Networking-FAQ

Beginner's Guide to TCP/IP Networking and
Networking FAQ for
NetBSD-Amiga

15 May 1994

by Hubert Feyrer

1 Preface

This guide was written to describe how TCP/IP networking is set up on an Amiga running NetBSD, a flavour of Unix. Therefore, whenever I mention Unix or NetBSD, NetBSD-Amiga is meant, if not otherwise stated.

1.1 Some warm words

This summary is intended to help people with little knowledge about networks to participate in that game. The reader is assumed to know about basic system administration tasks: how to become root, edit files, change permissions, stop processes, etc. See [AeleenFrisch] for further information on this topic.

Besides that, you should know how to handle the utilities we're going to set up here, i. e. you should know how to use telnet, FTP, . . . I will not explain the basic features of those utilities, please refer to the appropriate man-pages or to the references listed instead.

This guide is devided into four chapters In the first one I'll try to introduce all the basic concepts which will be needed throughout the whole guide. The second chapter shows the most basic steps to set up networking in practice, while chapter three explains how to set up some common network services. The last chapter lists some frequently asked questions (FAQs) and answers to them.

I write this guide with the intention in mind to give the unweary some basic knowledge. If you really want to know what's it all about, read [CraigHunt]. This book does not only cover the basics, but goes on and explains all the concepts, services and how to set them up in detail. It's great, I love it! :-)

1.2 Disclaimer

I hereby refuse to take any responsibility for any mistakes and wrong information that is contained within this document.

Any reprint is allowed, as long as the origin of the information and the author's name (mine :-) is stated. I'd like to get one issue then.

Further distribution of this text is allowed as long as it remains unchanged.

I'd also like to be informed if this is to be included on any PD-disks or CD-ROM-distributions.

Furthermore, I'd be glad to hear any comments (good & bad) about this guide. Please (e)mail me!

1.3 About the author

SMail: Hubert Feyrer
Bachstr. 40
84066 Mallersdorf

Phone: +49 (0)941 / 943-2415 (Work, best chances on weekdays)
+49 (0)941 / 701788 (Home on weekday's, rarely)
+49 (0)8772 / 6084 (Weekends)

EMail: `hubert.feyrer@rz.uni-regensburg.de`

2 Theory

In this first section, I'd like to give you some technical background and explain all the concepts, before actually starting to set up anything.

2.1 Protocols supported by NetBSD

There are several protocol suites supported by NetBSD. The first one implemented was DARPA's Transmission Control Protocoll/Internet Protocoll (TCP/IP). Shortly afterwards, the Xerox Network System (XNS) was added. The last protocol suites — parts of them still being implemented — are the ISO protocol suite, CCITT X.25 and ARGO TP.

Today, TCP/IP is the most widespread of the above. It's implemented on almost any hardware and operating system and it is also the most-used protocol in heterogenous environments. So, if you just want to connect your Amiga running NetBSD to some other machine at home, or you want to integrate it into your company's or university's network, TCP/IP is most probably the right choice.

The Xerox Network System was only implemented at UCB to connect isolated machines to the net, and the ISO-protocols are by far not as widespread as TCP/IP, also their popularity grows.

However, there are no such things as DECnet and Novell's NetWare for NetBSD-Amiga, though! These two protocols differ from the protocols mentioned above in that they are proprietary, in contrast to the others, which are well-defined in several RFCs and other open standards.

2.1.1 IP over serial lines

TCP/IP can be used on a wide range of media, NetBSD-Amiga supports Ethernet and serial lines. There are three reasons for using a serial line protocol in preference to ethernet:

- It's almost impossible to get a A2065 or Ameristar board for an Amiga.
- Your remote host is only reachable via telephone, so you have to use your modem to access it.
- Every Amiga has a serial port (which you don't have to pay for) and the cable you need is also cheaper than the one you need for Ethernet.

The disadvantage of a serial connection is, that you can connect only to one host, as no multi-serial boards (e.g. A2232) are supported (yet). Besides that, NetBSD can use at most 57,6kBd making it a lot slower than Ethernet's 10MBd.

There are two possible protocols to connect a host running NetBSD-Amiga to another host using a serial line (possibly over a phone-line):

- Serial Line IP (SLIP)
- Point to Point Protocol (PPP)

The choice here is mainly influenced by the host you want to connect to.

If you want to connect to another Amiga, possibly running under AmigaOS, use SLIP, as this is supported by AS225, AmiTCP and NetBSD. Also, if you want to connect to a Linux box or (using your modem) to a terminal server or another unix box which offers SLIP, the choice is clear.

Otherwise, if you want to connect to a host which only offers PPP, you can use that one.

From my personal experiences, if I had the choice I'd prefer SLIP instead of PPP, as it's more difficult to get a connection with the latter one. With PPP, both sides do an initial handshake which is **very** easy to timeout if you do the startup by hand; with SLIP, there's no such initial handshake, i.e. you start up one side, and when the othersite has its first packet, it will send it over the line.

RFCs 1331 and 1332 describe PPP and TCP/IP over PPP. SLIP is defined in RFC 1055.

2.2 TCP/IP address format

TCP/IP uses 4-byte (32-bit) addresses, also called IP-numbers (Internet-Protocol numbers).

Those IP-numbers are worldwide unique. To assure this, they are administrated by one central organisation, the DDN Network Information Center. They give certain ranges of addresses (network-addresses) directly to sites which want to participate in the internet or to internet-providers, which give the addresses to their customers.

If your university or company is connected to the Internet, it has (at least) one such network-address for it's own use.

If you just want to run your private network at home, see below on how to "build" your own IP-numbers. However, if you want to connect your machine to the (real :-) internet, you should get an IP-number from your local network-administrator or -provider.

When writing down IP-numbers, this is done in "dotted quad"-notation most of the time, i. e. the four bytes are written down in decimal (MSB first), separated by dots. For example, 132.199.15.99 would be a valid address. Another way to write down IP-addresses would be as one 32-bit hex-word, e.g. 0x84c70f63. This is not as convenient as the dotted-quad, but quite useful at times, too. (See below!)

Being assigned a network means nothing else but setting some of the above-mentioned 32 address-bits to certain values. These bits that are used for identifying the network are called network-bits. The remaining bits can be used to address hosts on that network, therefore they are called host-bits.

In the above example, the network is 132.199.0.0 (host-bits are set to 0 in network-addresses), the host's address is 15.99 on that network.

How do you know that the host's address is 16 bit wide? Well, there are four classes of networks. Each one starts with a certain bit-pattern identifying it. Here are the four classes:

- Class A starts with "0" as most significant bit. The next seven bits of a class A address identify the network, the remaining 24 bit can be used to address hosts. So, within one class A network there can be 2^24 hosts. It's not very likely that you (or your university, or company, or whatever) will get a whole class A address.
- Class B starts with "10" as most significant bits. The next 14 bits are used for the networks address, the remaining 16 bits can be used to address more than 65000 hosts. Class B addresses are usually given to universities.

  Returning to our above example, you can see that 132.199.15.99 (or 0x84c70f63, which is more appropriate here!) is on a class B network, as 0x84... = **10**00... (base 2).

  Therefore, the address 132.199.15.99 can be split into an network-address of 132.199.0.0 and an host-address of 15.99.
- Class C is identified by the MSBs being "110", allowing only 256 (actually: only 254, see below) hosts on each of the 2^21 possible class C networks. Class C addresses are usually found at (small) companies.
- There are also other addresses, starting with "111". Those are used for special purposes (e. g. multicast-addresses) and are not of interrest here.

Please note that the bits which are used for identifying the network-class are part of the network-address.

When seperating host-addresses from network-addresses, the "netmask" comes in handy. In this mask, all the network-bits are set to "1", the host-bits are "0". Thus, putting together IP-address and netmask with a locical AND-function, the network-address remains.

To continue our example, 255.255.0.0 is a possible netmask for 132.199.15.99. When applying this mask, the network-address 132.199.0.0 remains.

By default, every network-class has a fixed netmask assigned:

Class A: default-netmask: 255.0.0.0, first byte of address: 1-127

Class B: default-netmask: 255.255.0.0, first byte of address: 128-191

Class C: default-netmask: 255.255.255.0, first byte of address: 192-223

Another thing that should be mentioned here is the "broadcast-address". When sending to this address, *all* hosts on the corresponding network will receive the message sent. The broadcast address is characterized by having all host-bits set to "1".

Taking 132.199.15.99 with its netmask 255.255.0.0 again, the broadcast-address would result in 132.199.255.255.

You'll ask now: But what if I want a hosts address to be all bits "0" or "1"? Well, this doesn't work, as network- and broadcast-address must be present! Because of this, a class B network can contain at most 2^16-2 hosts, a class C network can hold no more than 254 hosts.

Besides all those categories of addresses, there's the special IP-address 127.0.0.1 which always refers to the "local" host, i. e. if you talk to 127.0.0.1 you'll talk to yourself without starting any network-activity. This is sometimes useful to use services installed on your own machine or to play around if you don't have other hosts to put on your network.

Let's put together the things we've introduced in this section:

- IP-address: 32 bit-address, with network- and host-bits.
- Network-address: IP-address with all host bits set to "0".
- Netmask: 32-bit mask with "1" for network- and "0" for host-bits.

- Broadcast: IP-address with all host bits set "1".
- The local host's IP-number is always 127.0.0.1.

2.3 Subnetting and Routing

After talking so much about netmasks, network-, host- and other addresses, I have to admit that this is not the whole truth!

Imagine the situation at your university, which usually has a class B address, allowing it to have up to 65534 hosts on that net. Maybe it would be a nice thing to have all those hosts on one single network, but it's simply not possible due to limitations in the transport media commonly used today!

For example, when using thin wire ethernet, the maximum length of the cable is 185 meters. Even with repeaters in between, which refresh the signals, this is not enough to cover all the locations where machines are located. Besides that, there is a maximum number of 1024 hosts on one ethernet wire, and you'll loose quite a bit of performance if you go to this limit.

So, are you hosed now? Having an address which allows more than 60000 hosts, but being bound to media which allows far less than that limit?

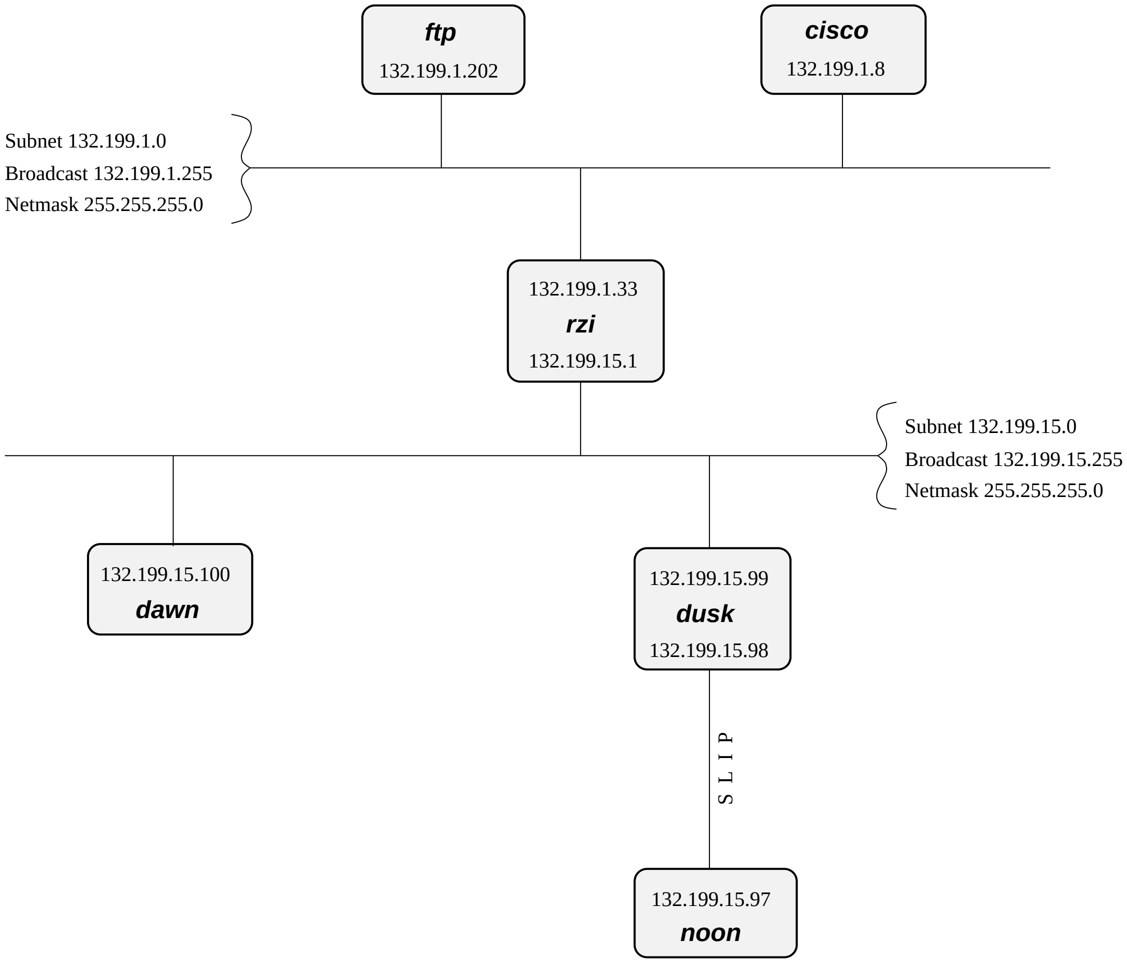
Well, of course not! :-)

The idea is to devide the "big" class B net into several smaller networks, commonly called sub-networks or simply subnets. Those subnets are only allowed to have, say, 254 hosts on them (i.e. you devide one big class B network into several class C networks!).

To do this, you adjust your netmask to have more network- and less host-bits on it. This is usually done on a byte-boundary, but you're not forced to do it there! (I don't really know if NetBSDs networking software understands subnets on non-byte-boundaries; Linux' is said to have problems there) So, commonly your netmask will not be 255.255.0.0 as supposed by a class B network, but it will be set to 255.255.255.0.

This gives you one additional network-byte to assign to each (physical!) network. All the 254 hosts on that subnet can now talk directly to each other, and you can build 256 such class C nets. This should fit your needs.

To explain this better, let's continue our above example. Let's have our host 132.199.15.99 (I'll call him DUSK from now; we'll talk about assigning hostnames later) have a netmask of 255.255.255.0 and thus being on the subnet 132.199.15.0. Let's furthermore introduce some more hosts so we have something to play around with:



(Picture 1: This demo-network shows a part of the University of Regensburg's campus-wide network as of March 1st 1994. All hosts except noon are really there.)

In the above network, DUSK can talk directly to DAWN, as they are both on the same subnet. (There are other hosts attached to the 132.199.15.0-subnet, I'm just too lazy to list them all ;-)

But what, if DUSK wants to talk to a host on another subnet?

Well, the traffic will then go through one or more gateways (routers), which are attached to two subnets. Because of this, a router always has two different addresses, one for each of the subnets.

The router is functionally transparent, i. e. you don't have to address it to reach hosts on the "other" side. Instead, you address that host directly and the packets will be routed to it correctly.

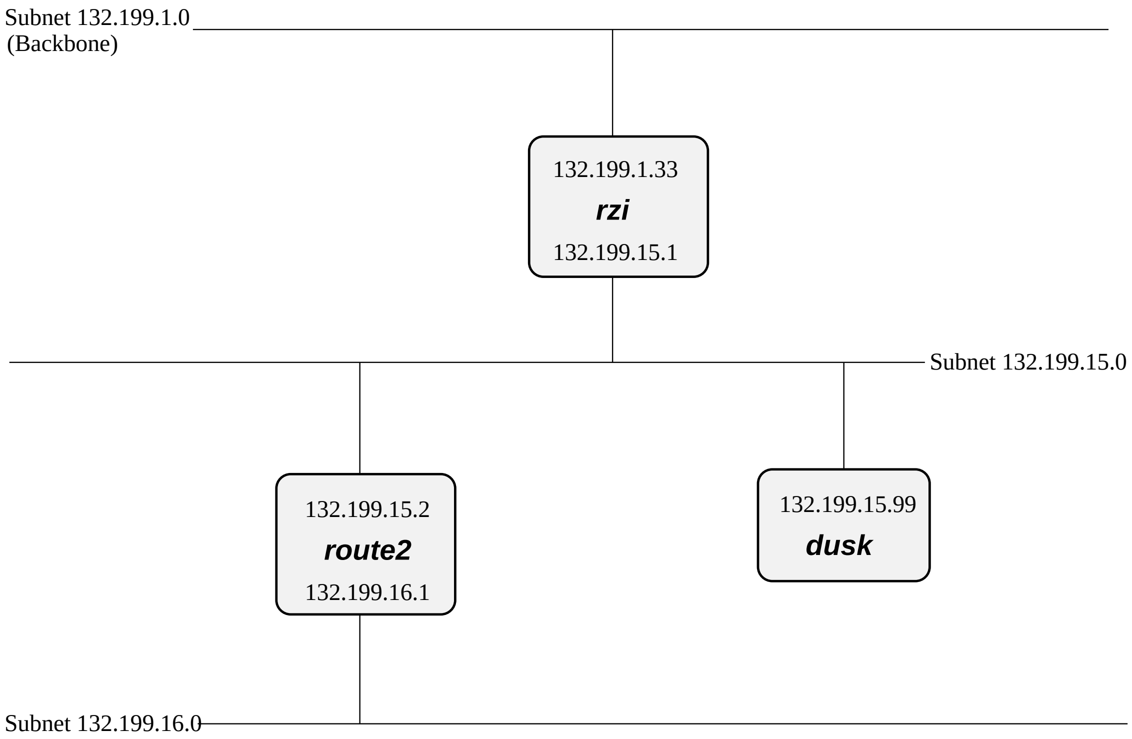
Example. Let's say DUSK wants to get some files from the local ftp-server. As DUSK can't reach FTP directly (because it's on a different subnet), all its packets will be forwarded to it's "defaultrouter" RZI (132.199.15.1), which knows where to forward the packets to.

DUSK knows the address of it's defaultrouter in its network (RZI, 132.199.15.1), and it will forward any packets to it which are not on the same subnet, i.e. it will forward all IP-packets in which the third address-byte isn't 15.

The (default)router then gives the packets to the appropriate host, as it's also on the FTP-server's network.

In this example, **all** packets are forwarded to the 132.199.1.0-network, simply because it's the network's backbone, the most important part of the network, which carries all the traffic that passes between several subnets. Almost all other networks besides 132.199.15.0 are attached to the backbone in a similar manner.

But what, if we had hooked up another subnet to 132.199.15.0 instead of 132.199.1.0? Maybe something like this:



(Picture 2: Attaching one subnet to another one.)

When you now want to reach a host which is located in the 132.199.16.0-subnet from DUSK, it won't work routing it to RZI, but you'll have to send it directly to ROUTE2 (132.199.15.2). DUSK will have to know to forward those packets to ROUTE2 and send all the others to RZI.

When configuring DUSK, you tell it to forward all packets for the 132.199.16.0-subnet to ROUTE2, and all others to RZI. Instead of specifying this default as 132.199.1.0, 132.199.2.0, etc., 0.0.0.0 can be used to set the default-route.

Returning to ***picture 1, there's a similar problem when DAWN wants to send to NOON, which is connected to DUSK via a serial line running. When looking at the IP-addresses, NOON seems to be attached to the 132.199.15.0-network, but it isn't really. Instead, DUSK is used as gateway, and DAWN will have to send its packets to DUSK, which will forward them to NOON then.

The same goes when hosts from other subnets want to send to NOON. They have to send their packets to DUSK (possibly routed via RZI),

2.4 Name service concepts

In the previous sections, when I talked about hosts, I referred to them by their IP-addresses. This was necessary to introduce the different kinds of addresses. When talking about hosts in general, it's more convenient to give them "names", as I did when talking about routing.

Most applications don't care whether you give them an IP-number or an hostname. However, they'll use IP-numbers internally, and there are several methods for them to map hostnames to IP-numbers, each one with its own way of configuration. In this section I'll introduce the idea behind each method, in the next chapter, I'll talk about the configuration-part.

The mapping from hostnames (and domainnames) to IP-addresses is done by a piece of software called the "resolver". This is not an extra service, but some library routines which are linked to every application using networking-calls. The resolver will then try to resolve (hence the name ;-) the hostnames you give into IP-numbers. See RFCs 1034 and 1035 for details on the resolver.

Hostnames are usually up to 10 characters long, and contain letters, numbers, dashes ("-") and underscores ("_"); case is ignorred.

Just as with networks and subnets, it's possible (and desirable) to group hosts into domains and subdomains. When getting your network-address, you also obtain a domainname by your provider. As with subnets, it's up to you to introduce subdomains. Other as with IP-addresses, (sub)domains are not directly related to (sub)nets; for example, one domain can contain several subnets. (Host- and domainnames will be written in SMALL CAPS in this document).

***Picture 1 shows this: Both subnets 132.199.1.0 and 132.199.15.0 (and others) are part of the subdomain "RZ.UNI-REGENSBURG.DE". The domain the University of Regensburg got from it's IP-provider is "UNI-REGENSBURG.DE" (".DE" is for Deutschland, Germany), the subdomain "RZ" is for Rechenzentrum, computing center.

Hostnames, subdomain- and domainnames are separated by dots ("."). It's also possible to use more than one stage of subdomains, although this is not very common. An example would be FOX_IN.SOCS.UTS.EDU.AU.

A hostname which includes the (sub)domain is also called a fully qualified domain name (FQDN). For example, the IP-address 132.199.15.99 belongs to the host with the FQDN DUSK.RZ.UNI-REGENSBURG.DE.

Further above I told you that the IP-address 127.0.0.1 always belongs to the local host, regardless what's the "real" IP-address of the host. Therefore, 127.0.0.1 is always mapped to the name "LOCALHOST".

2.4.1 '/etc/hosts'

The first and most simplest way to translate hostnames into IP-addresses is by using a table telling which IP-number belongs to which hostname(s). This table is stored in '/etc/hosts' and has the following format:

```
<IP-address>    <hostname> [<nickname> [...]]
```

Lines starting with a hash mark ("#") are treated as comments. The other lines contain one IP-address and the corresponding hostname(s).

It's not possible for a hostname to belong to several IP-numbers, even if I made you think so when talking about routing. RZI for example has really two distinct names for each of its two addresses: RZI and RZIA (but please don't ask me which name belongs to which address!).

Giving a host several nicknames can be convenient if you want to specify your favourite host providing a special service with that name, as is commonly done with FTP-servers. The first (leftmost) name is usually the real (canonical) name of the host.

Besides giving nicknames, it's also convenient to give a host's full name (including domain) as its canonical name, and using only its hostname (without domain) as a nickname.

Important: There *must* be an entry mapping localhost into 127.0.0.1!

2.4.2 The Domain Name Service (DNS)

'`/etc/hosts`' bears an inherent problem, especially in big networks: when one host is added or one hosts's address changes, all the '`/etc/hosts`'' on all machines have to be changed! This is not only time-consuming, it's also very likely that there will be some errors and inconsistencies, leading to problems.

Another appoach is to hold only one hostnames-table (-database) for a network, and make all the clients query that "name-server". Updates will be made only on the name-server.

This is the basic idea behind the Domain Name Service (DNS).

Usually, there's one name-server for each domain (hence DNS), and every host (client) in that domain knows which domain it is in and which DNS to query for its domain.

When the DNS gets a query about an host which is not in its domain, it will forward the query to a DNS which is either the DNS of the domain in question or knows which DNS to ask for the specified domain. If the DNS forwarded the query doesn't know how to handle it, it will forward that query again to a DNS one step higher. This is not ad infinitum, there are several "root"-servers, which know about any domain.

2.4.3 Network Information Service (NIS) / Yellow Pages (YP)

Yellow Pages (YP) was invited by Sun Microsystems. The name has been changed into Network Information Service (NIS) because YP was already a trademark of the british telecom. So, when I'll talk about NIS you'll know what I'm talking about. ;-)

There are quite some configuration files on a unix-system, and often it's desired to maintain only one set of those files for a couple of hosts. Those hosts are grouped together in a NIS-domain (which has **nothing** to do with the domains built by using DNS!) und are usually contained in one workstation cluster.

Examples for the config-files shared among those hosts are '`/etc/passwd`', '`/etc/group`' and — last but not least — '`/etc/hosts`'.

So, you can "abuse" NIS for getting a unique name-to-address-translation on all hosts throughout one (NIS-)domain.

There's only one drawback, which prevents NIS from actually being used for that translation: In contrast to the DNS, NIS provides no way to resolve hostnames which are not in the hosts-table. There's no hosts "one level up" which the NIS-server can query, and so the translation will fail! (Suns NIS+ seems to take measures against that problem, but as NIS+ is only available on Solaris-systems, this is of little use for us now.)

Don't get me wrong: NIS is a fine thing for managing e.g. user-information ('`/etc/passwd`', ...) in workstation-clusters, it's simply not useful for resolving hostnames!

3 Practice - Essential setup

In the previous chapter I've introduced all the basic concepts necessary for setting up networking even in non-trivial environments. Here, I will show you how to bring your machine up to use networking-applications such as finger, FTP and telnet.

Throughout this chapter, I'll use DUSK (see ***Picture 1) as an example for a host hooked up to an ethernet based network, and the connection between DUSK and NOON will show how to set up PPP and SLIP.

3.1 Requirements

There are several things which are needed to do networking. Most significant is — of course — the hardware you'll use. This has impact on all the other things, mainly the packages you've to compile into your kernel and the informations you need to get everything running.

3.1.1 Hardware

There are two possible types of networking hardware:

- Ethernet
- Serial line

3.1.1.1 Ethernet

There are drivers available for NetBSD-Amiga for the following Ethernet-cards:

- Commodore A2065
- Ameristar Board (sorry, don't know exact type)

The significant thing with ethernet is, that it uses a broadcast-medium, i.e. there can be several hosts attached to one cable.

3.1.1.2 Serial line

In order to use TCP/IP over a serial line, you only need a null modem. Here's how the pins are connected on the cable I use:

```
 2 <--->   3
 3 <--->   2
 4 <--->   5
 5 <--->   4
 6 <--->  20
 7 <--->   7
 8 <--->  20
20 <---> 6+8
```

This cable has proved to work with SLIP and PPP, as well as for using the other side as simple dumb terminal.

3.1.2 Kernel-requirements

Here are the necessary changes for '.../conf/MACHINE' in order to incorporate the various networking-facilities into your kernel:

- Enable TCP/IP networking in general:
```
option          INET            # Basic networking support - mandatory
pseudo-device   LOOP            # Loopback network - mandatory
```
- Put this entry in if you want to run SLIP:
```
pseudo-device   sl      1       # Serial Line IP (SLIP)
```
- This one is needed to do PPP:
```
pseudo-device   ppp     1       # Point-to-Point-Protocol (PPP)
```
- If you're proud owner of a Ameristar or Commodore A2065 Ethernet-card, add the following two items:
```
pseudo-device   ether           # Ethernet support
device le0      at manufacturer ?       product ?
```
- Enable the following two options, if you want to run your system as NFS-client or -server:
```
option          NFSSERVER       # NFS server side code
option          NFSCLIENT       # NFS client side code
```

All those options are already included in the GENERIC kernel. They are listed here rather as an hint for those who want to known what to *exclude* from a kernel. Leaving out all the networking stuff should save you about 300-500k of memory, but please note that e.g. X needs TCP/IP networking facilities.

3.1.3 Addresses: IP, Broadcast, Netmask, ...

If you're about to hook your machine up to your company's, school's or university's network (i.e. most probably the real Internet :), go to your local network-administrator and get the following informations:

- your IP-Number
- your host's name, including domain.
- Netmask
- Broadcast-address
- Defaultrouter (IP-number)
- Nameserver (primary and secondary)

If you're about to use SLIP or PPP, possibly via a telephone line, you'll probably need the following informations:

- Phone-number of your terminal-server
- Account, password etc. to get access to your terminal-server
- IP-number(s) of the terminal-server's dial-ins

If you just want to run your own little LAN at home, you can choose your own values for most of the things above:

IP-number:
: Choose an IP-number from either class B or C. As you're isolated from the internet, it doesn't really matter what address(es) you choose, as long as they are valid addresses (see Section 2.2 [TCP/IP address format], page 4).

  If you choose a couple of addresses, please pay attention that they are all in the same (sub)net! (see Section 2.3 [Subnetting and Routing], page 7)

Hostname:
: Any valid hostname you like, see Section 2.4 [Name service concepts (Theory)], page 10.

  You'd better not choose a domainname because you'll only have to type longer hostnames (and believe me, you'll have to type those hostnames quite some times during tuning your network! :-).

Netmask: Determine this one according to the rules from Section 2.2 [TCP/IP address format], page 4. As you surely don't want to invent subnets, the netmask goes hand in hand with the IP-number(s) you choose.

Broadcast-address:
: If there's no 4.2BSD-system on your network, determine your broadcast-address after the rules stated in Section 2.2 [TCP/IP address format], page 4, i.e. set all host-bits to "1".

  If you've got one or more 4.2BSD-systems on your network, you've to pay attention to set the right broadcast-address, as 4.2BSD has a bug in its networking code, concerning the broadcast address. This bug forces you to set all host-bits in the broadcast-address to "0"!!!

Defaultrouter:
: Most probably not needed at home.

Nameserver:
: You don't need this for the first steps, and most probably you won't set up DNS at home. See Section 2.4.2 [Domain Name Service (Theory)], page 12, for some details.

To illustrate this, I'll give you the addresses for DUSK and for connecting DUSK and noon.

Example 1: DUSK

- IP-number: 132.199.15.99
- Netmask: 255.255.255.0
- Broadcast: 132.199.15.255
- Name: DUSK.RZ.UNI-REGENSBURG.DE
- Defaultrouter: 132.199.15.1 (RZI.RZ.UNI-REGENSBURG.DE)
- Nameserver(1): 132.199.1.2
- Nameserver(2): 132.199.1.1

I got all these values from the local network admin. As the system is connected to the internet, I use the University of Regensburg's class C network (132.199.0.0) and their domainname (UNI-REGENSBURG.DE).

Example 2: DUSK & noon

- DUSK:
  - IP-number: 132.199.15.98

  - Broadcast: 132.199.15.255
  - Name: DUSK
- noon:
  - IP-number: 132.199.15.97
  - Broadcast: 132.199.15.255
  - Name: noon

I used the second setup at home, with no connection to the internet. Therefore, I have chosen neither domainname nor defaultrouter or nameserver. (I choose those IP-numbers for the case that I'm going to hook up noon to the net, just for fun :-).

3.2 Configuring the interface

I'll tell you here what's to do to get up your network connection. I'll tell you how you can set this up permanently in '/etc/rc' etc. later, see Section 3.5 [How to use on '/etc/*'], page 24.

Before configuring any network-device, let's first configure the loopback device:

```
# ifconfig lo0 inet 127.0.0.1
```

By now, you should be able to ping 127.0.0.1.

3.2.1 Configuring your ethernet-board

Enter the following:

```
# ifconfig le0 inet <ip-number> netmask <netmask> broadcast <broadcast>
```

For example, I can use the following commands to configure DUSK:

```
# ifconfig le0 inet 132.199.15.99 netmask 255.255.255.0 \
              broadcast 132.199.15.255
```

If you've got a defaultrouter on your network, route all unknown packets to it:

```
# route add default <defaultrouter-ip>
```

E.g.

```
# route add default 132.199.15.1
```

After that, you can try to reach several hosts on the local and other networks with the ping-command. Here's what I did on DUSK to check if everything's fine (please refer to ***Picture 1):

```
# ping 132.199.15.99            # dusk.rz.uni-regensburg.de
# ping 132.199.15.100           # dawn.rz.uni-regensburg.de
# ping 132.199.15.1             # rzi.rz.uni-regensburg.de
# ping 132.199.1.202            # ftp.uni-regensburg.de
# ping 128.252.135.4            # wuarchive.wustl.edu
```

3.2.2 Setting up serial protocols

Here, I'll tell you how to set up NetBSD for dial-out, either on directly connected machines or via modem.

Any hints for how to setup dial-in welcome (`sliplogin`, ...)!

Before you start setting up anything, be sure to kill your getty first:

1. comment out the line starting with '`/dev/tty00`' in '`/etc/ttys`'
2. Get the kernel to read the new '`/etc/ttys`': `kill -HUP 1`
3. Kill any still-running gettys: `` kill -9 `ps -aux | grep gett[y] | awk '{ print $2; }'` ``

3.2.2.1 Point to Point Protocol (PPP)

Here's what Markus Landgraf (`landgraf@crunch.ikp.physik.th-darmstadt.de`) does:

1. ifconfig ppp0
   ```
   # ifconfig ppp0 inet <local-ip> -arp -trailer <remote-ip>
   ```
2. Connect to remote machine via kermit:

```
kermit> set line /dev/tty00
kermit> set speed 9600          # or whatever
kermit> set flow rts/cts
kermit> connect
```

If your're using a modem, you'll have to dial before connecting:

```
kermit> dial <your terminal-server's phone number>
```

Log into your remote machine and start `dplogin`, `pppd` or whatever's used to start PPP on the remote site.

After that (when you get weird chars on your display) terminate kermit (CTRL-`\` q) and perform the next step *fast* to avoid a timeout.

3. run pppd

```
# pppd /dev/tty00 9600
```

4. ppp0 up

```
# ifconfig ppp0 up
```

5. Turn on routing:

```
# route add 0.0.0.0 <remote-ip>
```

6. ping some remote site, see Section 3.2.1 [Configuring your ethernet-board], page 19. Those pings should succeed.

3.2.2.2 Serial Line IP (SLIP)

The steps for SLIP are basically the same:

1. Configure sl0:

```
# ifconfig sl0 inet <local-ip> -arp -trailers <remote-ip>
```

2. Connect to remote machine via kermit:

```
kermit> set line /dev/tty00
kermit> set speed 9600          # or whatever
kermit> set flow rts/cts
kermit> connect
```

If your're using a modem, you'll have to dial before connecting:

```
kermit> dial <your terminal-server's phone number>
```

Log into your remote machine's SLIP-account or start SLIP by hand (using `slattach` or such; please consult your network-admin!).

After that (when you get weird chars on your display) terminate kermit.

3. Start up the local SLIP service:

```
# slattach -s 9600 /dev/tty00
```

4. Enable the network connection

```
# ifconfig sl0 inet up
```

5. Turn on routing:

```
# route add 0.0.0.0 <remote-ip>
```

6. ping some remote site, see Section 3.2.1 [Configuring your ethernet-board], page 19. Those pings should succeed.

You can use higher baud-rates than 9600 on both, PPP and SLIP. NetBSD supports baud-rates up to 38400Bd.

3.2.2.3 term

As I don't own a modem, I haven't made any experiences with term, sorry.

From what I know, term is not really a method to do TCP/IP, but rather to tunnel TCP/IP-packets over a serial line using its own protocol and some own servers on both sides. Markus Wild's comment on term: "A Linux-beast".

Anybody who likes to put some wise words here, please contact me (first)!

3.3 Routing

Let's talk one more word about routing. When running SLIP or PPP, it's sufficient to have a `route add default <remote-ip>` somewhere. However, if you want to hook up your machine to a more complex network, it's wise to use `routed` instead of static routes. Therefore, set `routed_flags` to `"-q"` in '`/etc/netstart`' then, and it will listen for routing-updates.

If you are a gateway yourself (e.g. SLIP/PPP-Ethernet), set `routed_flags` to `""` instead to advertise that route. Also, if there are several gateways on your network, put information about them into '`/etc/gateways`'.

For example, when DUSK is the SLIP-gateway for NOON (see ***Picture 1), I set `routed_flags` to `""` and put the following into DUSKs '`/etc/gateways`':

```
host    132.199.15.97   gateway 132.199.15.98   metric 1 active
```

This example establishes a route to NOON (132.199.15.97) via DUSKs SLIP-interface (132.199.15.98). "metric 1" says that NOON is one hop away from DUSK, i.e. that it's directly connected.

3.4 Translating names to IP-numbers

At this point, you should be able to use all TCP/IP-applications such as ftp, telnet, etc. But up to now, you have to specify all hosts by their IP-number, which is not very convenient. So, here are the different ways to set up IP-to-name-resolving.

3.4.1 '/etc/hosts'

As explained previously (see Section 2.4.1 ['/etc/hosts'], page 11), '/etc/hosts' contains a table telling which hostname to map to which IP-number.

If you plan to use DNS, you will nevertheless have at least entries for localhost (127.0.0.1), your hostname (with it's own IP-number) and maybe the defaultrouter in '/etc/hosts'.

For example, here's a minimal '/etc/hosts' for DUSK:

```
# /etc/hosts
127.0.0.1       localhost
132.199.15.99   dusk
```

Besides these two entries, it's convenient to put any hosts into it which your system relies on, e.g. NFS-servers. This way, you can reach those hosts even during boot-time or if DNS is down.

But, if you're just running your private network at home, it's sufficient to put all your hostnames in '/etc/hosts', there's no need to set up DNS at home!

3.4.2 Domain Name Service (DNS)

The Domain Name Service is the usual way to resolve IP-numbers from hostnames in larger networks. All you have to know to set it up is your domainname and the nameserver's (and maybe it's secondaries, if any) IP-numbers. Put all these informations into '/etc/resolv.conf':

```
# Example /etc/resolv.conf
domain your.domain.here
nameserver <primary-IP>
nameserver <secondary-ip>
```

As an example, here's DUSKs '/etc/resolv.conf':

```
# dusk's /etc/resolv.conf
domain rz.uni-regensburg.de
nameserver 132.199.1.2
nameserver 132.199.1.1
```

This file is all that's necessary to use DNS.

3.4.3 Network Information Service (Yellow Pages)

Sorry, I haven't tried NetBSD's NIS yet. If anyone did, please tell me!

Besides that, there's no server-code for NetBSD, so you need another machine, which is able to run a NIS-server.

3.5 How to use the above on '/etc/*'?

Now, as you know how to set up everything by hand, I'll tell you how to use that knowledge to change the config-files to get all networking-services started at boot-time.

I want to go through the following files from '/etc' and show you what you need and what it's good for:

- '/etc/netstart'
- '/etc/rc'
- '/etc/rc.local'

3.5.1 '/etc/netstart'

```
#!/bin/sh -
```

```
#
#       @(#)netstart   5.9 (Berkeley) 3/30/91
#       $Id: netstart,v 1.15 1994/01/10 16:57:24 mycroft Exp $

# set these to "NO" to turn them off.  otherwise, they're used as flags
routed_flags=NO         # for 'normal' use: routed_flags=-q
```

As stated before, the following rules apply to `routed_flags`:

- `NO`: no need for routed if you're hooked up via SLIP or PPP.
- `"-q"`: Useful if you've got a Ethernet-card and you are hooked up to a non-trivial network (i.e. there's at least one gateway, ...)
- `""`: Use this if your machine's a gateway itself.

```
rarpd_flags=NO          # for 'normal' use: rarpd_flags="-a"
```

If you want to become a RARP-server (Reverse Address Resolution Protocol, converts Ethernet- to IP-addresses; see RFC903), enable this. Do this only if you know what you do, and read `rarpd`(8) before!

```
bootparamd_flags=NO     # for 'normal' use: bootparamd_flags=""
```

Set this to `""` to run the bootparamd-RPC-service which is needed for remote boot of diskless clients. Do this only if you know what you do, and read `rpc.bootparamd`(8) before!!!

```
sendmail_flags=NO       # for 'normal' use: sendmail_flags="-bd -q30m"
```

If you want to send and receive mail, you'll need to set this to `"-bd -q30m"` or any appropriate settings that fit your needs. You will also need a properly configured '`/etc/sendmail.cf`' for this to run.

Warning! If you're not on your own network, please consult your postmaster before doing anything fatal. It's **very** easy to produce mailloops etc. which can blow your whole site's mailsystem!!!

```
timed_flags=NO
```

Leave this at `"NO"`, it doesn't help anyway.

The `timed` is thought to keep all the clocks on a network in sync, but it doesn't help with that CIA-timer-inaccuracy in NetBSD-Amiga (744).

```
# set the following to "YES" to turn them on, "NO" to disable.
rwhod=NO
```

Set this to `NO`. `rwhod` is good for burning quite some CPU-cycles to tell other hosts on your network who's logged on. Set to `"YES"` if you want to be able to use `rwho` anyway.

```
nfs_server=YES
```

This is useful if you've got some directories to export to other machines via NFS. If you do so, set it to `YES`. If you want to mount your own disks via NFS (which is quite nonsense, but nevertheless possible), do so, too.

Set it to `NO` otherwise. E.g. it's *definitely* no fun to do NFS-mounts via a SLIP- or PPP-link, as this will be dead slow.

```
nfs_client=YES
```

If there's a host on your network which disks you want to use or you want to mount your own disks (see above), set this to `YES`. Set to `NO` otherwise.

```
name_server=NO
```

Leave at `"NO"` unless you know how to set up your own nameserver. See [CraigHunt] for details.

```
gated=NO
```

Leave at `"NO"`.

This is a replacement for routed which is only useful in very complex network-setups, e. g. if you need to set up wide-area networking (WAN).

```
kerberos_server=NO
```

I've never used this, so you can most probably live without it, too. Set to `YES` if your site depends on Kerberos-security.

```
amd=NO
```

If you're a NFS-client and don't want to mount all the remote disks all the time, you can mount them "on demand" using the Auto Mount Daemon `amd`.

If you want to use this, read `amd`'s man-page.

```
# miscellaneous other flags
# gated_flags only used if gated == YES
gated_flags=
```

If you need to use `gated`, put the appropriate flags for it to run here.

```
# /etc/myname contains my symbolic name
#
hostname=`cat /etc/myname`
hostname $hostname
```

Put your host's name without domain into '`/etc/myname`', e.g. `echo dusk >/etc/myname`

```
if [ -f /etc/defaultdomain ]; then
        domainname `cat /etc/defaultdomain`
fi
```

This is only used by NIS/YP, so if you're about to use NIS, put the name of your NIS-domain into '`/etc/defaultdomain`', e.g. `echo nis1.rz.uni-regensburg.de >/etc/defaultdomain`.

Beware! The domainname used here has *nothing* to do with the domain introduced by the domain name service (DNS)!

```
# configure all of the interfaces which we know about.
# do this by reading /etc/hostname.* files, where * is the name
# of a given interface.
#
# these files are formatted like the following, but with no # at the
# beginning of the line
#
# addr_family hostname netmask broadcast_addr options
# dest dest_addr
#
# addr_family is the address family of the interface, generally inet
# hostname is the host name that belongs to the interface, in /etc/hosts.
# netmask is the network mask for the interface.
# broadcast_addr is the broadcast address for the interface
# options are misc. options to ifconfig for the interface.
#
```

```
# dest is simply the string "dest" (no quotes, though) if the interface
# has a "destination" (i.e. it's a point-to-point link, like SLIP).
# dest_addr is the hostname of the other end of the link, in /etc/hosts
#
# the only required contents of the file are the addr_family field
# and the hostname.

(
    tmp="$IFS"
    IFS="$IFS."
    set -- `echo /etc/hostname.*`
    IFS=$tmp
    unset tmp

    while [ $# -ge 2 ] ; do
        shift            # get rid of "hostname"
        (
            read af name mask bcaddr extras
            read dt dtaddr

            if [ ! -n "$name" ]; then
                echo "/etc/hostname.$1: invalid network configuration file"
                exit
            fi

            cmd="ifconfig $1 $af $name "
            if [ -n "$mask" ]; then cmd="$cmd netmask $mask"; fi
            if [ -n "$bcaddr" ]; then cmd="$cmd broadcast $bcaddr"; fi
            cmd="$cmd $extras"
            if [ "${dt}" = "dest" ]; then cmd="$cmd $dtaddr"; fi

            $cmd
        ) < /etc/hostname.$1
        shift
    done
)
```

First, please note that the order of arguments of the `ifconfig`-command, which are built here, might be different in your '`/etc/netstart`'. Put them in the above order (using your favourite editor), paying special attention that the destination-address (if any) is the last option to the ifconfig-command, after those extra-options!!!

Here's what `diff` says:

```
*** /usr/src/current/etc/netstart       Thu Feb  3 20:35:52 1994
--- /etc/netstart       Mon Mar 14 12:27:35 1994
***************
```

```
*** 73,83 ****
              fi

            cmd="ifconfig $1 $af $name "
-           if [ "${dt}" = "dest" ]; then cmd="$cmd $dtaddr"; fi
            if [ -n "$mask" ]; then cmd="$cmd netmask $mask"; fi
            if [ -n "$bcaddr" ]; then cmd="$cmd broadcast $bcaddr"; fi
            cmd="$cmd $extras"
--- 73,84 ----
            if [ -n "$mask" ]; then cmd="$cmd netmask $mask"; fi
            if [ -n "$bcaddr" ]; then cmd="$cmd broadcast $bcaddr"; fi
            cmd="$cmd $extras"
+           if [ "${dt}" = "dest" ]; then cmd="$cmd $dtaddr"; fi

            $cmd
          ) < /etc/hostname.$1
```

After that, create appropriate files '/etc/hostname.*', which describe your network-device(s):

Ethernet: Put the following into '/etc/hostname.le0':

```
inet <hostname> <netmask> <broadcast>
dest
```

SLIP: Put the following into '/etc/hostname.sl0':

```
inet <local-hostname> <netmask> <broadcast>
dest <remote-hostname>
```

PPP Put the following into '/etc/hostname.ppp0':

```
inet <local-hostname> <netmask> <broadcast>
dest <remote-hostname>
```

Note also that both, the local and the remote host together with their IP-numbers must be in '/etc/hosts', as the resolver and default-router are not known at that time (and which you need to use the DNS).

```
# set the address for the loopback interface
ifconfig lo0 inet localhost

# use loopback, not the wire
route add $hostname localhost
```

As the comments say, this configures the loopback-device (127.0.0.1, localhost), so don't forget this in '/etc/hosts'. Furthermore, packets which are sent to `$hostname` will go to straight back instead of using any Ethernet, PPP- or SLIP-device.

```
# /etc/mygate, if it exists, contains the name of my gateway host
# that name must be in /etc/hosts.
if [ -f /etc/mygate ]; then
        route add default `cat /etc/mygate`
fi
```

If you're on a subnetted network, here's the chance to set up your default-router when booting up: just put it's name into '/etc/mygate'. For example, on DUSK (see ***Picture 1) I did `echo 132.199.15.1 >/etc/mygate`.

Note that you can use a hostname here, but it has to be in '/etc/hosts', as the nameserver is most probably not in your subnet and thus wouldn't be reachable at boottime to resolve the router's name.

3.5.2 '/etc/rc'

There's only NIS left, for which there isn't a flag yet:

```
if [ -f /usr/sbin/ypbind -a -d /var/yp ]; then
        echo -n ' ypbind';              ypbind
fi
```

This is only started if the directory '/var/yp' exists. As there need to be several config- and datafiles in this directory in order to have a working NIS, be sure that you know what you do when creating '/var/yp'.

3.5.3 '/etc/rc.local'

Besides starting local daemons, '/etc/rc.local' is useful for either starting `pppd` or `slattach`. In order to not block any networking-services that are also started in '/etc/rc.local', the corresponding command should occur quite early, best place is after '/etc/motd' is generated.

Configure '/etc/hostname.ppp0' or '/etc/hostname.sl0' and '/etc/mygate' as described above. Also, change the baudrate to fit your needs.

SLIP: In order to start up SLIP at boottime, insert the following lines into '/etc/rc.local':

```
# Start SLIP-networking
echo -n 'Preparing SLIP-interface ... '
```

```
slattach 9600 /dev/tty00 >/dev/null 2>&1
echo -n 'ready.'
```

PPP: This should work, although I've never tried it:

```
# Start PPP-networking
echo -n 'Preparing PPP-interface ... '
pppd /dev/tty00 9600
echo -n 'ready.'
```

The big problem that stays with PPP is that you've to start it up on the other side at (exactly) the same time, and there must no timeout occur in order to get a connection. Use SLIP if this is a problem.

Note that there's barely a way to dial out during boot-time, so the above mainly belongs to direct (nullmodem) connections.

4 Advanced features and how to set them up

4.1 Anonymous FTP server

Read `ftpd`(8).

4.2 Network File System (NFS)

Sun's Network File System (NFS) has become a standard for using remote disks not only in the Unix- but in the whole TCP/IP-world. The idea is quite simple: one host "exports" a directory, and other hosts can mount that directory and access it and its subdirectories.

When users from several machines want to use the same set of files, special care has to be taken for the user-ids and group-ids the files have: UID and GID of the users must be unique across the (NFS-)network, or one won't be able to read a file on one machine created on another one (where the same user had a different UID).

There are also some security-mechanisms built in to prevent unauthorized machines from mounting directories or which prevent root-access to files from remote machines. I won't introduce those mechanisms here and I'll assume no special measures for mounting/exporting filesystems here. If you need to know about those mechanisms, please read `mount`(8), `exports`(5).

See RFC 1094 for a description of NFS.

4.2.1 Mounting remote filesystems

Mounting a directory from a remote host is pretty simple. All you have to know is the host's name (`remote-host`), the directory exported by the remote host (`remote-dir`) and the directory from which you want to access those files (`local-dir`, must be absolute!).

All you have to do then is:

```
# mount <remote-host>:<remote-dir> <local-dir>
```

To make the same mount permanent, put the following line into '`/etc/fstab`' (See `mount`(8) for a description of all those options: `rw,soft,`...):

```
<remote-host>:<remote-dir>  <local-dir>  nfs  rw,soft,bg,retry=4  0  0
```

Here's an example I use on DUSK: How to mount '`/usr/aftp/pub/os/NetBSD/NetBSD-Amiga`' from ftp.uni-regensburg.de (which is only an alias for the rrzs3) on DUSKs '`/usr/ftp/pub/NetBSD-Amiga`'. This can be done by issuing `mount ftp.uni-regensburg.de:/usr/aftp/pub/os/NetBSD/NetBSD-Amiga /usr/ftp/pub/NetBSD-Amiga` or putting the following line into '`/etc/fstab`':

```
rrzs3:/usr/aftp/pub/os/NetBSD/NetBSD-Amiga /usr/ftp/pub/NetBSD-Amiga nfs
                                                    rw,soft,bg,retry=4 0 0
```

(This line is split only to fit on the page. Put this all in one line!)

4.2.2 Exporting filesystems

To mount a directory from a remote host, the host has to export that directory via NFS. To do this, put the directorys name into '`/etc/exports`' on the remote host. Then issue `showmount -e 127.0.0.1` to (re-)read '`/etc/exports`' and actually export that filesystem. Also, this command will show you all the directories you currently export.

Here's what FTP.UNI-REGENSBURG.DE's '`/etc/exports`' looks like to give DUSK (and everyone else) access to '`/usr/aftp/pub/NetBSD-Amiga`':

```
/usr/aftp/pub -rw=dusk.rz.uni-regensburg.de,root=dusk.rz.uni-regensburg.de
```

Again, there are a number of options to restrict access. Please refer to `export`(5) for documentation.

4.3 Berkeley r-tools

The university of Berkeley has developped its own set of networking applications, of which the most important are:

`rlogin`: Interactive login into remote host, similar to `telnet`.

`rsh`: Execute command on remote host. Stdin is read from the local stdin, stdout and stderr of the remote command are returnet to the local host.

`rcp`: Copy file from local to remote machine or vice versa. The difference to `ftp` is that `rcp` is not used interactively but via commandline-arguments similar to the `cp`-command.

4.3.1 Prerequirements/Security

All the r-tools are based on the concept of trusted hosts and users, i.e. on one host, you say which user(s) from what host(s) you allow to access a specific account. There are two places where this information is kept:

- '`/etc/hosts.equiv`': Systemwide information, should be either removed or zero-length (`cp /dev/null /etc/hosts.equiv`) as this file is mostly a big security hole.
- '`~/.rhosts`': This file contains information on which users and hosts to allow to login or execute commands (via `rsh`). If you're really upset about your system's security, keep your users from having such files.

Both files contain pairs of *host*-*user*-combinations, where *host* is the host that users are allowed to log in from, and *user* tells *which* user is actually allowed to log in from that host (to that specific account, in the case of '`~/.rhosts`'.

Example! I've got an account `"c9020"` on RRZSG1.RZ.UNI-REGENSBURG.DE. When I want to login into hubert's account on DUSK without giving a password, I've got to put the following into hubert's '`~/.rhosts`':

```
rrzsg1.rz.uni-regensburg.de c9020
```

If you've trouble what to take as hostname (i.e., with or without domain, or even IP-number), login (probably *with* giving a password), then start `who`. This will tell you the hostname you've to put into your '`~/.rhosts`':

```
dusk% who
hubert   ttyp0   Mar 21 13:59   (rrzsg1.rz.uni-reg)
```

This shows that I have to use *rrzsg1.rz.uni-regensburg.de* as hostname.

4.4 X11

The most common question concerning networking and X is "How far do I have to start networking to be able to work with X?".

Well, it should be sufficient to configure the loopback-device properly. As this is done by default, there should be no network-problems with X.

Tell me if this is wrong!

4.5 Domain Name Server (DNS)

See RFCs 1032 and 1033 for guides on operation and domain administration.

See also [CraigHunt] for a detailed description.

4.6 Mail

In order to set up electronic mail, there are several steps to be performed:

1. Set `sendmail_flags` to `-bd -q30m` in '`/etc/netstart`'. This tells sendmail to start as a daemon (`-bd`) and scan the queues for mail every 30 minutes (`-q30m`).
2. Get a '`/etc/sendmail.cf`', e.g. from sun-lamp or one of its mirrors.
3. Ask your DNS-admin for an MX-entry on your host and — if your machine's not always on the net — also one on a machine which stores your mail while your machine's down and forwards it later.

Most of the time, sending mail is no problem, but receiving is. So, if you experience any problems, consult your local postmaster!

4.7 Remote Printing

This is a topic which I haven't tried out yet, but which I'd really like to see here. If anyone has detailed information about

1. using a remote printer
2. offering print services

please tell me and I'll insert it here!

5 FAQs

5.1 How do I set up networking?

Read the "NetBSD-Amiga Beginners Guide to Networking and Networking-FAQ".

5.2 I've choosen two IP-numbers, 1.1.1.1 and 2.2.2.2, but nothing works!

These two numbers are in different subnets, so you either have to

- set up routing propperly for them to work, or
- choose numbers which are in the same subnet. See Section 2.2 [TCP/IP address format], page 4.

5.3 Netstat doesn't output anything

Please ensure that the running Kernal is the same as '`/vmunix`'.

5.4 I can't get `vmunix.613` to work with my Ethernet-board

Ethernet-support was first introduced in 622, so you've to update your kernel and some of the networking-programs.

5.5 The system hangs when going into multiuser-mode

Set `name_server=NO` in '`/etc/netstart`' or set up your '`/etc/resolv.conf`' properly to get access to the DNS.

5.6 `timed` and `routed` report some errors. Should I comment them out, too?

You can if you want, although those don't disturb the rest of the system, they may just fail, but so what. The corresponding flags in '`/etc/netstart`' are:

- routed_flags (set to NO to disable)
- timed_flags (set to NO to disable)

5.7 `xhost` says "must be on local host", but I'm already there!

Try setting your `DISPLAY` to "`:0`". There seem to be some problems when using "`localhost:0`" or "<nodename>`:0`".

5.8 `ifconfig` doesn't init my point-to-point-devices (SLIP/PPP) right

Try setting the remote IP-address as the very last argument at the ifconfig-command. If you want this to run from '`/etc/netstart`'/'`/etc/hostname.*`', please note the options' order given in '`/etc/netstart`' and fix your '`/etc/netstart`', if necessary (see Section 3.5.1 ['`/etc/netstart`'], page 24).

5.9 What's the Major and Minor device numbers for the le0 device?

There's no `/dev/le0`, and so you can't figure out any major/minor number. If you want to check whether you've got an ethernet-driver in you kernel, do `netstat -i` and watch out for `le0` there.

Appendix A Abbreviations

CSRG	Computer Systems Research Group, core developers of BSD
DARPA	Defense Advanced Research Projects Agency, sponsor for developing TCP/IP
DDN	Data Defense Network
DNS	Domain Name Service, method to map hostnames to IP-addresses and back
FTP	File Transfer Protocol, program & TCP/IP-based protocol to transfer single files between machines.
ISO	International Standard Organisation, defined networking-protocols such as X.25, X.400, X.500, ...
NFS	Networking File System, gives transparent access to remote files
NIS	Network Information Service, method to share one database (e. g. '`passwd`'-file) between several machines; former Yellow Pages (YP)
PPP	Point to Point Protocol, transports several protocols (TCP/IP, DECnet, ... over serial lines
RFC	Request For Comment, open definition of internet standards
SLIP	Serial Line IP, transports IP-packets over serial line
TCP/IP	Transmission Control Protocol/Internet Protocol, most widespread networking protocol today
UCB	University of California at Berkeley; origin of the BSD-Unix and NetBSD
X11	Version 11 of the X-Window-System developped at MIT
YP	Yellow Pages, see NIS; renamed after conflicts with british telecom.

Appendix B References

[AeleenFrisch]
Aeleen Frisch: "Essential System Administration", O'Reilly & Associates, Sebastopol, 1991.

[CraigHunt]
Craig Hunt: "TCP/IP Network Administration", O'Reilly & Associates, Sebastopol, 1993.

[Leffer] Samuel J. Leffer, Marshall Kirk McKusick, Michael J. Karels, John S. Quarterman: "The Design and Implementation of the 4.3BSD UNIX Operating System", Addison Wesley, Reading, 1989.

[RFC977] B. Kantor, P. Lapsley: "Network News Transfer Protocol", February 1986, 27 pages.

[RFC1032]
M. Stahl: "Domain administrators guide", November 1987, 14 pages.

[RFC1033]
M. Lotter: "Domain administrators operations guide", November 1987, 22 pages.

[RFC1034]
P. Mockapetris: "Domain names - concepts and facilities", November 1987, 55 pages.

[RFC1035]
P. Mockapetris: "Domain names - implementation and specification", November 1987, 55 pages.

[RFC1055]
J. Romkey: "Nonstandard for transmission of IP datagrams over serial lines: SLIP", June 1988, 6 pages.

[RFC1094]
Sun Microsystems, Inc.: "NFS: Network File System Protocol specification.", March 1989, 27 pages.

[RFC1331]
W. Simpson: "The Point-to-Point Protocol (PPP) for the Transmission of Multi-protocol Datagrams over Point-to-Point Links", May 1992, 66 pages.

[RFC1332]
G. McGregor: "The PPP Internet Protocol Control Protocol (ICPC)", May 1992, 12 pages.

Table of Contents