Device connection and startup

COMMUNICATION PROTOCOLS AND NETWORK SECURITY

CONTENT

- computer startup
- startup via network– bootp
- connection to the network

COMPUTER STRATUP

- when powered on the CPU sets the PC (program counter) on a predefined value
    - challenge: what value is the PC set to on an Intel processor? What value on powerpc? Which on arm?
- after that it starts executing commands
    - normal operation
- important: what is located in the memory location where the CPU starts it's work

BIOS

- *Basic I/O System – firmware*
- Consists of two sections:
  - code that starts executing on startup
  - I/O drivers

  - code uses the drivers to access external devices (hard drive, floppy drive, CD...) and loads a (special) program, that we call an operating system
  - with this the hardware is “booted” – has boots, *boot*

OPERATING SYSTEM– CLASSIC

- operating system (OS) is an interface between user programs and hardware, it is responsible for managing resources (I/O devices, files, processor time…)
- primarelly the OS used drivers from BIOS to work with I/O devices
- these had two disadvantages: i) they were not “friendly”; ii) they were not effective
- OS started to use it’s own drivers

LOADING OS – MODERN

- BIOS actually loads a program that it then executes
- it finds it on the first block of the I/O device – *master boot record, MBR*
- the loaded program doesn't have to be an OS, but can load the next (or one of the next) program that is an OS
  - option to load one of multiple OS
    - challenge: what is this program called? Find at least two examples.

LOADING A PROGRAM - DIFFERENT

- BIOS actually i) loads a program that it then - ii) executes.
- What if BIOS would load a program from a server on the network instead of a hard drive (different i) but we keep the second part the same).
- We need a definition of a way of communication between our computer and a server – we need a protocol.

LOADING A PROGRAM FROM THE NETWORK

- Advantages:
  - We don’t need a hard drive on the computer
  - OS is easily changed for all computers, because we only change it on the server
- Disadvantages:
  - vulnerability
  - slowness
  - security?

IT'S ALL IN THE NUMBERS

- www.fri.uni-lj.si = 212.235.188.25
- Service DNS converts between letter strings and numbers.
  - instead of DNS service we can use a mapping table in the file /etc/hosts
- How do we find the DNS service server?
- How does a DNS service server find other DNS servers?
  - it has to know their IP addresses
  - file /etc/namedb/named.root

IT'S ALL IN THE NUMBERS

- DNS service uses gate number 53.
- We have no service that would convert between the name DNS and 53
  - we have a mapping table in the file /etc/services
    - challenge: how is the DNS service really called in the table mentioned above?

```
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, most entries here have two entries
# even if the protocol doesn't support UDP operations.
#
# The latest IANA port assignments can be gotten from
#
#    http://www.iana.org/assignments/port-numbers
#
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 through 65535
#
# $FreeBSD: src/etc/services,v 1.89 2002/12/17 23:59:10 eric Exp $
#    From: @(#)services    5.8 (Berkeley) 5/9/91
#
# WELL KNOWN PORT NUMBERS
#
rtmp               1/ddp    #Routing Table Maintenance Protocol
tcpmux             1/udp     # TCP Port Service Multiplexer
tcpmux             1/tcp     # TCP Port Service Multiplexer
#                           Mark Lottor <MKL@nisc.sri.com>
nbp                2/ddp    #Name Binding Protocol
compressnet        2/udp     # Management Utility
compressnet        2/tcp     # Management Utility

                                                             ...
ftp-data          20/udp     # File Transfer [Default Data]
ftp-data          20/tcp     # File Transfer [Default Data]
ftp               21/udp     # File Transfer [Control]
ftp               21/tcp     # File Transfer [Control]
ssh               22/udp     # SSH Remote Login Protocol
ssh               22/tcp     # SSH Remote Login Protocol
telnet            23/udp     # Telnet
telnet            23/tcp     # Telnet
smtp              25/udp     # Simple Mail Transfer
smtp              25/tcp     # Simple Mail Transfer
                                                             ...
```

IT’S ALL IN THE NUMBERS

- DNS protocol uses UDP packages.
- In the head of a package we mark that it is an UDP package whith the number 17.
- We have no service that would convert between the name UDP and 17.
  - We have a mapping table in the file /etc/protocols
    - challenge: which protocol has the number 50 and what is it used for? What are the formats for all three etc files?

SO WHERE DO THE NUMBERS COME FROM

- world agreement about numbers
- the numbers are stored and advertised by IANA – *The Internet Assigned Numbers Authority*, www.iana.org
  - root DNS servers: www.iana.org/domains/root/db/arpa.html
  - *gates: www.iana.org/assignments/port-numbers*
    - challenge: write a program that produces automatically the file services from the data on the IANA server
  - *protocols: www.iana.org/protocols/*
    - challenge: what kind of data is on www.iana.org/domains/root/db/si.html?

LOADING OS FROM A NETWORK

- on startup the computer can know or doesn't know some of it's data:
  - name
  - IP address
  - …
- it certainly has to know the protocol that will enable the loading of the OS
  - like it has to know a way of reading data form a hard drive - driver
  - the protocol handler has to be short and informative

LOADING OS FROM A NETWORK– STEPS

- To load succesfully the computer has to:
  1. know how to find a server from which the OS will be loaded
  2. know how to set itself as advised/demanded by the server
  3. transfer the OS to itself
  4. install the OS and run it
- The last step is the same as with loading from a hard drive..
- Design decision: steps 1. and 2. in one protocol (bootp) and step 3. in a different protocol (eg. tftp)

PROTOCOL BOOTP

- Defined in RFC 951, BOOTSTRAP PROTOCOL (BOOTP)
  - *mandatory: find it on the internet and read it – literature!*
  - challenge: find the other RFC documents, that deal with bootp and see what they say.
- Conversation in steps between the client and the server: the client asks and the server answeres
- There can be more servers present and there can be more clients trying to load the OS at the same time

BOOTP – SOME DETAILS

- In the begining the client doesn’t know the IP address of the server so he sends (*broadcast*) on the 2. layer of the local network a desire for loading the OS
- The server assigns an IP address to the client (or doesn’t) and lets him know where is the client’s OS
  - not necessarily on the local network
- bootp is an application that uses connectionless mode – UDP protocol – on the transport layer
- This is where the conversation ends
    - Challenge: what is with security and trojans? Check RCPs.

BOOTP – OBLIKA PAKETA

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     op (1)    |   htype (1)   |   hlen (1)    |   hops (1)    |
+---------------+---------------+---------------+---------------+
|                            xid (4)                            |
+-------------------------------+-------------------------------+
|           secs (2)            |           flags (2)           |
+-------------------------------+-------------------------------+
|                          ciaddr  (4)                          |
+---------------------------------------------------------------+
|                          yiaddr  (4)                          |
+---------------------------------------------------------------+
|                          siaddr  (4)                          |
+---------------------------------------------------------------+
|                          giaddr  (4)                          |
+---------------------------------------------------------------+
|                                                               |
|                          chaddr  (16)                         |
|                                                               |
|                                                               |
+---------------------------------------------------------------+
|                                                               |
|                          sname   (64)                         |
+---------------------------------------------------------------+
|                                                               |
|                          file    (128)                        |
+---------------------------------------------------------------+
|                                                               |
|                          vend    (64)                         |
+---------------------------------------------------------------+
```

- op: zahteva ali odgovor
- htype: vrsta medija
- hlen: dolžina naslova
- chaddr: odjemalčev naslov plasti 2
- hops: število skokov
- xid: id zahteve
- secs: koliko časa je minilo od prvega pošiljanja
- flags: zastavice – samo razpošiljanje ali ne

BOOTP – PACKAGE FORM

```
0                   1                   2                   3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     op (1)    |   htype (1)   |   hlen (1)    |   hops (1)    |
+---------------+---------------+---------------+---------------+
|                            xid (4)                            |
+-------------------------------+-------------------------------+
|           secs (2)            |           flags (2)           |
+-------------------------------+-------------------------------+
|                          ciaddr  (4)                          |
+---------------------------------------------------------------+
|                          yiaddr  (4)                          |
+---------------------------------------------------------------+
|                          siaddr  (4)                          |
+---------------------------------------------------------------+
|                          giaddr  (4)                          |
+---------------------------------------------------------------+
|                                                               |
|                          chaddr  (16)                         |
|                                                               |
|                                                               |
+---------------------------------------------------------------+
|                                                               |
|                          sname   (64)                         |
+---------------------------------------------------------------+
|                                                               |
|                          file    (128)                        |
+---------------------------------------------------------------+
|                                                               |
|                          vend    (64)                         |
+---------------------------------------------------------------+
```

- ciaddr: client address
- yiaddr: set address
- siaddr: server address
- giaddr: gate address
- sname: name of the server with OS
- file: file containing OS
- *vend: possible extensions*
  - challange: record both packages on the network and comment on them

SOFTWARE

- on FreeBSD: bootpd and bootpgw
- configuration in /etc/bootptab

  - Challenge: find the handbook and just set the file then run the server and the transient server

```
client.test.net:\
        :ht=ether:\
        :ha=CCCCCCCCCCCC:\
        :sm=255.255.255.0:\
        :lg=192.168.1.5:\
        :ip=192.168.1.10:\
        :hn:\
        :bf=[/tftpboot/]OS:\
        :bs=auto:\
        :rp=/export/client/root/:\
        :vm=auto:
        :vm=rfc1048:
```

- Challenge: upper record uses a special notation of data - format. Is it used somewhere else? How is it defined exactly? How does an interface in C look when you read it? Is the upper form without errors?

PROTOCOL TFTP

- defined in RFC 1350, The TFTP Protocol (Trivial File Transfer Protocol)
  - *mandatory: find it on the internet and read it – literature!*
  - challenge: find the other RFC documents, that deal with tftp and see what they say.
- very simplified functionality of the ftp protocol – mainly preserved the option of data transfer
- no directory printout, authentification and encryption, allows very large packets, can’t load a file bigger than 1TB
  - Challenge: what is the syndrome of the wizard’s assistant (SAS)? Where and how does this concern tftp?

TFTP – SOME DETAILS

- In the beginning the client knows the IP address of the server because he gets it through the bootp protocol
- tftp is an application that uses connectionless mode – UDP protocol – on the transport layer
    - Challange: both bootp and tftp use UDP protocol – why?

TFTP – EXAMPLE OF CONVERSATION WHEN READING

1. client sends a request to read (RRQ)
2. ferver responds with a DATA package and data that was requested by the client; they are sent from a new gate and all the communication with the client must from now on run through this gate (NAT port?)
3. for every package the client responds with an ACK package, after that the server sends the next package (previous step) – if there is no confirmation in a certain amount of time, the server sends the package again
4. the specialty is the last package that is smaller than the maximum allowed size

TFTP – PACKAGE FORM

```
RRQ, WRQ:
 2 bytes     string    1 byte     string   1 byte
 ------------------------------------------------
| Opcode |  Filename   |   0  |    Mode     |   0  |
 ------------------------------------------------

DATA:

 2 bytes     2 bytes       n bytes
 ----------------------------------
| Opcode |   Block #  |   Data     |
 ----------------------------------

ACK:
 2 bytes     2 bytes
 ---------------------
| Opcode |   Block #  |
 ---------------------
```

- Opcode: request
- Filename 0: file name
- Mode 0: data format
- Block #: number of sent packages

  - Challange: record the packages on the network and comment on them

SOFTWARE

- on FreeBSD: tftpd
- no configuration file
- files that are served are in the directory /tftpboot
- example of the entire communication of loading an OS on *www.eventhelix.com/RealtimeMantra/Networking/**Bootp.pdf***
    - Challenge: find the handbook and install a tftp server with any files. tftp doesn’t allow strings like,,../’’ or ,,/../’’ in the file name – why?

CONNECTION TO A NETWORK

- Some computers have their own disc and load the OS on their own, but still want to connect to a netowork:
  - static IP number works only on stationary computers
  - mobile computers need a different number every time
  - providers want to provide more clients then they have IP addresses
- Protocol bootp sends data for setting the IP address and IP address of the port to the client in the first step
  - idea!! – lets use bootp protocol

BOOTP PROTOCOL FOR CONNECTION TO A NETWORK

- It is not a bad idea, but the problems:
  - aside from the IP address, we also need the port address, the DNS server address, proxy server address…
- Lets utilize/change the purpose of the *vend* field in bootp protocol

VEND EXTENSIONS

- defined in RFC 1497, BOOTP Vendor Information Extensions
  - *mandatory: find it on the internet and read it – literature!*
  - challenge: find the other RFC documents, that deal with this type of content and see what they say.
- first value is the "*magic cookie*" with a vlue of 99.130.83.99
- two types of fields (in lenght):
  - permanent: syllable 1: badge [data]
    - Subnet Mask Field (badge: 1, data: 4 syllables): 1.255.255.255.0
  - variable: syllable 1: badge, syllable 2: data length, other syllables: data
    - Gateway Field (badge: 3, data: N/4 addresses): 3.4.1.2.3.4
- badges 128-254: local extensions
  - Challenge: use bootp and add your own extension.

DHCP PROTOCOL

- there are versions for IPv4 and IPv6, first IPv4
- defined in RFC 2131, **Dynamic Host Configuration Protocol**
    - *mandatory: find it on the internet and read it – literature!*
    - challenge: find the other RFC documents, that deal with DHCP and see what they say.
- actually an extension of bootp protocol
    - renaming of *vend* field into *options* and it's extension – RFC 2132, *DHCP Options and BOOTP Vendor Extension*

DHCP – SOME DETAILS

- In the begining the client doesnt't know the IP address of the server
- DHCP is an application that uses connectionless mode – UDP protocol – on the transport layer
  - Challenge: how is security with DHCP protocol? If possible make an attack on the client.

DHCP – PROTOCOL CORE

- basic idea: the client gets an IP address to use for a limited amount of time
- possible requests:
  - DHCPDISCOVER: searching fo a server
  - DHCPOFFER: offer for the client
  - DHCPREQUEST: client confirms received settings; even the request for extending IP address usage
  - DHCPACK, DHCPNAK: server’s confirmation/denial to the client
  - DHCPDECLINE: client to server that the IP address is already in use
  - DHCPRELEASE: client returning address before expiration
  - DHCPINFORM: client only wants other data, he already has the address
- special badge in *options*: *DHCP message type*
    - Challenge: what is the value of this badge?

DHCP – LIFE CYCLE

```
 Server          Client          Server
(not selected)                    (selected)

      v               v               v
      |               |               |
      |     Begins initialization     |
      |               |               |
      | _____________/|\____________  |
      |/DHCPDISCOVER | DHCPDISCOVER  \|
      |               |               |
  Determines          |          Determines
 configuration        |         configuration
      |               |               |
      |\              |  ____________/ |
      | \________     | /DHCPOFFER     |
      | DHCPOFFER\    |/               |
      |           \   |               |
      |       Collects replies        |
      |             \|                |
      |     Selects configuration     |
      |               |               |
      v               v               v
```

```
      |               |               |
      | _____________/|\____________  |
      |/ DHCPREQUEST  |  DHCPREQUEST\ |
      |               |               |
      |               |     Commits
  configuration
      |               |               |
      |               | ____________/ |
      |               |/ DHCPACK      |
      |               |               |
      |    Initialization complete    |
      |               |               |
      .               .               .
      |               |               |
      |      Graceful shutdown        |
      |               |               |
      |               |\ ____________ |
      |               | DHCPRELEASE  \|
      |               |               |
      |               |     Discards lease
      |               |               |
      v               v               v
```

DHCP HAZZARDS

- DHCP doesn’t foresee an authentication
- Possible attacks:
  - unauthorized servers provide the wrong information
  - unauthorized clients gain access to resources that sould be restricted to them
  - emptying of resources by unauthorized clients
    - Challenge: do at least one of these attachs. What does RFC 3118 refer to and how does it work?

SOFTWARE

- on FreeBSD client *dhclient* with cofiguration file /etc/dhclient.conf
- see: www.freebsd.org/doc/handbook/network-dhcp.html
  - Challenge: configure a client and run it. What does right configuration actually mean?

```
send host-name "andare.fugue.com";
send dhcp-client-identifier 1:0:a0:24:ab:fb:9c;
send dhcp-lease-time 3600;
supersede domain-name "fugue.com home.vix.com";
prepend domain-name-servers 127.0.0.1;
request subnet-mask, broadcast-address, time-offset, routers,
      domain-name, domain-name-servers, host-name;
require subnet-mask, domain-name-servers;
timeout 60;
retry 60;
reboot 10;
select-timeout 5;
initial-interval 2;
script "/etc/dhclient-script";
media "-link0 -link1 -link2", "link0 link1";
reject 192.33.137.209;

alias {
  interface "ep0";
  fixed-address 192.5.5.213;
  option subnet-mask 255.255.255.255;
}

lease {
  interface "ep0";
  fixed-address 192.33.137.200;
  medium "link0 link1";
  option host-name "andare.swiftmedia.com";
  option subnet-mask 255.255.255.0;
  option broadcast-address 192.33.137.255;
  option routers 192.33.137.250;
  option domain-name-servers 127.0.0.1;
  renew 2 2000/1/12 00:00:01;
  rebind 2 2000/1/12 00:00:01;
  expire 2 2000/1/12 00:00:01;
}
```

SOFTWARE

- on FreeBSD server *net/isc-dhcp31-server* with configuration file /usr/local/etc/dhcpd.conf
  - Challenge: configure a server and run it. What does the program dhcp_probe do – install and run it.

```
option domain-name "example.com”;
option domain-name-servers 192.168.4.100;
option subnet-mask 255.255.255.0;

default-lease-time 3600;
max-lease-time 86400;
ddns-update-style none;

subnet 192.168.4.0 netmask 255.255.255.0 {
  range 192.168.4.129 192.168.4.254;
  option routers 192.168.4.1;
}

host mailhost {
  hardware ethernet 02:03:04:05:06:07;
  fixed-address mailhost.example.com;
}
```

DHCPV6 PROTOCOL

- defined in RFC 3315, *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*
  - *mandatory: find it on the internet and read it – literature!*
  - challenge: find the other RFC documents, that deal with DHCP and see what they say.
- completely different protocol for IPv6
- two ways of configuring a computer:
  - *stateless* where a computer can set itself; and
  - *statefull* where a computer is set using other devices

DHCPV6 – SOME DETAILS

- In the begining the client doesnt't know the IP address of the server
- DHCP is an application that uses connectionless mode – UDP protocol – on the transport layer

DHCPV6 – PROTOCOL CORE

- possible requests (*msg-type*):
  - SOLICIT: request for settings
  - ADVERTISE: advertising an address
  - REQUEST: request for settings parameters
  - CONFIRM: confirming if an address given to a client is still valid
  - RENEW: request to renew
  - REBIND: request to maintain
  - REPLY: reply to a client
  - RELEASE: release an address
  - DECLINE: reject an assigned address
  - RECONFIGURE: server is telling the client to renew the settings
  - INFORMATION-REQUEST: request for settings withoit an IP address
  - RELAY-FORW: forwarding
  - RELAY-REPL: confirmation for the forwarder that contains the reply to the client
    - Challenge: how does forwarding of requests work?

DHCPV6 – MESSAGE FORM

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg-type   |               transaction-id                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
.                            options                            .
.                           (variable)                          .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

normal message

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    msg-type   |   hop-count   |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                         link-address                          |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                                                               |
|                         peer-address                          |
|                                                               |
|                               +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-|
|                               |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
.                                                               .
.            options (variable number and length)   ....        .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

forwarded message

- izziv:what kind of options do we have? Where did the fields from IPv4 go? What is this DUID?

SOFTWARE

- on FreeBSD client, server and forwarder *dhcp6* with configuration file /usr/local/etc/dhcp6{c,s}.conf

  - Challenge: configure a client and run it. What does right configuration actually mean?

```
option domain-name-servers 2001:db8::35;
interface fxp0 { address-pool pool1 3600; };
pool pool1 { range 2001:db8:1:2::1000 to 2001:db8:1:2::2000 ; };
```

nastavitvena datoteka strežnika

ZAKLJUČEK

- we have seen how a computer can boot of a network and
- how it can connect to a network

- Next time: network management