowledge number)

It also contains flags, checksums for integrity, and field lengths.

The flags play an important role in session set-up and tear-down. The session is set-up using a handshake along these lines: SYN /id=a – SYN/ACK/(id=b,id=a) – ACK/(id=a+1,id=b). SYN and ACK are flags, the id is the sequence number. The sequence number allows for packets to arrive out-of-order: they can be sorted again. Packets can arrive out of order within a certain window (number of packets with some missing), and are confirmed after a number of packets, either by a data carrying packet, or a packet only serving this purpose.

Packets need not be acknowledged individually, one can acknowledge a set of messages, again using the sequence number of the last OK package received.

TCP provides a simple form of connection authentication. The basic authentication is based on sequence numbers. Both side pick a sequence number, and check if the numbers are correct. If the numbers are unpredictable, a man-in-the-middle attack can only succeed in a reasonable time for people that are on the path of the packets. However, if you are on the path, the protection is voided: you see both numbers and can take over the connection (hijack it).

This is only a very short introduction to TCP. It is a simple protocol on the one hand, but it gets complex enough when looking at the details, like speed optimizations, packet size optimizations, and retransmission behavior.

## ***Fragmentation***

If packets become too large for sending it over to the next hop, they may be split in multiple smaller packets: fragments. Such splitting may occur as a normal side effect of tunneling, or specific technologies. It is therefore not necessarily a problem or a hacking attempt or such. That said, fragmentation does introduce a feature that has been used to conceal attacks.

The basic idea is simple, and has been reused against other systems. If the defender uses packet based inspection to detect attack signatures, fragmentation allows you to split the data of the packet in such a way that the signature on each packet will fail. Given sufficient fragmentation, at some point the defender will create too many false positives if he attempts to deal with this attack on a per-packet basis.

The obvious approach to prevent this filter evasion from working, is to recompose packets before inspection. This changes the complexity of the filtering solution, and may be used to attempt a DoS against the filtering tool.

There is a second problem with fragmentation. Fragment contain a start and a length, so it is perfectly possible to create overlapping fragments. If the filter and the real target handle this situation differently, that is a problem. If the filter allocates resources for the full packet on receiving the second fragment, artificially sending only “second fragment, high offset” messages may overrun available memory.

## ***Windows***

TCP/IP uses acknowledgement on multiple packets to speed up communication. These ACKs are required because TCP/IP also supports retransmission to overcome temporary failures. Here too, an attacker might just try killing acks (remove ack flags) to cause retransmissions, send packets out of order, and forgetting the initial ones, and other ideas to abuse this feature.

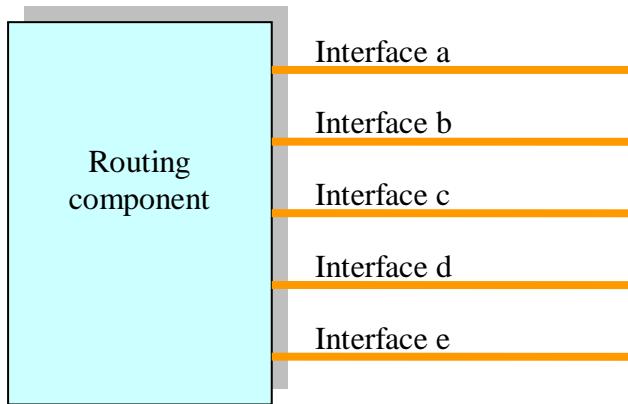
## ***Duplicate packets***

The normal behavior is to drop duplicate packets, or invalid ones, with no further thoughts. If an attacker can beat you with one packet that is ok in the stream, your session is hijacked.

## ***Routing***

In this part we give a very short introduction to IP routing. Note that we say IP routing, not TCP nor UDP routing. Packets are routed independently, and packets from a single TCP session may follow different paths.

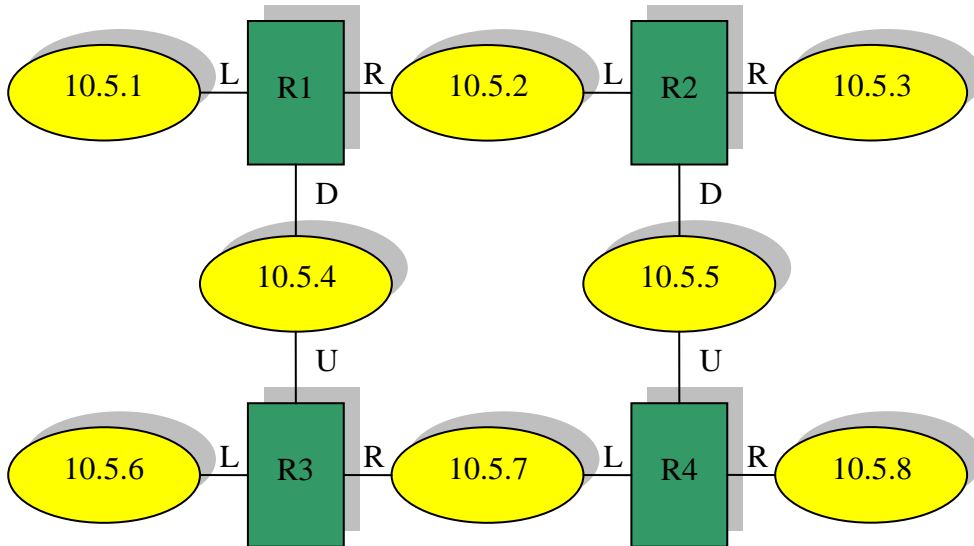
The problem is easy to state: the Internet consist of many LANs. Some nodes are part of more than one LAN (otherwise they could not communicate). These nodes are willing to forward packets from one LAN onto another LAN.



But which packets should go where? The problem is often compared to the traffic problem: what way to take at a crossing?

**Static (Configured) routing**

One way that works with small networks is similar to the physical world: place road signs. In IP this is saying to configure routes on each device manually. In small and critical networks this may be the way to go, but in general this is neither manageable nor flexible enough.



R1	10.5.1	L	R3	10.5.1	U
	10.5.2	R		10.5.2	U
	10.5.3	R		10.5.3	U (R)
	10.5.4	D		10.5.4	U
	10.5.5	R		10.5.5	R
	10.5.6	D		10.5.6	L
	10.5.7	D		10.5.7	R

	10.5.8	D (R)		10.5.8	R
R2	10.5.1	L	R4	10.5.1	U (L)
	10.5.2	L		10.5.2	U
	10.5.3	R		10.5.3	U
	10.5.4	L		10.5.4	L
	10.5.5	D		10.5.5	U
	10.5.6	D (L)		10.5.6	L
	10.5.7	D		10.5.7	L
	10.5.8	D		10.5.8	R

### Routing protocols

The first idea to compute routes automatically that comes to mind is that for every LAN you know which way to send the packet, based on the shortest distance to that LAN. This is called Distance Vector Routing.

Every node in the connection graph computes the shortest distance to every other LAN. The way to compute these distances is a classic graph algorithm: you ask your direct neighbors for their distances, compute your point of view, and you also distribute your point of view to the neighbors.

Distance Vector Routing is used most often internally, where the number of LANs is limited, and there is less risk of rogue routers. The typical protocol is RIP.

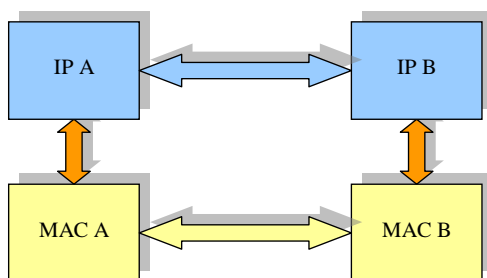
The second type of routing is Link State routing. IS-IS and OSPF are two of the better known link state routing algorithms.

### IP – Ethernet

The network layering should make layers independent. Yet, some interaction, and thus dependencies, are needed. The addresses of the IP network are different from the addresses on the Ethernet layer. Somehow, the IP addresses have to be translated into MAC addresses.

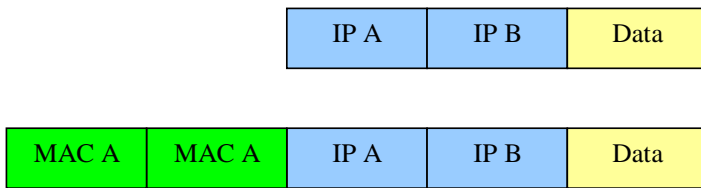
On a single LAN, the operation is rather simple. To send a packet to IP<sub>B</sub>, the Ethernet layer needs to know which MAC the component with IP<sub>B</sub> has. To preserve the independence of the layers, the Ethernet layer can find out the MAC of IP<sub>B</sub> all by itself. It broadcasts a request on the LAN asking for the MAC of IP<sub>B</sub>.

The Ethernet layer does not have to know the above layer is IP. It can treat the IP address as opaque data: what is the MAC for the logical address in this field?



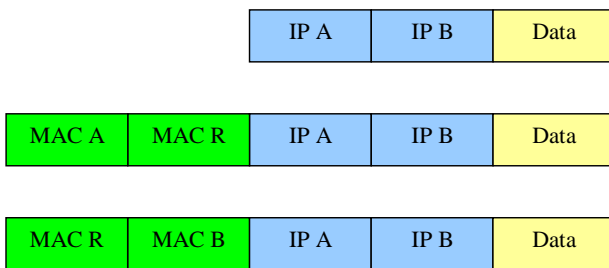
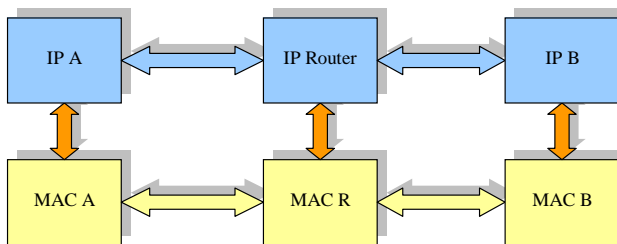
From this simple situation one may derive wrong conclusions. Let us look at the data packets. The application data is part of a packet where an IP header is added: the IP address “from”

and the IP address “to”. At the Ethernet level, again a header is added: MAC “from” and MAC “to”. (More data is present in the headers, and the order is not necessarily the one indicated).



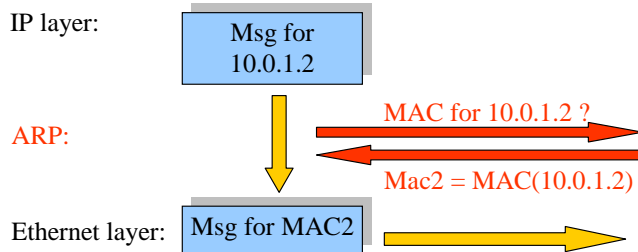
So it would seem that the MAC “to” is always the MAC of the IP “to”, and that the MAC “from” is the MAC of the IP “from”. This is not true for more complex set-ups where there are multiple LANs. The situation for more complex LANs is, well, more complex.

The IP level knows about IP subnets. If the destination IP is local, the request to deliver the packet contains the destination IP as target. If not, the destination address given to the Ethernet level is the address of the correct router component. The MAC “to” will then be the MAC of the router that is the IP target (next hop) for the packet.



## Address Resolution Protocol: ARP protocol

Each component keeps an ARP cache. This is a table with (IP,MAC) association information. All IPs in the cache should be systems on the same IP LAN. If an entry is not in the cache, via a layer 2 broadcast the system asks: which is the MAC for this (IP) address?



When the system with IP B answers, all other systems can also update their caches. The ARP cache can have static entries. Entries expire (aging or LRU, for instance). The expiration allows the system to cope with configuration changes.

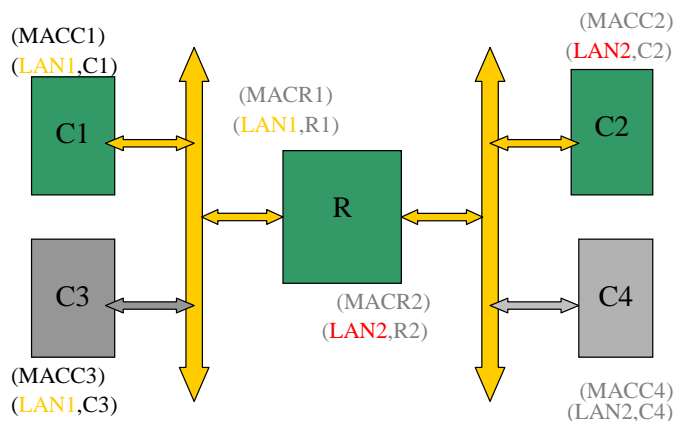
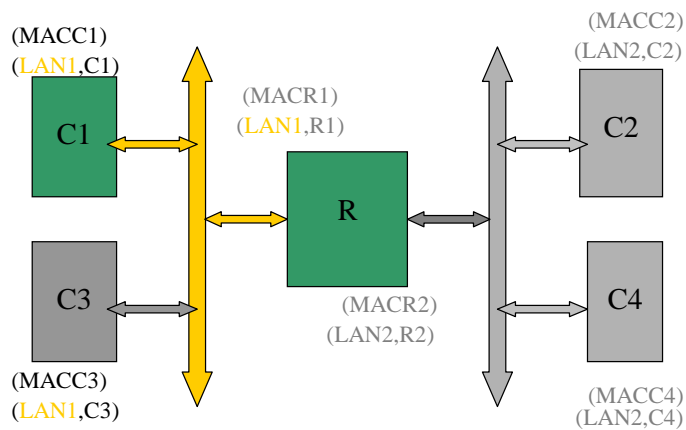
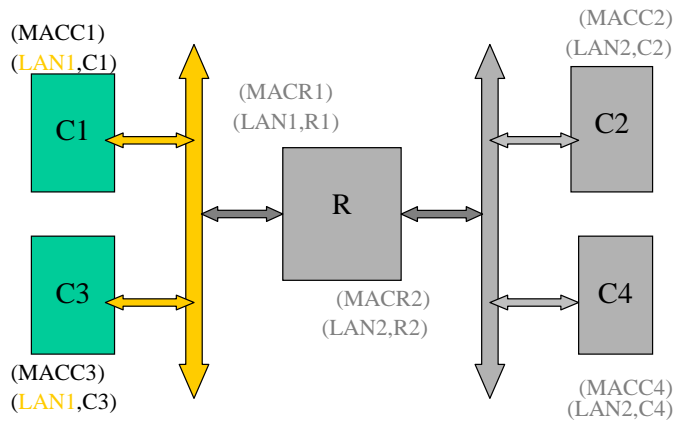
ARP: RFC 826 (STD 37)

RARP: RFC 903 (STD 38)

ARP answers the question: which is the MAC for this (IP) address?

RARP answers the question: which is the (IP) address for this MAC?

## Network communication: IP and Ethernet interactions



## Network Address Translation

Network address translation is the introduction of a mechanism to translate IP addresses in new addresses in a consistent way so that the other component does not notice that the addresses are changed.

A few reasons to use NAT:

- ✓ security: to hide internal addresses
- ✓ cost: it requires less official addresses if all internal addresses are translated at the organization's border
- ✓ transparency for internal components
- ✓ internal use of "special" addresses (RFC 1918)
  - ✓ 10.0.0.0 - 10.255.255.255
  - ✓ 172.16.0.0 - 172.31.255.255
  - ✓ 192.168.0.0 - 192.168.255.255

NAT breaks the original connection in two new connections:

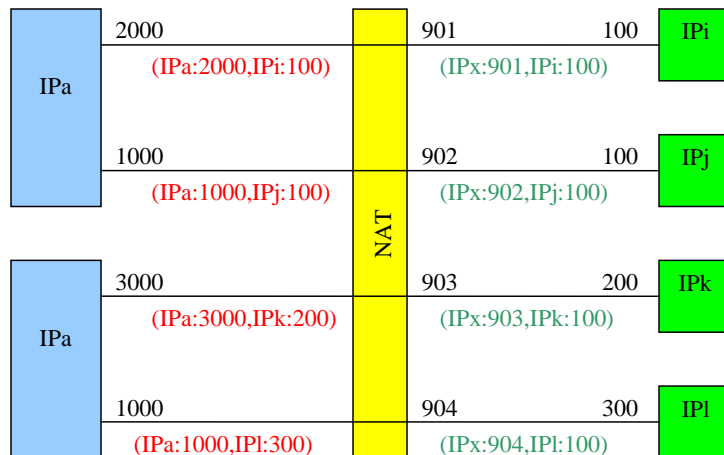
- ✓ "from" - NAT component
- ✓ NAT component - "to"

NAT modifies the packets: it changes either

- ✓ source IP/port into NAT-IP/NAT-port
- ✓ destination IP/port into NAT-IP/NAT-port

### ***NAT intern to extern***

The internal system sets up a connection to an external system. To hide the internal address, the "from" address must be changed. The NAT component has a set of addresses it can use to establish the external connections. One IP address can only cope with a limited, even if rather high, number of connections, because the port numbers must all be different. The NAT component must also track sessions to know when an IP/port combination could be reused. For outgoing UDP connections there will be a timeout to stop the reverse translation. Protocol knowledge is required to know which packet triggers which response. The NAT component has complete freedom in its choice of IP and port number (assuming it owns the IP and stays clear of low ports to be a nice Netizen).



The NAT component cannot just replace the source IP. Also the port must be changed. Otherwise, there would be collisions if two components with a different IP but same source port would be translated.

### ***NAT: extern to intern***

The NAT components receives a packet for IPx port 80. How should it be translated? In this case, there is no automatic translation possible. The goal is to provide virtual addresses for

internal servers. Which virtual address corresponds to which internal address must be configured.

## Automatic Configuration

### ***Automatic Configuration: What is the problem?***

IP configuration is not that simple. It requires correct setting for many parameters. You need to know the correct values for the parameters, as there are no obvious default settings that can be used. The parameters need to be specific for the given infrastructure. It is unlikely that the settings in one case will work unmodified elsewhere, even in the same organization.

A particular problem case are mobile solutions, like portables. Portables are hooked up to more than one network, in more than one location. They need re-configuration per connection point. Given that many of the users of portables could be described as “Technically challenged” an automated solution is very desirable.

With the advent of wireless networks the cry for automatic configuration is increased. Note that automatic configuration and connection to network may pose other concerns, in the realm of security.

### ***Parameters***

We give a short overview of the parameters that might be configured automatically, but that need to be configured for a proper operation of the network.

#### **Link-layer**

Per interface:

- Trailers (on/off)
- ARP cache timeout (integer)
- Ethernet encapsulation (RFC 894/RFC 1042)

#### **IP-layer**

Per host

- Be a router (on/off)
- Maximum reassembly size (integer)
- Default TTL (integer)

Per interface

- IP address (address)
- Subnet mask (address mask)
- MTU (integer)
- All-subnets-MTU (on/off)
- Broadcast address flavor (0x00000000/0xffffffff)
- Perform mask discovery (on/off)
- Be a mask supplier (on/off)
- Perform router discovery (on/off)
- Router solicitation address (address)

#### **Routing**

List of Default routers

- router address (address)
- preference level (integer)

List of static routes

- destination (host/subnet/net)

- destination mask (address mask)
- type-of-service (integer)
- first-hop router (address)
- ignore redirects (on/off)
- PMTU (integer)
- perform PMTU discovery (on/off)

## TCP

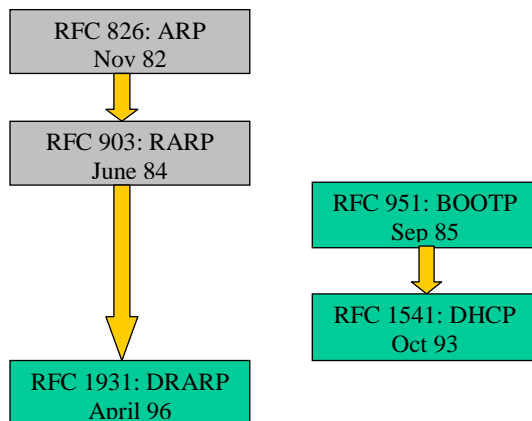
### Per host:

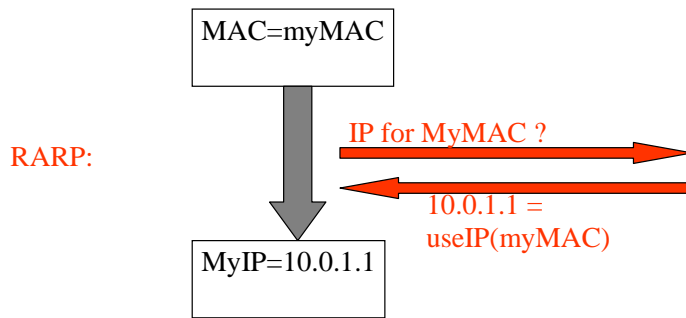
- TTL (integer)
- Keep-alive interval (integer)
- Keep-alive data size (0/1)

### **Auto Configuration: alternatives**

- ✓ Dynamic RARP (DRARP)
- ✓ Trivial File Transfer Protocol (TFTP)
- ✓ Internet Control Message Protocol (ICMP)
  - additional routers: "ICMP redirect"
  - subnet mask: "ICMP mask request"
  - locate routers through the ICMP router discovery mechanism
- ✓ BOOTP
- ✓ Dynamic Host Configuration Protocol (DHCP)

The following scheme shows the chronological relation between the RFCs related to automatic configuration.



**RARP and DRARP**

ARP and RARP do the opposite. RARP starts from a MAC address and asks for an appropriate IP for this MAC. It is limited to configuring the IP address. It is a layer 2 protocol, so it cannot be routed over the internet. Each LAN that offers RARP must have a RARP server.

**BOOTP**

- RFC 951
- uses IP (network layer), hence routable
- compare: (D)RARP: link layer
- obsolete: replaced by DHCP, RFC 1541

**DHCP**

RFC 1541: Dynamic Host Configuration Protocol

DHCP consists of two components: a protocol for delivering host-specific configuration parameters from a DHCP server to a host, and a mechanism for allocation of network addresses to hosts

DHCP is a client – server protocol. It follows the client-server model. It assumes that there exist designated DHCP servers. The DHCP servers are responsible for the allocation of network addresses, and to deliver configuration parameters to dynamically configured hosts DHCP uses UDP as its transport protocol.

Messages from a client to a DHCP server are sent to port 67, messages from the server to the clients use port 68.

**DHCP: IP to host mapping**

There are three main styles of IP address assignment: automatic, dynamic and manual.

Automatic assignment creates a permanent link between a component and the address it gets assigned. The typical use is for assigning IP addresses to desktops: they are located at a fixed place in the network.

Dynamic assignment hands out an IP address for a limited period of time (lease period). The host may explicitly relinquish the address. This type is used for instance with laptops. They move around.

With manual assignment an IP address is assigned by the network administrator directly: he defines which IP address a specific component gets. The major type of component for this style of assignment are servers.

## ***DHCP - BOOTP: relations***

BOOTP inheritance:

- ✓ DHCP can use BOOTP relay: RFC 951
- ✓ uses BOOTP message format defined in RFC 951
- ✓ DHCP must provide service to existing BOOTP clients

## ***DHCP - BOOTP: primary differences***

DHCP clients

- ✓ can be assigned a network address for a fixed lease
- ✓ serial reassignment of network addresses to different clients
- ✓ can acquire all of the IP configuration parameters

DHCP provides an explicit client identifier, which may be the hardware address (MAC<sup>o</sup>, or a DNS name.

## ***DHCP: the basics***

A DHCP client requests an address for some period of time (lease period). The server's allocation mechanism guarantees that:

- ✓ the address it hands out is not re-allocated within lease period
- ✓ it returns the same IP address to the same client

A DHCP client may:

- ✓ refresh the lease: a refresh every hour is common
- ✓ release the address: when ending working at one place and moving to another, it is best to release the address. This may be part of the shutdown procedure, but this is not always the case.
- ✓ ask for an infinite lease: that way, the dynamic system is used as a static one. Some people prefer this above manual set-up.

## ***DHCP: Client-Server Protocol***

### ***Message format***

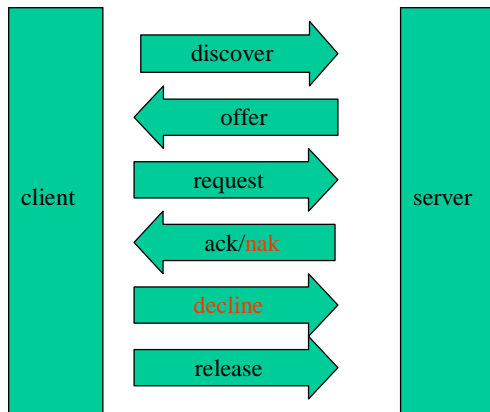
DHCP reuses the BOOTP message format defined in RFC 951.

There are two types of message formats:

- client->server: BOOTREQUEST format
- server->client: BOOTREPLY format

They contain the following fields:

- operation: 1 = REQUEST, 2 = REPLY
- xid: Transaction ID, a random number, used by the client and server
- ciaddr: Client IP; filled in by client when verifying previous parameters.
- yiaddr: 'your' (client) IP address
- giaddr: Relay agent IP address
- chaddr: Client hardware address



Messages: client to server

- DHCPDISCOVER: broadcast to locate available servers
- DHCPREQUEST: broadcast requesting offered parameters from one server (implicitly declining offers from all others)
- DHCPDECLINE: received configuration invalid
- DHCPRELEASE: relinquishing IP address (also: canceling remaining lease)

Messages: server to client

- DHCPOFFER: in response to DHCPDISCOVER with offer of configuration parameters.
- DHCPACK: configuration parameters, including committed network address.
- DHCPNAK: refusing request for configuration parameters (e.g., requested network address already allocated).

### **Chicken&Egg problem**

How can the server send an IP datagram to the client, if the client doesn't know its own IP address (yet)?

### **Chicken&Egg solution**

- client knows its own IP address:
  - normal IP: client will respond to ARPs
- client does not yet know its IP address
  - the client cannot respond to ARPs, two options:
    - 'manually' construct an ARP address cache entry
    - fill in an entry using the 'chaddr' and 'yiaddr' fields
    - send the bootreply to the client's IP address
    - send the bootreply to the IP broadcast address on the appropriate interface

### **Security of DHCP**

DHCP is built directly on UDP and IP. Both UDP, IP are inherently insecure.

DHCP is quite insecure:

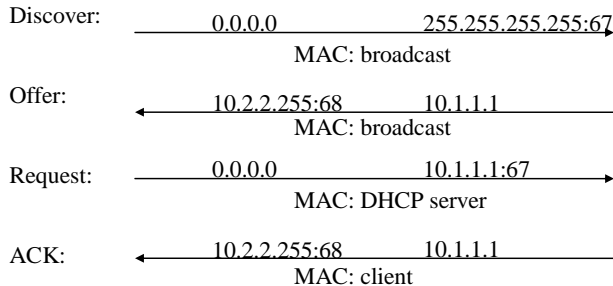
Pirate DHCP servers are easy to set up.

Malicious DHCP clients can:

- masquerade as legitimate clients
- retrieve information intended for legitimate clients.

## DHCP relay

To use DHCP as described poses a problem. The idea is to use IP as the layer to configure IP. In order to achieve this weird functionality, IP must be used in a peculiar way. The following scheme describes how it could work (we discuss the real solution next).



The client does not know neither its own IP address, nor the IP address of the DHCP server. Hence, it uses a source IP 0.0.0.0 (unassigned), and a destination IP 255.255.255.255: a broadcast to any. On the Ethernet level a broadcast is the only option.

For the offer, the DHCP server can reply with his own IP address as source, and the target IP address as destination. However, on the local LAN of the client, an ARP request to find the MAC for the client's newly assigned IP will fail: the client does not know his IP. Hence, an Ethernet broadcast address must be used. The client must be in promiscuous mode for IP: any packet for any IP address may be the offer from the server.

The client request can use either 0.0.0.0 (the address has not been officially assigned) or the address it requests (which might have been assigned to another station in the meantime). It does know the IP of the server now (from the offer). If the server is local, the ARP request will yield the MAC of the server. The client only knows if the server is local depending on the netmask, which is, again, a parameter to be determined via DHCP.

The server can acknowledge, using both sides IP address. The client's MAC is known, so if the DHCP server is on the LAN it can use Ethernet unicast. The client knows which IP the packet will be sent to (the IP of the offer), so it needs only listen for that IP.

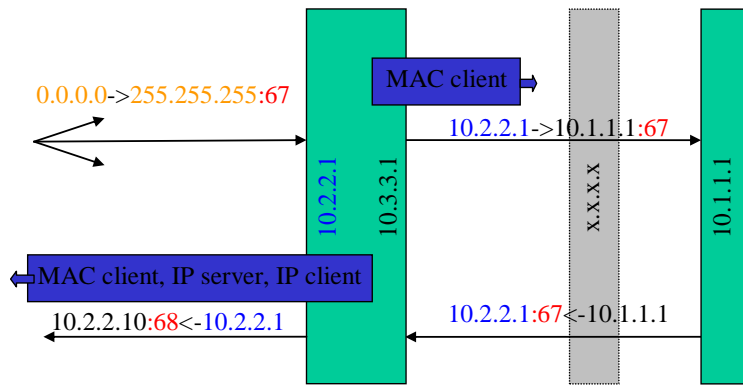
There are a few problems with this scheme. The most important one is that there are too many broadcasts, with ugly IP addresses: source 0.0.0.0, destination 255.255.255.255.

The solution for this problem is the introduction of DHCP relays.

### ***RFC 1542: Clarifications and Extensions for the Bootstrap Protocol***

W. Wimer, Carnegie Mellon University, October 1993

DHCP relays listen on the local LAN. They capture DHCP requests and turn them into normal UDP communication, including information on the source LAN. When they receive answers, they dispatch them locally.



### Fields in packages

- ✓ c i addr: client IP address
- ✓ y i addr: your IP address (assigned address)
- ✓ s i addr: server IP address
- ✓ g i addr: gateway IP address
- ✓ c h addr: client hardware address

The main function of the protocol is to apply the function:  
`yiaddr = dhcp(chaddr)`

To provide more detail, the functionality is:  
`yiaddr = dhcp(chaddr, siaddr [,ciaddr][,broadcast])`

### Use of fields

Server tells the client

- The client's IP address: yiaddr
- The server's IP address: siaddr

Client tells the server

- The IP address it wants/had last time: ciaddr
- The request to use broadcast reply: broadcast flag

### Gateway address usage in request

If giaddr == 0 (0.0.0.0)

Then

the relay agent MUST fill this field with the IP address of the interface on which the request was received.

Else

the 'giaddr' field MUST NOT be modified.

### The server reply handling

If giaddr != 0.0.0.0 Then

send the BOOTREPLY as an IP unicast to 'giaddr':67  
 that relay agent will then perform the final delivery to the client.

Else If broadcast flag on Then

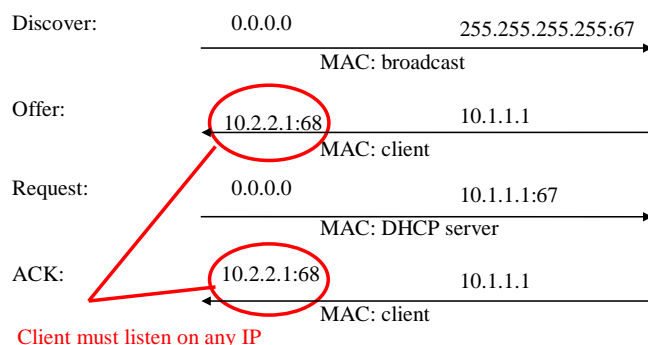
- use IP broadcast 255.255.255.255:68
- use link layer broadcast

Else

use IP unicast to yiaddr:68 (assigned/proposed IP address)

use link layer unicast to chaddr (client hardware address == MAC)

### The actual scheme:



## IP Multicast

### Motivation

There are many situations where multiple clients are interested in the same information. The mechanism that is used most often is called publish – subscribe. A publisher pushes information out, and subscribers can register to receive it.

Technically, there are multiple ways to support this style of operation.

The sender could maintain a subscription list and send the packet to each and everyone of the subscribers. This is how mailing lists operate, for instance.

On the other hand, packets could be broadcasted, so that multiple hosts could read the same message at the same time, leading to a much increased performance.

### Definition

IP Multicast is introduced in the RFC 1112: “Host Extensions for IP Multicasting”:

“IP multicasting is the transmission of an IP datagram to a "host group", a set of zero or more hosts identified by a single IP destination address. A multicast datagram is delivered to all members of its destination host group with the same "best-efforts" reliability as regular unicast IP datagrams,”

### Sending

First, on the IP level, a set of addresses is reserved for multicast use. Anyone can send a multicast to one of the addresses. The addresses are defined in two sets: sets which are globally managed by IANA, and addresses for occasional, specific local use.

Multicast addresses are those addresses in the range 224.0.0.0 to 239.255.255.255. The address 224.0.0.0 means “nobody”, and 224.0.0.1 means “all”.

A list of assignments can be found at: <http://www.iana.org/assignments/multicast-addresses>

Some examples:

```

224.0.0.0 Base Address (Reserved)
224.0.0.1 All Systems on this Subnet
224.0.0.2 All Routers on this Subnet
224.0.0.4 DVMRP Routers
224.0.0.5 OSPFIGP OSPFIGP All Routers
224.0.0.6 OSPFIGP OSPFIGP Designated Routers
224.0.0.9 RIP2 Routers
224.0.0.10 IGRP Routers

```

```
224.0.0.12 DHCP Server / Relay Agent
224.0.0.16 designated-sbm
224.0.0.17 all-sbms
224.0.0.18 VRRP
224.0.0.22 IGMP
```

## Receiving

Client can dynamically join or leave a group. It is a local decision. There is no server involved. Clients can subscribe to any number of groups.

A host group may be permanent or transient. A permanent group has a well-known, administratively assigned IP address. It is the address, not the membership of the group, that is permanent; at any time a permanent group may have any number of members, even zero. Those IP multicast addresses that are not reserved for permanent groups are available for dynamic assignment to transient groups which exist only as long as they have members.

## Ethernet link

Ethernet supports multicast. So there is no need to use Ethernet broadcast. To use IP multicast over Ethernet, the 23 low-order bits of the IP Multicast address are put in the low-order 23 bits of the Ethernet multicast address 1.0.94.0.0.0. (01-00-5E-00-00-00).

See <http://www.iana.org/assignments/ethernet-numbers> for other multicast addresses.

The STP for instance uses 01-80-C2-00-00-00.

## Routing

See RFC 3376: Internet Group Management Protocol, Version 3.

The Internet Group Management Protocol (IGMP) provides multicast routers with information on group membership. Multicast routers request membership information, and nodes respond with a membership report.

Multicast routers can use the Distance Vector Multicast Routing Protocol (DVMRP) protocol to route multicast messages.

See: <http://www.ietf.org/internet-drafts/draft-ietf-idmr-dvmrp-v3-11.txt>

## References

For all RFC references, consult the internet at the URL: <http://www.ietf.org/rfc/rfcxxx.txt>

RFC 917: Internet Subnets

RFC 919: Broadcasting Internet Datagrams.

RFC 791: Internet Protocol

RFC 792: Internet Control Message Protocol

RFC 793: Transmission Control Protocol

RFC 768: User Datagram Protocol

RFC 1533: DHCP Options and BOOTP Vendor Extensions

RFC 1122: Requirements for Internet Hosts -- Communication Layers

RFC 1123: Requirements for Internet Hosts -- Application and Support

RFC 783: The TFTP Protocol (Revision 2)

Interconnections, second edition

bridges, routers, switches, and Internetworking Protocols

Radia Perlman

ISBN 0201634481

TCP/IP Illustrated, Volume 1: The Protocols, Addison-Wesley, 1994, ISBN 0-201-63346-9.

TCP/IP Illustrated, Volume 2: The Implementation, Addison-Wesley, 1995, ISBN 0-201-63354-X.

TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX Domain Protocols, Addison-Wesley, 1996, ISBN 0-201-63495-3.