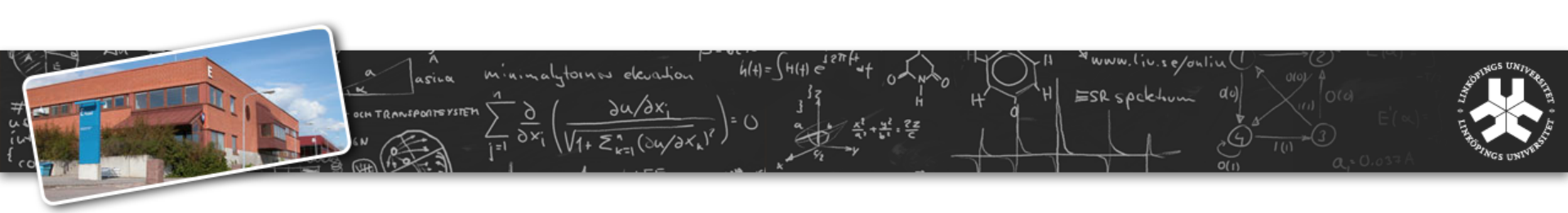


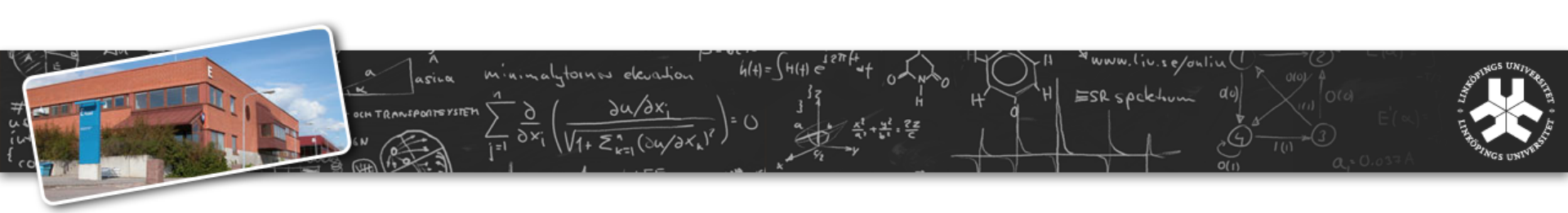
TDDI41

- Agenda
  - Frågor?
  - Labbar
  - Process, singnals, logging
  - Linux networking

LiU expanding reality

Labbar

- Samma hårdvara som tidiagre år
- Alla labbar är genomförda innan kursstart
- Varit uppe I flera dagar
  - Innan ni loggade på , hmmm
- Inget i någon logg
  - Ingen kontakt med varken nät eller vga/tangenbord/mus
- Bytt ut alla delar
- Ny kernel, bootas om nu (4.6)
- Vi har testat med att dra igång 50 UML utan att kunna återskapa felet.
- Efter föreläsning prova gärna att stresstesta systemet



Linux Process, Signals & Logging

LiU expanding reality

A Look at Linux: Processes

- All execution takes place in processes
  - Each process may consist of several threads
  - Every process has its own (protected) address space
  - Every process has an ID, a parent, and a controlling tty
  - Processes have a state (running, stopped, suspended, etc)

- Processes can communicate
  - Signals are simple asynchronous messages
  - Processes can share memory areas
  - Processes can communicate using pipes
  - Processes can communicate using sockets

Example of processes

```
%  ps -H -eo s,pid,ppid,tty,user,cmd
S    PID   PPID TT       USER     CMD
S      1      0 ?        root     init [2]
S   2188      1 ?        snmp       /usr/sbin/snmpd -Lsd -Lf /dev/null -u snmp
S   2194      1 ?        root       /usr/sbin/sshd
S  24294   2194 ?        root         sshd: davby [priv]
S  24296  24294 ?        davby          sshd: davby@pts/0
S  24297  24296 pts/0    davby            -sh
R  24304  24297 pts/0    davby              ps -H -eo s,pid,ppid,tty,user,cmd
S   2206      1 ?        uml-net    /usr/bin/uml_switch -tap tap0 -unix
S   2273      1 ?        statd      /sbin/rpc.statd
S   2297      1 ?        root       sendmail: MTA: accepting connections
S   2323      1 ?        ntp        /usr/sbin/ntpd -p /var/run/ntpd.pid
S   2333      1 ?        daemon     /usr/sbin/atd
```

Process ID

Parent process ID

Controlling terminal

EUID

Command

Signals

- User point-of-view: suspend, resume, kill processes

```
% ps axu | grep '[e]macs'
Andla63 24613  0.6  0.2   9604  4596 pts/1    S+   15:47   0:00 emacs -nw
% kill -HUP 24613
% ps axu | grep '[e]macs'
%
```

- Send arbitrary signals using kill command
- If typing directly to process's controlling tty
  - C-c sends INTR
  - C-z sends TSTP
  - C-\ sends QUIT

Privilege elevation

- Users gain extra privileges by changing EUID or starting processes with a different EUID than the current one

```
%  ps -H -eo s,pid,ppid,tty,user,cmd
S    PID   PPID TT       USER     CMD
S      1      0 ?        root     init [2]
S   2194      1 ?        root       /usr/sbin/sshd
S  24294   2194 ?        root         sshd: davby [priv]
S  24296  24294 ?        davby          sshd: davby@pts/0
S  24297  24296 pts/0    davby            -sh
R  24321  24297 pts/0    davby              ps -H -eo s,pid,ppid,tty,user,cmd
S  24312   2194 ?        root         sshd: davby [priv]
S  24314  24312 ?        davby          sshd: davby@pts/1
S  24315  24314 pts/1    davby            -sh
S  24319  24315 pts/1    root               passwd
```

sshd changed EUID from root to davby

passwd being run by davby with EUID root

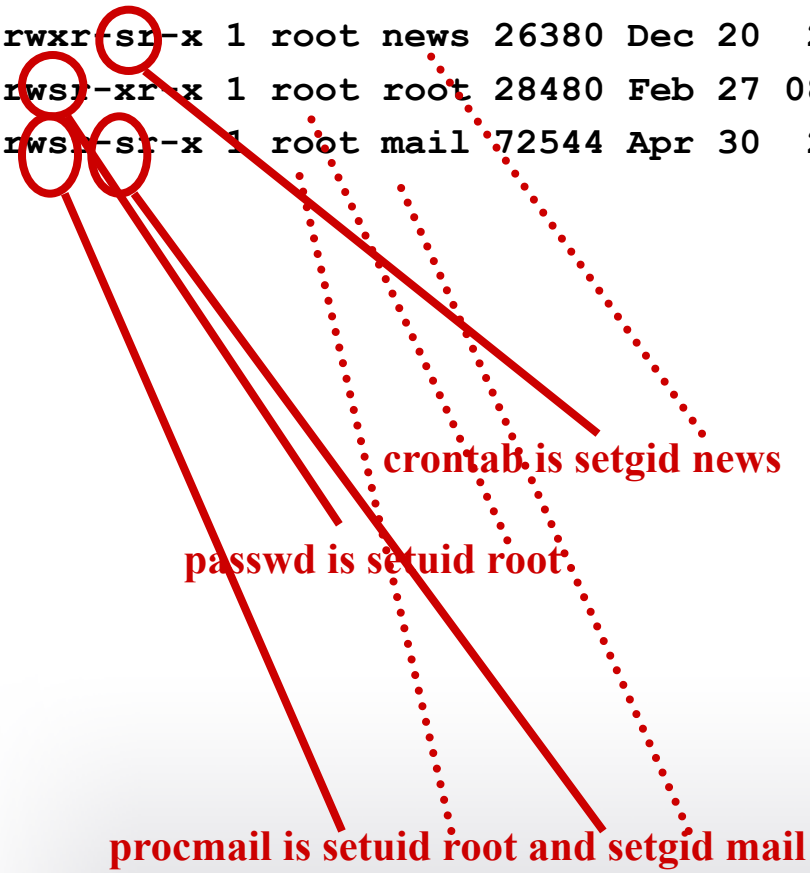
How does privilege elevation work?

- Programs can change their own EUID/EGID
  - The seteuid system call changes the EUID
  - The setegid system call changes the EGID
  - Very strict limitations on who can change to what

- Programs can have the setuid/setgid bits set
  - When setuid program started, process assumes file owner as EUID
  - When setgid program started, process assumes file group as EGID

Example of setuid/setgid programs

```
% ls -l passwd crontab mail
-rwxr-sr-x 1 root news 26380 Dec 20  2006 crontab
-rwsr-xr-x 1 root root 28480 Feb 27 08:53 passwd
-rwsr-sr-x 1 root mail 72544 Apr 30  2006 procmail
%
```



The shell

- When a user logs in, the login program starts a shell
- The shell accepts and interprets commands from the user
  - Handles I/O redirection, environment variables, etc
- Two kinds of commands: built-in and external
  - Built-in: affect the shell itself (e.g. cd) or are run often (e.g. echo)
  - External: most everything else
  - Also: programming structures (e.g. if-then-else)
- External commands are just files with the execute permission set that are in a directory listed in the PATH variable

System startup

- What happens when you start Linux
  1. The computer firmware (BIOS) loads the boot loader
  2. The boot loader loads and executes the operating system
  3. The operating system runs the /sbin/init program
  4. The /sbin/init program does what /etc/inittab says to do

```
% cat /etc/inittab
id:2:initdefault:
si::sysinit:/etc/init.d/rcS
l1:1:wait:/etc/init.d/rc 1
l2:2:wait:/etc/init.d/rc 2
l3:3:wait:/etc/init.d/rc 3
1:23:respawn:/sbin/getty 38400 tty1
2:23:respawn:/sbin/getty 38400 tty2
3:23:respawn:/sbin/getty 38400 tty3
%
```

Set default run level to 2

To do when initializing system

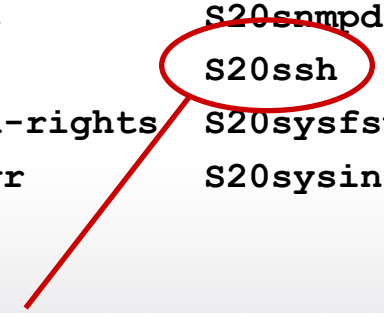
To do (once) when entering run level 2

To do when entering run level 2 or 3 (and when process terminates it is restarted

Typical (sysvinit) system startup

- /etc/init.d/rc script is run with run level as argument
  - Runs scripts in /etc/rc*N*.d directory
  - Scripts that start with K are run with argument **stop**
  - Scripts that start with S are run with argument **start**
- Note that /etc/rcS.d scripts are also run during boot

```
% ls /etc/rc2.d
S10sysklogd          S20cupsys            S20snmpd             S21exim
S11klogd             S20dbus              S20ssh               S23ntp
S18portmap           S20devtun-rights     S20sysfsutils        S89atd
S19autofs            S20dirmngr           S20sysinfo           S89cron
%
```



Run with argument start to start ssh service

System logging

- Linux (and unix) systems and software are often very chatty
  - Detailed messages often show up in system and application logs
  - By default, log files are stored in /var/log

```
% ls -F /var/log
account/        cfengine.log.0      fsck            mail.log.0      quagga/
acpid           cfengine.log.1.gz   kern.log        mail.log.1.gz   samba/
acpid.1.gz      daemon.log          kern.log.0      mail.warn       syslog
apache/         daemon.log.0        kern.log.1.gz   mail.warn.0     syslog.0
apache2/        daemon.log.1.gz     ksymoops/       mail.warn.1.gz  syslog.1.gz
aptitude        debug               lastlog         messages        user.log
aptitude.1.gz   debug.0             lpr.log         messages.0      user.log.0
auth.log        debug.1.gz          mail.err        messages.1.gz   user.log.1.gz
auth.log.0      dmesg               mail.err.0      mysql/          uucp.log
auth.log.1.gz   dpkg.log            mail.err.1.gz   nagios/         vtund/
boot            dpkg.log.1          mail.info       nessus/         wtmp
btmp            fai/                mail.info.0     news/           wtmp.1
btmp.1          faillog             mail.info.1.gz  ntpstats/       wtmp.report
cfengine.log    fontconfig.log      mail.log        pycentral.log
%
```

A Look at Linux: System logging

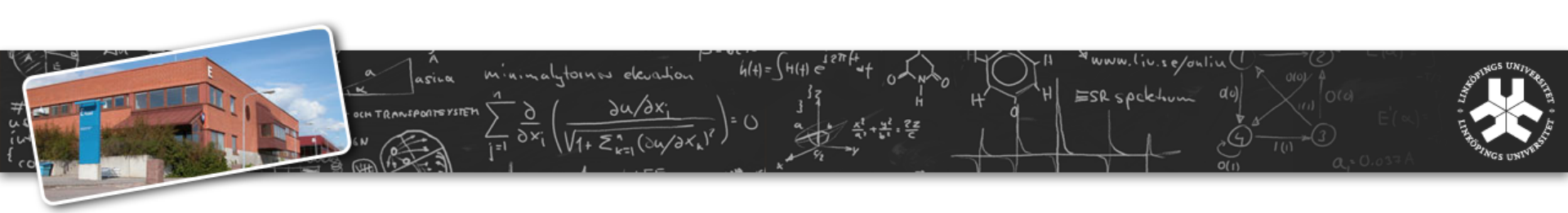
- Controlled by system logging service (syslogd)
  - Configured in /etc/syslog.conf
  - Controls what information goes where
  - Controls what **level** of information is logged

```
% head -2 /var/log/syslog.conf
auth,authpriv.*                 /var/log/auth.log
*.*;auth,authpriv.none          -/var/log/syslog
```

Log everything except auth messages to /var/log/syslog

System logging

```
% /etc/init.d/bind9 reload
% tail -12 /var/log/syslog
Aug 20 08:29:22 sysinst-gw postfix/cleanup[26219]: 3FE26748B9: message-
id=<20070820062922.3FE26748B9@sysinst-gw.sysinst.ida.liu.se>
Aug 20 08:29:22 sysinst-gw postfix/bounce[26258]: F24931F4B7: sender non-
delivery notification: 3FE26748B9
Aug 20 08:29:22 sysinst-gw postfix/local[26259]: 3FE26748B9: to=<nagios@sysinst-
gw.sysinst.ida.liu.se>, relay=local, delay=0.16, delays=0.05/0.04/0/0.07,
dsn=2.0.0, status=sent (delivered to mailbox)
Aug 20 09:00:54 sysinst-gw named[7673]: loading configuration from
'/etc/bind/named.conf'
Aug 20 09:00:54 sysinst-gw named[7673]: zone 189.236.130.in-addr.arpa/IN: loaded
serial 2007081500
Aug 20 09:00:54 sysinst-gw named[7673]: zone 189.236.130.in-addr.arpa/IN:
sending notifies (serial 2007081500)
Aug 20 09:00:54 sysinst-gw named[7673]: zone sysinst.ida.liu.se/IN: loaded
serial 2007081500
Aug 20 09:00:54 sysinst-gw named[7673]: zone sysinst.ida.liu.se/IN: sending
notifies (serial 2007081500)
Aug 20 09:00:54 sysinst-gw named[7673]: client 130.236.177.25#34505: transfer of
'189.236.130.in-addr.arpa/IN': AXFR-style IXFR started
Aug 20 09:00:54 sysinst-gw named[7673]: client 130.236.177.25#34505: transfer of
'189.236.130.in-addr.arpa/IN': AXFR-style IXFR ended
```



Linux Networking

LiU expanding reality

Review: Protocols

Data link layer
- Shared physical medium

Data link layer protocols
- Ethernet

Network layer
- Hosts on different networks

Network layer protocols
- Internet Protocol (IP)

Transport layer
- Between transport processes

Transport layer protocols
- TCP/UDP

Data link layer: Ethernet

Ethernet addressing

MAC address

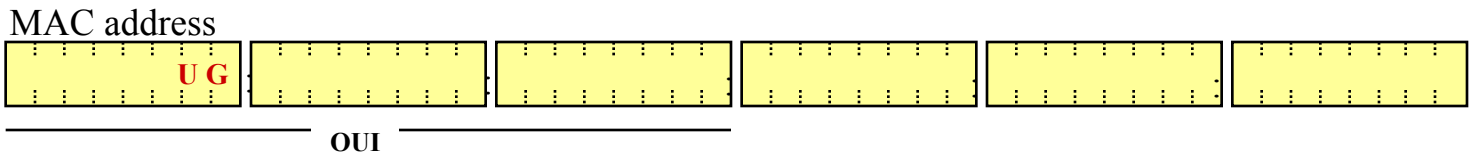
- Address on LAN (48 bits)
- Vendor ID (OUI)
- Group/individual bit
- Universal/local bit

Broadcast

- Sent to ff:ff:ff:ff:ff:ff

Multicast

- Sent to address with **G** set



To send an Ethernet frame to a recipient one must know the recipient's MAC address!

Ethernet in Linux

Logical interface

- Access with ifconfig/ip
- Configure with ifconfig/ip

Hardware interface

- Access with ethtool/mii-diag
- Configure with ethtool/mii-tool

```
% ip link show dev eth0
2: eth0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:0f:20:6b:76:f3 brd ff:ff:ff:ff:ff:ff

% ifconfig eth0
eth0      Link encap:Ethernet  HWaddr 00:0F:20:6B:76:F3
          inet6 addr: fe80::20f:20ff:fe6b:76f3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:183363968 errors:0 dropped:0 overruns:0 frame:0
          TX packets:139578378 errors:0 dropped:0 overruns:0 carrier:0
          RX bytes:2407195224 (2.2 GiB)  TX bytes:3814089863 (3.5 GiB)
```

LINKÖPINGS UNIVERSITET

Ethernet in Linux

Logical interface

- Access with ifconfig/ip
- Configure with ifconfig/ip

Hardware interface

- Access with ethtool/mii-diag
- Configure with ethtool/mii-tool

```
% mii-diag eth0
Basic registers of MII PHY #1:  1000 796d 0020 6162 05e1 cde1 000d 2001.
 The autonegotiated capability is 01e0.
The autonegotiated media type is 100baseTx-FD.
 Basic mode control register 0x1000: Auto-negotiation enabled.
 You have link beat, and everything is working OK.
 Your link partner advertised cde1: Flow-control 100baseTx-FD 100baseTx
10baseT-FD 10baseT, w/ 802.3X flow control.
   End of basic transceiver information.

% mii-tool eth0
eth0: negotiated 100baseTx-FD flow-control, link ok
```

LINKÖPINGS UNIVERSITET

Network layer: IPv4

Internet Protocol Family

IP is a family of protocols

- ICMP for control and error messages
- TCP for reliable data streams
- UDP for best-effort packet delivery
- GRE for tunneling other protocols
- ESP and AH for secure IP (IPSEC)
- SAT-MON for monitoring SATNET

You can have your own! Talk to IANA.

ICMP

IP Control Messages

- Error messages – ”Can’t reach that address”
- Control messages – ”Slow down, you’re sending too fast”
- Test messages – ”Tell me if you get this message”
- Autoconfiguration – ”Is there a router here?”

Some messages have sub-types

- Can’t reach destination because TTL was exceeded
- Can’t reach destination because the port does not exists
- Can’t reach destination because the network is unreachable

IPv4 addressing

IPv4 address

- Network address (**N** bits)
- Host address (**M** bits)
- **N** + **M** = 32 bits

CIDR notation

- *A.B.C.D/**N***

Broadcast

- 255.255.255.255 (undirected)

Multicast

- 224.0.0.0/4

IPv4 addressing

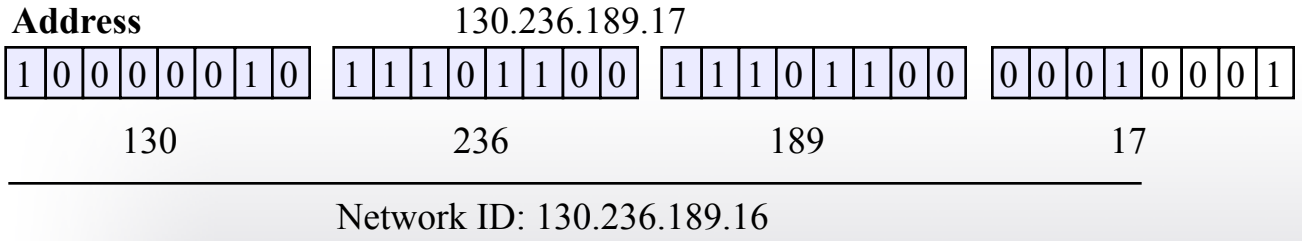
- Addresses are divided into classes
  - Class A has 8 bits network ID
  - Class B has 16 bits network ID
  - Class C has 24 bits network ID
  - Class D and E are special cases

- Subnetting divides large networks into several small ones
- Supernetting is used to combine small networks into larger ones

IPv4 addressing

- 32 bits divided into network ID and host ID
- **Netmask** determines what is what

- Given address and netmask, compute:
  - Network ID — netid = addr & netmask
  - Host ID — host = addr & (~netmask)
  - Broadcast — bcast = addr | (~netmask)
  - Address range — netid to bcast



Bitwise Operators

Multiply (And)

&	0	1
0	0	0
1	0	1

Addition (Or)

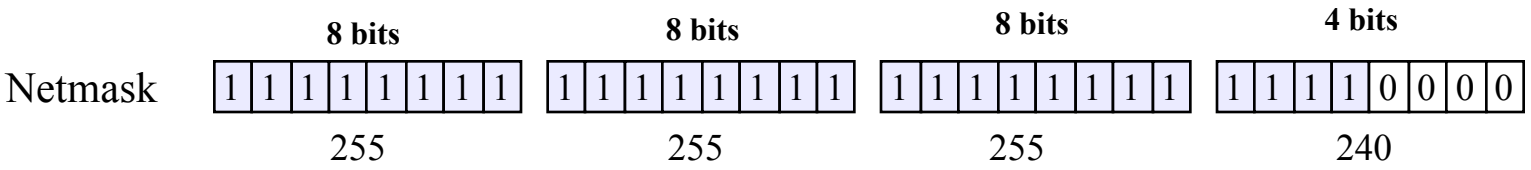
|	0	1
0	0	1
1	1	1

Negate (Not)

~	0	1
	1	0

130.236.189.17/28 **netmask**

130.236.189.16/**28** → 28 bit netmask



Bitwise Operators

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

~	0	1
	1	0

130.236.189.17/28 **network**

addr & mask

Address	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	1
Netmask	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
Network	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0
	130								236								189								16							

Bitwise Operators

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

~	0	1
	1	0

130.236.189.17/28 **broadcast**

addr | (~mask)

Address	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	1
~Netmask	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
Broadcast	1	0	0	0	0	0	1	0	1	1	1	0	1	1	0	0	1	1	1	0	1	1	0	0	0	0	0	1	1	1	1	1
	130								236								189								16							

Bitwise Operators

&	0	1
0	0	0
1	0	1

|	0	1
0	0	1
1	1	1

~	0	1
	1	0

130.236.189.17/28 **summary**

- CIDR block: 130.236.189.16/28
- Network: 130.236.189.16
- Lowest host: 130.236.189.17
- Highest host: 130.236.189.30
- Broadcast: 130.236.189.31

10.0.0.0/29 summary

- CIDR block: 10.0.0.0/29
- Network: ?
- Broadcast: ?
- Lowest host: ?
- Highest host: ?

Network ID netid = addr & netmask
Broadcast bcast = addr | (~netmask)

10.0.0.0/29 **summary**

- CIDR block: 10.0.0.0/29
- Network: 10.0.0.0
- Lowest host: 10.0.0.1
- Highest host: 10.0.0.6
- Broadcast: 10.0.0.7

192.168.12.163/29 **summary**

- CIDR block: 192.168.12.160
- Network: ?
- Broadcast: ?
- Lowest host: ?
- Highest host: ?

192.168.12.163/29 **summary**

- CIDR block: 192.168.12.160
- Network: 192.168.12.160
- Lowest host: 192.168.12.161
- Highest host: 192.168.12.166
- Broadcast: 192.168.12.167

IPv4 in Linux

- Addresses assigned to interfaces (e.g. eth0)
- Each interface can have multiple addresses

- Configure with **ifconfig** or ip

```
% ifconfig br0
br0       Link encap:Ethernet  HWaddr 00:0F:20:6B:76:F3
          inet addr:130.236.189.1  Bcast:130.236.189.63  Mask:255.255.255.192
          inet6 addr: fe80::20f:20ff:fe6b:76f3/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:183373446 errors:0 dropped:0 overruns:0 frame:0
          TX packets:139594398 errors:0 dropped:0 overruns:0 carrier:0
          RX bytes:3350149494 (3.1 GiB)  TX bytes:2985901093 (2.7 GiB)
```

IPv4 in Linux

- Addresses assigned to interfaces (e.g. eth0)
- Each interface can have multiple addresses

- Configure with ifconfig or **ip**

```
% ip addr show dev br0
7: br0: <BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue
    link/ether 00:0f:20:6b:76:f3 brd ff:ff:ff:ff:ff:ff
    inet 130.236.189.1/26 brd 130.236.189.63 scope global br0
    inet 10.17.1.1/24 scope global br0
    inet6 fe80::20f:20ff:fe6b:76f3/64 scope link
       valid_lft forever preferred_lft forever
```

Delivery of IP over Ethernet

Network cards have MAC-addresses, not IP addresses

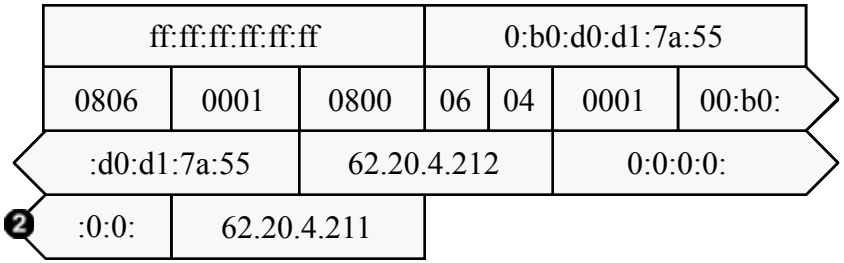
- MAC addresses are not assigned systematically so can't be used directly
- Translation from IP to MAC address needed

ARP – Address Resolution Protocol

- ARP Request = What MAC address does this IP address correspond to
- ARP Reply = This one

ff:ff:ff:ff:ff:ff	0:b0:d0:d1:7a:55	0806	0001	0800	06	04	0001	0:b0:d0:d1:7a:55	62.20.4.212	0:0:0:0:0:0	62.20.4.211
0:b0:d0:d1:7a:55	0:50:ba:7c:92:cc	0806	0001	0800	06	04	0002	0:50:ba:7c:92:cc	62.20.4.211	0:b0:d0:d1:7a:55	62.20.4.212

ARP Examples

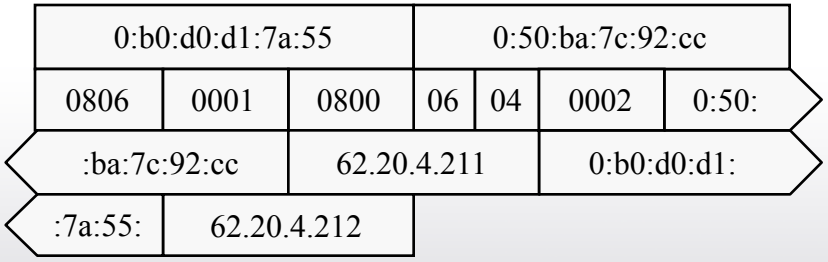


ARP Request

Hardware type	(2)	Sender MAC
Protocol	(2)	Sender protocol address
Hardware size	(1)	Target MAC
Protocol size	(1)	Target protocol address
Opcode	(2)	

```
tcpdump -ennqti eth0 \( arp or icmp \)
tcpdump: listening on eth0
0:80:c8:f8:4a:51 ff:ff:ff:ff:ff:ff 42: arp who-has 192.168.99.254 tell 192.168.99.35
0:80:c8:f8:5c:73 0:80:c8:f8:4a:51 60: arp reply 192.168.99.254 is-at 0:80:c8:f8:5c:73
```

ARP Reply



Sending an IP packet

1. Destination in routing table?
   - YES: Continue
   - NO: Signal no route to host
2. Is it directly connected?
   - YES: Recipient = destination address
   - NO: Recipient = gateway address
3. ARP for recipient
4. Got ARP reply?
   - YES: Send IP packet to Ethernet address in ARP reply
   - NO: Signal host unreachable

Linux routing table

Where do we send a given packet?

- To its final destination?
- Somewhere else?
- On which interface?

Determined by routing table

- Match destination against prefixes in kernel routing table
- Longest match wins
- No match? No route to host!

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
130.236.190.56  0.0.0.0         255.255.255.252 U     0      0        0 eth1
130.236.189.128 130.236.189.38  255.255.255.248 UG    2      0        0 eth0
130.236.189.0   0.0.0.0         255.255.255.192 U     0      0        0 eth0
10.17.219.0     10.17.1.219     255.255.255.0   UG    2      0        0 eth0
10.17.1.0       0.0.0.0         255.255.255.0   U     0      0        0 eth0
10.17.224.0     10.17.1.224     255.255.255.0   UG    2      0        0 eth0
0.0.0.0         130.236.190.57  0.0.0.0         UG    0      0        0 eth1
```

Linux routing

Sources for routes

- Connected interfaces
- Static routes
- Routing protocol (e.g. RIP)

Typically:

- Connected interfaces
- Static default route

Configure with route or ip

- **route –n** or **ip route list**
- **route add** or **ip route add**
- **route del** or **ip route del**

Routing with RIP

Review

- Distance-Vector protocol
- Distributed Bellman-Ford
  - Announce known prefixes with a cost to reach destination
  - For each prefix use neighbor with lowest cost to destination

Routing vs. Forwarding

- Routing: calculating paths
- Forwarding: sending packets received on another interface
- Separate functions!

Practicalities

- Announce which prefixes?
- Accept which announcements?
- Run on which interfaces?
- Which version to use?
- Use of authentication?
- What to install in kernel routing table (FIB)?

Routing with RIP

What prefixes to announce

- Redistribution of prefixes
- Sources of prefixes
  - Other RIP routers
  - Other routing protocols
  - Directly connected networks
  - Static routes
  - Kernel routing table
- Filter announcements?
  - distribute-list out

What announcements to accept

- What peers do we trust?
- What routes do we expect?
- Filter incoming prefixes
  - distribute-list in

LINKÖPINGS UNIVERSITET

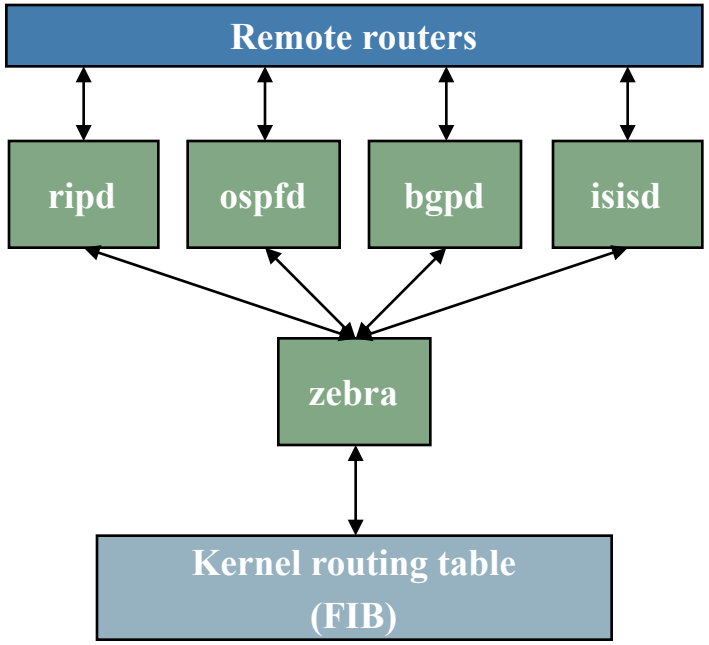
Routing with quagga

What it is

- Open source portable routing software suite
- Supports RIP, OSPF, BGP…

Working with quagga

- Command-line interface similar to Cisco IOS
- Access via telnet connection



Quagga (ripd) configuration example

```
interface br0
 ip rip send version 2
 ip rip receive version 2

router rip
 version 2
 default-information originate
 redistribute connected
 redistribute static
 network br0
 distribute-list prefix listen in br0
 distribute-list prefix announce out br0

ip prefix-list announce seq 5  permit 130.236.189.0/24 le 32
ip prefix-list announce seq 10 permit 10.0.0.0/8
ip prefix-list announce seq 25 permit 0.0.0.0/0
ip prefix-list announce seq 30 deny any

ip prefix-list listen seq 5   permit 10.0.0.0/8     le 32
ip prefix-list listen seq 25  permit 130.236.189.64/29
ip prefix-list listen seq 30  permit 130.236.189.72/29
[…]
ip prefix-list listen seq 999 deny any
```

Transport layer: TCP/UDP

TCP and UDP in Linux

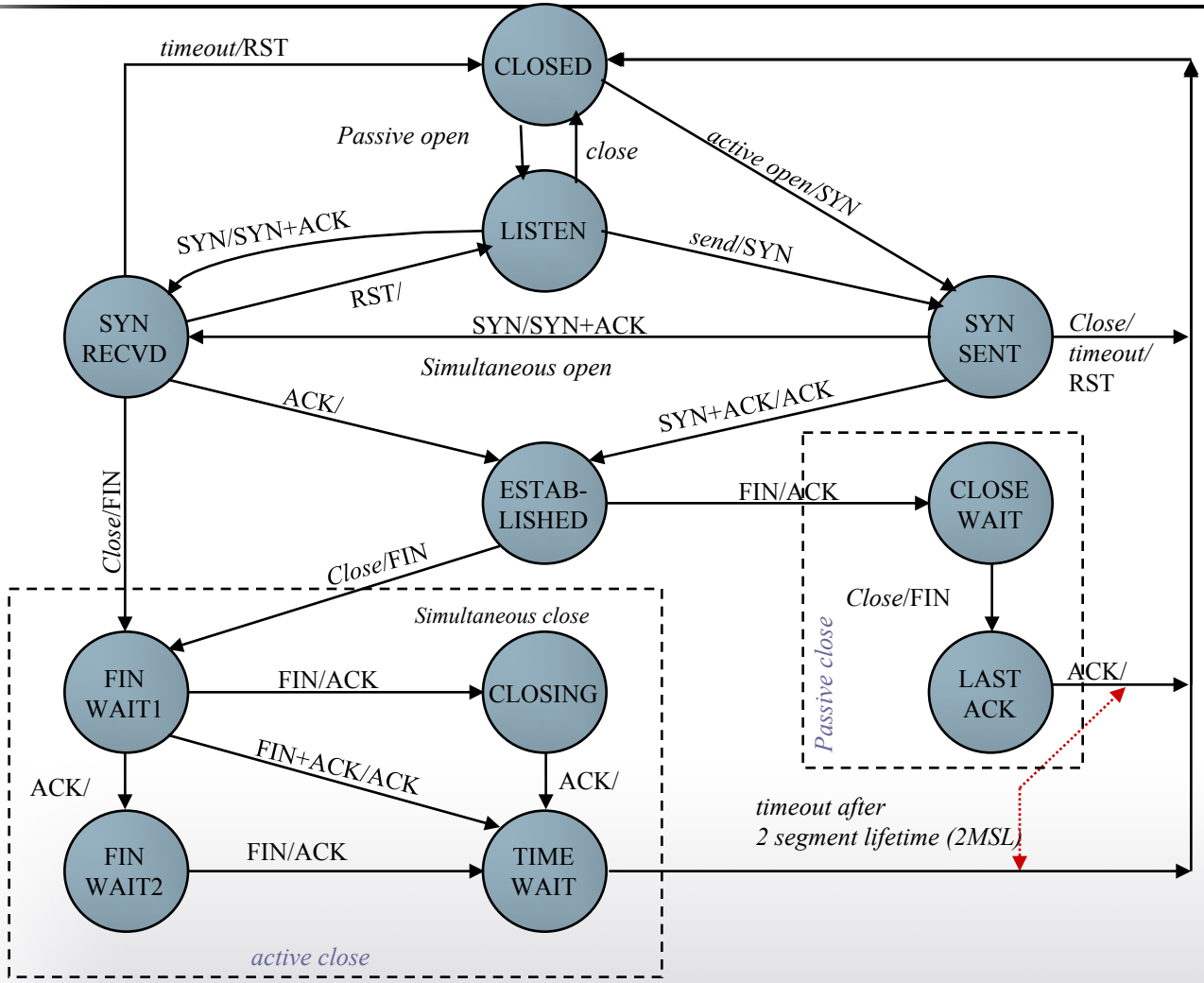
Review

- Port concept
- Socket concept
- TCP state diagram

Tools

- Tuning parameters
  - **/proc/sys/net/…**
- Examining sockets etc
  - **netstat**

TCP state diagram



```
% netstat -alp -A inet
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 *:login                 *:*                     LISTEN      22705/inetd
tcp        0      0 *:7937                  *:*                     LISTEN      15600/nsrexecd
tcp        0      0 *:shell                 *:*                     LISTEN      22705/inetd
tcp        0      0 *:7938                  *:*                     LISTEN      15599/nsrexecd
tcp        0      0 *:printer               *:*                     LISTEN      27352/lpd Waiting
tcp        0      0 *:sunrpc                *:*                     LISTEN      24838/portmap
tcp        0      0 *:www                   *:*                     LISTEN      27245/apache
tcp        0      0 *:629                   *:*                     LISTEN      25040/ypbind
tcp        0      0 *:nessus                *:*                     LISTEN      30517/nessusd: wait
tcp        0      0 localhost:953           *:*                     LISTEN      32675/named
tcp        0      0 *:smtp                  *:*                     LISTEN      28650/master
tcp        0      0 localhost:6010          *:*                     LISTEN      5891/83
tcp        0      0 localhost:6011          *:*                     LISTEN      9720/138
tcp        0      0 localhost:6012          *:*                     LISTEN      32607/202
tcp        0      0 *:732                   *:*                     LISTEN      26838/rpc.statd
tcp        0      1 sysinst-gw.ida:webcache 222.90.98.244:1350      FIN_WAIT1   -
tcp        0      1 sysinst-gw.ida:webcache h225n10c1o1049.br:13394 FIN_WAIT1   -
tcp        0      0 sysinst-gw.ida.liu.:www obel19.ida.liu.se:62599 FIN_WAIT2   -
udp        0      0 *:7938                  *:*                                 15599/nsrexecd
udp        0      0 *:902                   *:*                                 25040/ypbind
udp        0      0 *:route                 *:*                                 13790/ripd
udp        0      0 *:726                   *:*                                 26838/rpc.statd
udp        0      0 *:729                   *:*                                 26838/rpc.statd
udp        0      0 *:sunrpc                *:*                                 24838/portmap
udp        0      0 *:626                   *:*                                 25040/ypbind
udp        0      0 10.17.1.1:ntp           *:*                                 25800/ntpd
udp        0      0 sysinst-gw.sysinst.:ntp *:*                                 25800/ntpd
udp        0      0 sysinst-gw.ida.liu.:ntp *:*                                 25800/ntpd
udp        0      0 localhost:ntp           *:*                                 25800/ntpd
udp        0      0 *:ntp                   *:*                                 25800/ntpd
```

Applications

The Internet Super-Server

inetd

- Manages network for other services
- Other services started on demand
- Configuration file: inetd.conf

```
# Internal services
echo          stream   tcp      nowait   root     internal
echo          dgram    udp      wait     root     internal

# Shell, login, exec and talk are BSD protocols.
shell         stream   tcp      nowait   root     /usr/sbin/tcpd /usr/sbin/in.rshd
login         stream   tcp      nowait   root     /usr/sbin/tcpd /usr/sbin/in.rlogind

# RPC based services
rstatd/1-5    dgram    rpc/udp  wait     nobody   /usr/sbin/tcpd /usr/sbin/rpc.rstatd
rusersd/2-3   dgram    rpc/udp  wait     nobody   /usr/sbin/tcpd /usr/sbin/rpc.rusersd
```

LINKÖPINGS UNIVERSITET

TCP wrappers

Access control for TCP and UDP services

- Configuration: /etc/hosts.allow, /etc/hosts.deny
- Built-in support or through tcpd

```
ALL:                    UNKNOWN:            DENY
in.rshd:                130.236.189.1:      ALLOW
sshd:                   ALL:                ALLOW
statd mountd nfsd       @nfsclients:        ALLOW
ALL:                    ALL:                DENY
```

Remote access with ssh

Secure shell

- Encrypted channel
- Mutual authentication

Features

- X11 forwarding
- File transfer

… and lots more

Interactive shell: ssh *remote_username@hostname*
To copy files from host: scp *remote_username@hostname:path local_path*
To copy files to host: scp *local_path remote_username@hostname:path*

X11 forwarding

Run GUI programs on remote host with local display

Prerequisites:

- X11 forwarding enabled on client
- X11 forwarding enabled on server
- Server has xauth program installed

Necessary to run GUI programs (e.g. ethereal) on UMLs

Network troubleshooting

LiU expanding reality

IP connectivity problem

- Is the destination interface configuration correct and interface enabled?
  - Tools: ifconfig or ip on destination
  - No: fix it and enable interface
- Is the source interface configuration correct and interface enabled?
  - Tools: ifconfig or ip on source
  - No: fix it and enable interface
- Is there a route from source to destination and from destination to source?
  - Tools: traceroute on source and destination and see where the problem starts
  - No: troubleshoot routing (e.g. RIP failure)
- Do all gateways have forwarding enabled?
  - No: enable forwarding where it is disabled

LINKÖPINGS UNIVERSITET

Simple RIP failures

What interfaces to run on → We are not running on the right interfaces

What version to use → We are using the wrong version

What authentication to use → We are using the wrong authentication

What prefixes to announce → We are not announcing the right prefixes

- What is the source of the prefixes? Are we redistributing that source?
- Do we have filters on outgoing announcements? Are they accurate?

What prefixes to accept → We are not accepting the correct prefixes

- Do we have filters on incoming announcements? Are they accurate?
- Do we install routes in the kernel as expected?

Troubleshooting tools

traceroute

- To trace path of packets

ping

- To check connectivity

socat

- To set up a simple server
- To act as a client

wireshark/tcpdump

- Analyze network traffic

ip neigh/link/addr/route

- To check configuration

netstat

- Lots of host-related information

Next time: directory services

Directory services

- Why directory services
- What directory services are

Domain Name System

- How it works in theory
- How it works in practice
- How to set it up

Network Information Service

- How it works in theory
- How it works in practice
- How to set it up

LDAP

- Brief introduction