Application Protocols for Administration

BootP, TFTP, DHCP, DNS, SNMP

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
  - Introduction
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol
- **SNMP**

BOOTP (RFC 951, 1542, 2132)

- **BOOTP was developed for bootstrapping**
  - allows diskless clients (and other network components without non-volatile memory) to load configuration parameters and operating system code from a central server
- **BOOTP is based on a client-server principle and uses UDP communication**
  - client-side: well known port 68
  - server-side: well known port 67

BOOTP-Principles

- **BOOTP-client sends request to the BOOTP-server**
  - using 255.255.255.255 as destination address (limited broadcast)
  - and 0.0.0.0 as source address (UDP relies upon IP!)
- **server uses the client's MAC-address for a database lookup to determine the IP-address of the client**
- **server replies with the desired boot information; again a limited broadcast is used as destination address**
  - alternatively, an ARP-cache entry without utilizing the ARP-request/response-procedure at the server-side
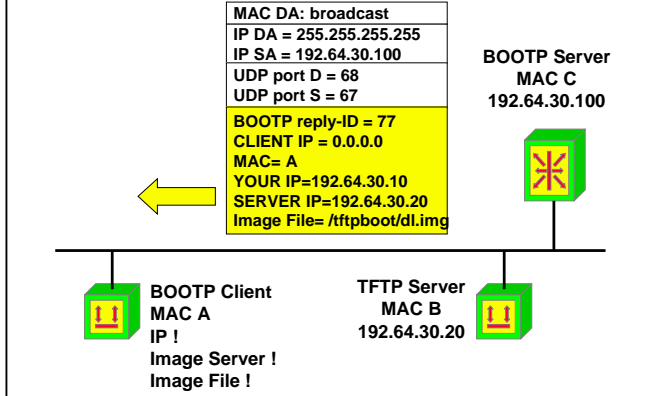- **end of the BOOTP-procedure**
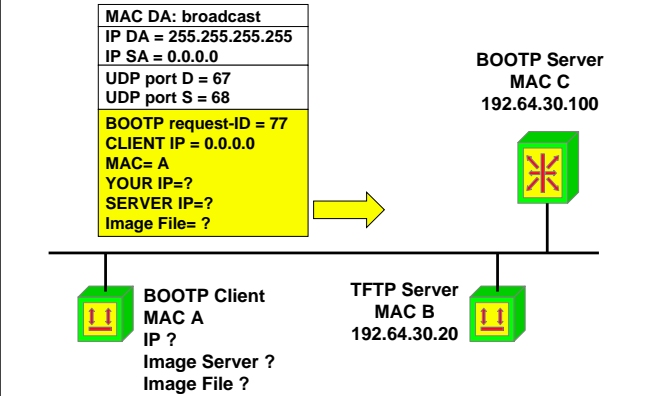
BOOTP-Principles

- **basically boot information contains**
  - the IP-address of an IP-host which provides appropriate bootfiles (image + configuration)
  - and also the filename of these bootfiles
- **client uses this information to load bootfiles via TFTP**
- **limited broadcast is restricted on a single LAN; in order to reach also BOOT-P servers of other subnets**
  - router or other computer-system must be designed and configured appropriately to act as BOOTP-relay agent
    - configuration of an IP-helper-address (Cisco specific) to forward specific UDP broadcasts

Bootstrap 1

| MAC DA: broadcast |
| IP DA = 255.255.255.255 |
| IP SA = 0.0.0.0 |
| UDP port D = 67 |
| UDP port S = 68 |
| BOOTP request-ID = 77 |
| CLIENT IP = 0.0.0.0 |
| MAC= A |
| YOUR IP=? |
| SERVER IP=? |
| Image File= ? |

BOOTP Server
MAC C
192.64.30.100

BOOTP Client
MAC A
IP ?
Image Server ?
Image File ?

TFTP Server
MAC B
192.64.30.20

Bootstrap 2

| MAC DA: broadcast |
| IP DA = 255.255.255.255 |
| IP SA = 192.64.30.100 |
| UDP port D = 68 |
| UDP port S = 67 |
| BOOTP reply-ID = 77 |
| CLIENT IP = 0.0.0.0 |
| MAC= A |
| YOUR IP=192.64.30.10 |
| SERVER IP=192.64.30.20 |
| Image File= /tftpboot/dl.img |

BOOTP Server
MAC C
192.64.30.100

BOOTP Client
MAC A
IP !
Image Server !
Image File !

TFTP Server
MAC B
192.64.30.20

BOOTP-Message Format

	1	2	3	4 bytes
OP	HTYPE	HLEN		HOPS
TRANSACTION ID				
SECONDS		Reserved		
CLIENT IP ADDRESS				
YOUR IP ADDRESS				
SERVER IP ADDRESS				
ROUTER IP ADDRESS				
CLIENT HARDWARE ADDRESS (16 Octets)				
SERVER HOST NAME (64 Octets)				
BOOTFILENAME (128 Octets)				
VENDOR SPECIFIC AREA (64 Octets)				

BOOTP Message Fields

- **OP (Operation Code):**
  - 1 ... Boot Request, 2 ... Boot Reply
- **HTYPE (Hardware Type):**
  - network type (1 for Ethernet); numbers similar to ARP
- **HLEN:**
  - length of the hardware address (e.g. 6 for ethernet)
- **HOPS:**
  - number of hops; optionally used by routers
  - initialized with zero by the client
  - increased by one if a BOOTP-server forwards the request to other servers (bootstrap over multiple servers)
    - BOOTP Relay Agent activated

BOOTP Message Fields

- **TRANSACTION ID:**
  - identification mark of related request-reply BOOTP-datagram's (random number)
- **SECONDS:**
  - seconds elapsed since client started trying to boot
- **CLIENT IP ADDRESS:**
  - client IP-address; filled in by client in boot-request if known
- **YOUR IP ADDRESS:**
  - client IP-address; filled in by server if client doesn't know its own address (if the client IP-address in the request was 0.0.0.0)

BOOTP Message Fields

- **SERVER IP ADDRESS:**
  - server IP-address where image is stored; returned in boot-reply by the server
- **ROUTER IP ADDRESS:**
  - server is part of another subnet
  - IP address of the BOOTP relay agent
- **CLIENT HARDWARE ADDRESS:**
  - MAC-address of client
  - advantage of BOOTP over RARP: server-application may rely upon UDP/IP protocol-stack to extract MAC-address; no need for layer 2 access

BOOTP Message Fields

- **SERVER HOST NAME:**
  - optional server host name
- **BOOTFILENAME:**
  - contains directory path and filename of the bootfile
- **VENDOR SPECIFIC AREA:**
  - may optionally contain vendor information of the BOOTP server
  - according to RFC 2132 it is also possible to mention the subnet-mask (opt. 1), hostname, domainname, IP-address of the DNS-server (opt. 6), IP-address of the Default Gateway (Router opt. 3), etc.
  - Here DHCP comes in (opt. 53) !!!

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
  - Introduction
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol
- **SNMP**

DHCP (Dynamic Host Configuration Protocol)

- **DHCP (RFC 2131, 3396) build on two components:**
  - Protocol to deliver host specific configuration from a server to a client
  - Mechanism to allocate temporary or permanent host addresses
- **Temporary address allocation**
  - DHCP server receives a request from a DHCP client and picks out an IP address from a configurable address pool and offers this address to the client
  - the client can use this leased address for a period of time
  - after the end of this lease, the address must again be requested by the client or is returned to the address pool

DHCP Configurable Parameters

- **DHCP eliminates**
  - a number of configuration tasks and problems associated with a manual TCP/IP configuration
- **A DHCP client can asks for:**
  - IP address
  - Subnet Mask
  - DNS Server, NetBIOS-Name Server
  - default TTL, Source Routing Option, MTU
  - max. Fragment Size, Broadcast Address
  - List of Default Gateways + Preferences, Static Routes
  - ARP Cache Timeout, TCP Keepalives
  - Ethernet Encapsulation
  - Path MTU Discovery (RFC1191)
  - Router Discovery (RFC 1256)

DHCP Address Allocation

- **DHCP provides three mechanisms for address allocation:**
  - Automatic:
    - DHCP assigns a permanent address to a host
  - Dynamic:
    - DHCP gives the client an address for a limited time period (LEASE). Automatic reuse of not active addresses is possible.
  - Manual:
    - Host addresses are still manually configured by a Network Administrator but other parameters configured by DHCP

BootP/DHCP Message Format

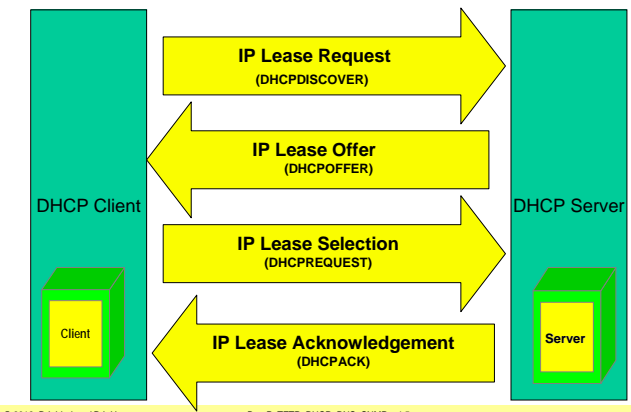| code	HWtype	length	hops
Transaction ID			
seconds		Flags field	
Client IP address			
Your IP address			
Server IP address			
Router IP address			
Client HW Address 64 byte			
Server host name 64 byte			
Boot file name 128 byte			
Options variable length (at least 312 byte) (here are the DHCP messages !!!)			

DHCP Message Types in Option Field

– DHCPDISCOVER (opt. 53 / type 1):
  • Client broadcast to find DHCP server(s)

– DHCPOFFER (opt. 53 / type 2):
  • Response to a DHCPDISCOVER, offering an IP address and other parameters

– DHCPREQUEST (opt. 53 / type 3):
  • Message form the client to the server to get the following:
    – Requests the parameters offered by one server, declines all other offers
    – Verification of a previously allocated address after a system reboot, or network change
    – Request the extension of the lease time

DHCP Message Types (cont.)

– DHCPACK (opt. 53 / type 5):
  • Acknowledgement from server to client, with IP address and parameters

– DHCPNACK (opt. 53 / type 6):
  • Negative ACK from server to client
  • Clients lease expired or requested IP address is invalid

– DHCPDECLINE (opt. 53 / type 4):
  • Message from a client to a server indicating an error

– DHCPRELEASE (opt. 53 / type 7):
  • Message from a client to a server canceling remainder of a lease and relinquishing network address

– DHCPINFORM (opt. 53 / type 8):
  • Message from a client that has already an externally configured IP address, asking for more local configuration parameters

DHCP Operation

DHCP Client — IP Lease Request (DHCPDISCOVER) → DHCP Server
DHCP Client ← IP Lease Offer (DHCPOFFER) — DHCP Server
DHCP Client — IP Lease Selection (DHCPREQUEST) → DHCP Server
DHCP Client ← IP Lease Acknowledgement (DHCPACK) — DHCP Server
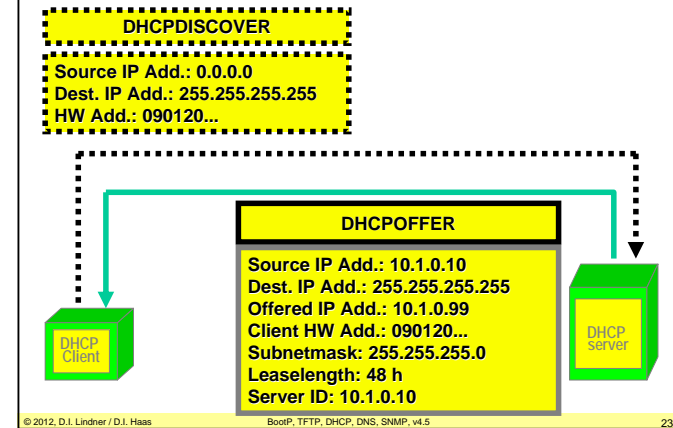
IP Lease Request

- **When the clients starts up**
  - sends a broadcast to all DHCP servers
  - since the client has no IP configuration, it uses 0.0.0.0 as source- and 255.255.255.255 destination address
  - this request is send in a DHCPDISCOVER message, together with the clients HW- address and the computer name
- **The IP lease is used when:**
  - TCP/IP initializes for the first time on this client
  - the client requests a specific IP address and is denied
  - the client previously leased an IP address, but released the lease and requires a new lease

IP Lease Offer

- **All DHCP servers**
  - that receive the DHCPDISCOVER message and has valid IP information for this client
  - send out a DHCPOFFER (broadcast) that includes:
    - clients HW address
    - an offered IP address (in the Your IP Address Field)
    - subnet Mask (in the Options Field)
    - length of the lease (time value)
    - server ID or the IP address of the offering DHCP server

IP Lease and Offer

DHCPDISCOVER

Source IP Add.: 0.0.0.0
Dest. IP Add.: 255.255.255.255
HW Add.: 090120...

DHCPOFFER

Source IP Add.: 10.1.0.10
Dest. IP Add.: 255.255.255.255
Offered IP Add.: 10.1.0.99
Client HW Add.: 090120...
Subnetmask: 255.255.255.0
Leaselength: 48 h
Server ID: 10.1.0.10

DHCP Client

DHCP server

IP Lease Selection

- **When a client receives**
  - an offer from at least one DHCP server
  - he sends a DHCPREQUEST (broadcast) out to the network, to tell all the other DHCP server that no more offers are accepted
  - the DHCPREQUEST message includes the server ID (IP address) of the server whose offer was accepted by the client

IP Lease ACK / NACK

- **In case of success a DHCPACK is send by the server whose offer was accepted**
  - DHCPACK contains a valid lease for an IP address and possible other configuration parameters
  - after the client receives the DHCPACK, TCP/IP is completely initialized and the client enters the BOUND state
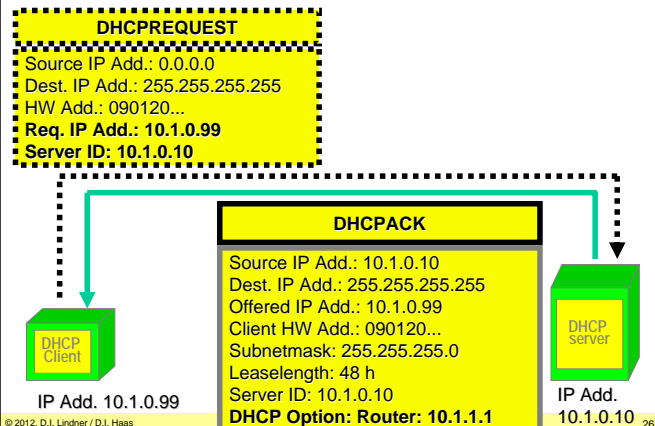 - if the client is bound, it can use TCP/IP as a base for communication

- **In case of no success a DHCPNACK will be send:**
  - e.g. Client tries to lease the previous IP address, but this address is no longer available
  - e.g. Client's IP address is invalid, the client may have been moved to an other subnet

IP Lease Selection and ACK

DHCPREQUEST

Source IP Add.: 0.0.0.0
Dest. IP Add.: 255.255.255.255
HW Add.: 090120...
Req. IP Add.: 10.1.0.99
Server ID: 10.1.0.10

DHCPACK

Source IP Add.: 10.1.0.10
Dest. IP Add.: 255.255.255.255
Offered IP Add.: 10.1.0.99
Client HW Add.: 090120...
Subnetmask: 255.255.255.0
Leaselength: 48 h
Server ID: 10.1.0.10
DHCP Option: Router: 10.1.1.1

DHCP Client — IP Add. 10.1.0.99

DHCP server — IP Add. 10.1.0.10

DHCP Lease Renew

- **When the server sends his DHCPACK**
  - containing the IP address for the client, the beginning of the lease period is registered
- **The lease time is located**
  - in the DHCPACK message in addition to two other time values T1 and T2
- **T1 (Renewal Attempt) and T2 (Sub Renewal Attempt)**
  - are configured at the DHCP server.
  - T1= 0,5 x lease time, T2= 0,875 x lease time.

DHCP Lease Renew (cont.)

- **T1 and T2 start their function**
  - when the client is bound.
  - the client attempt to renew the lease when 0,5 of the lease time has expired
  - the client enters the RENEWING state and sends an DHCPREQUEST (unicast) to the server forcing him to extend the lease
  - if the server accepts, an DHCPACK, containing a new lease time and the default values of T1/T2 are sent back to the client

DHCP Lease Renew (cont.)

- **If the lease could not be renewed**
  - at the 0,5 interval, the client will contact any other DHCP server DHCPREQUEST (using broadcast) when 0,875 of the lease time has expired to renew the clients lease time
- **The client enters the REBINDING state**
  - when 0,875 of the lease time has expired
- **Any DHCP server can answer to this request**
  - with an DHCPACK renewing the lease, or with an DHCPNACK, forcing the client to reinitialize and to get a new lease for an other IP address
- **Generally:**
  - if a lease expires or an DHCPNACK is received, the client must stop using it's present IP address
  - this will result in TCP/IP communication stop for this client
  - the client must request a new lease using DHCPDISCOVER

DHCP over Subnets

- **Note that:**
  - DHCP is related to BOOTP
  - DHCP messages are broadcast based (L2-Ethernet-Broadcast and IP-Limited Broadcast), so they can not be forwarded by a router
  - in case of connecting DHCP clients to their servers over a number of subnets which are connected with routers, it is unavoidable to enable the broadcast forwarding on this router = BOOTP relay agent
  - most of the routers support this specific function
  - on a router, broadcast forwarding is turned OFF by default

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
  - Introduction
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol
- **SNMP**

Trivial File Transfer Protocol (RFC 1350)

- **TFTP is suited for applications**
  - that do not require the rather complex procedures of FTP
  - or cannot provide enough resources (RAM, ROM)
- **typical utilization:**
  - boot helper for diskless clients
  - enables software-update for network components like bridges, router, SNMP agents of hubs, etc.
- **code size of TFTP is very small and easy to implement**
  - fits well in Bootstrap-ROMs of workstations

TFTP

- **TFTP has been designed to provide**
  - *simplest* transmission of files
  - client-server communication principle
- **TFTP do NOT support**
  - functions for reading directory contents
  - access verification mechanisms
- **TFTP is an unsecured protocol,**
  - there is no authentication (no username or password)

TFTP

- **TFTP uses UDP**
  - well know port server 69, datagram size = 512 bytes
- **TFTP is responsible for error recovery**
  - based on IdleRQ-protocol (stop and wait)
- **IdleRQ-principle**
  - every TFTP-datagram is marked with a sequence number
  - these datagram's are confirmed by short ACK-datagram's in the opposite direction
  - after receiving an acknowledge the next datagram is send
  - error recovery by retransmission after a timer expires
    - timer is activated after sending data or acknowledges
    - TFTP uses adaptive timeout (e.g. exponential backoff algorithm)

TFTP Message Formats

2 octet opcode	n octets	1 octet	n octets	1 octet
READ REQUEST (1)	FILENAME	0	MODE	0

Type 1

2 octet opcode	n octets	1 octet	n octets	1 octet
WRITE REQUEST (2)	FILENAME	0	MODE	0

Type 2

- **Type 1 and 2 initialize the TFTP transfer by specifying the direction of the transaction of the file**
- **MODE determines the type of data (NETASCII, BINARY, MAIL)**
- **FILENAME and MODE can have arbitrary length and consist of ASCII characters; the last character is always NULL**

TFTP Message Formats

2 octet opcode	2 octet seq.#	up to 512 octets
DATA (3)	BLOCK#	INFORMATION OCTETS

Type 3

2 octet opcode	2 octets
ACK (2)	BLOCK#

Type 4

- **Type 3 is used for the data transfer**
- **BLOCK# is the sequence number (starting with 1, increased by one for every block)**
- **last block has length < 512 (EOF mark)**
- **Type 4 is used to acknowledge every DATA message explicitly**
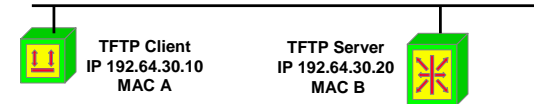
TFTP Protocol Description

– a TFTP transfer begins with the request to read or write a file

– if the server accepts the request, a connection is opened and datagram's, with a fixed size of 512 bytes, are sent
  • all datagram's are numbered consecutively beginning with 1,2,3,...and so on
  • each datagram must be acknowledged

– the connection will terminate if a datagram arrives with less than 512 bytes, or in case of errors
  • retransmission will start in case of datagram loss

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   37

TFTP (1)

```
MAC DA: B
MAC SA: A

IP DA: 192.64.30.20
IP SA: 192.64.30.10

UDP Port D: 69
UDP Port S: 1244

TFTP: Read
Filename: /tftpboot/dl.img
Mode: Bin
```
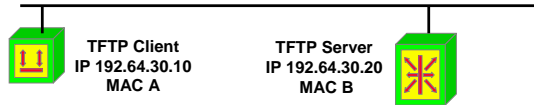
TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   38
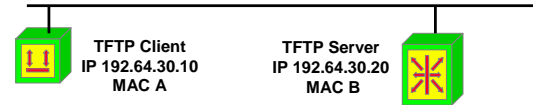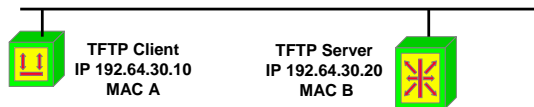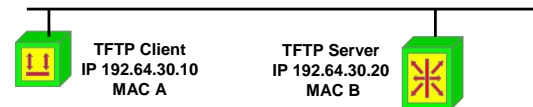
TFTP (2)

```
MAC DA: A
MAC SA: B

IP DA: 192.64.30.10
IP SA: 192.64.30.20

UDP Port D: 1244
UDP Port S: 2030

TFTP: Data
Block#: 1
Info: /tftpboot/dl.img
Octet: 0-511
```

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   39

TFTP (3)

```
MAC DA: B
MAC SA: A

IP DA: 192.64.30.20
IP SA: 192.64.30.10

UDP Port D: 2030
UDP Port S: 1244

TFTP: Ack
Block#: 1
```

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   40
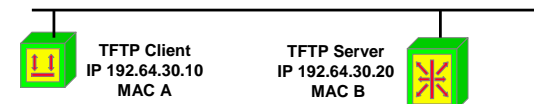
TFTP (4)

MAC DA: A MAC SA: B
IP DA: 192.64.30.10 IP SA: 192.64.30.20
UDP Port D: 1244 UDP Port S: 2030
TFTP: Data **Block#: 2** **Info: /tftpboot/dl.img** **Octet: 512-1023**

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

TFTP (1) Write

| |
|---|
| MAC DA: B
MAC SA: A |
| IP DA: 192.64.30.20
IP SA: 192.64.30.10 |
| UDP Port D: 69
UDP Port S: 1244 |
| **TFTP: Write**
Filename: /tftpboot/back.img
Mode: Bin |

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

© 2012, D.I. Lindner / D.I. Haas

Page Appendix 1 - 21

TFTP (2) Write

MAC DA: A MAC SA: B
IP DA: 192.64.30.10 IP SA: 192.64.30.20
UDP Port D: 1244 UDP Port S: 2030
TFTP: ACK **Block#: 0**

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

TFTP (3) Write

MAC DA: A MAC SA: B
IP DA: 192.64.30.10 IP SA: 192.64.30.20
UDP Port D: 1244 UDP Port S: 2030
TFTP: Data **Block#: 1** **Info: /tftpboot/back.img** **Octet: 0-511**

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B

© 2012, D.I. Lindner / D.I. Haas

Page Appendix 1 - 22

TFTP (4) Write

```
MAC DA: B
MAC SA: A

IP DA: 192.64.30.20
IP SA: 192.64.30.10

UDP Port D: 2030
UDP Port S: 1244

TFTP: Ack
Block#: 1
```

TFTP Client
IP 192.64.30.10
MAC A

TFTP Server
IP 192.64.30.20
MAC B
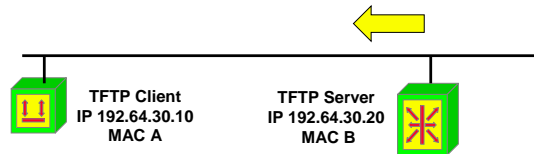
TFTP User Interface

- **Basic TFTP commands:**
  - **Connect** <host>: Destination host
  - **Mode** <ascii/binary>
  - **Get** <remote file> [<local filename>]: Retrieve a file
  - **Put** <remote file> [<local filename>]: Send a file
  - **Verbose** <on/off>: shows status information during the transfer.
  - **Quit**: Exit TFTP
- **TFTP data modes:**
  - **NETASCII**: 8 bit character set.
  - **OCTET**: Binary or 8 bit raw
  - **MAIL**: Allows sending a mail to a user, rather than transferring to a file.

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
  - Introduction
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol
- **SNMP**

History (1)

- **even in the early days of the Internet, hosts have been also identified by names**
  - e.g. /etc/hosts.txt file on UNIX systems
- **all names have been maintained**
  - by the Network Information Centre (NIC) in the single file "hosts.txt "
  - this file has been FTPed by all hosts in the Internet
- **this approach does not scale well**
  - additional drawbacks:
    - modifying hostnames on a local network became visible to the Internet only after a long (distribution-) delay
    - name space was not hierarchical organized

History (2)

- **rapid growth of the Internet demanded for a better, *more general* naming system**
- **in 1984 the <u>Domain Name System (DNS)</u> has been introduced by P. Mockapetris (IAB)**
  - RFC 1034: Domain Names - Concepts and Facilities (Internet Std. 13)
  - RFC 1035: Domain Names - Implementation and Specification (Internet Std. 13)
  - RFC 1713: Tools for DNS debugging (Informational)
  - RFC 1032: Domain Administrators Guide
  - RFC 1033: Domain Administrators Operations Guide
- **the future:**
  - RFC 2136:Dynamic Updates in DNS (Proposed Standard)
  - RFC 3007: Secure DNS Dynamic Update (Proposed Standard)

Mnemonic Approach

- **<u>Problem</u>: the 32-bit IP address-format encodes 2^32 single addresses (4 294 967 296)**
  - theoretically (!) – many of them have been wasted
  - how to build an effective <u>directory</u> for such a huge number of hosts?
- **<u>Solution:</u>**
  - <u>hierarchy</u> of simple, mnemonic names: *Domain Names*
    - e.g. instead of remembering all IP addresses from 216.32.74.50 to 216.32.74.55, it is sufficient to know "www.yahoo.com"
- **Why is the Internet so convenient to use?**
  - Domain names can be *guessed* and *bookmarked*
    - and of course search engines do the rest...

## What Basically Does DNS ?

- **DNS "replaces" the IP address of hosts to a <u>human readable</u> format**
  - DNS enables a mapping between names and addresses
  - often called "hostname resolution"
  - due to its size DNS is a world-wide *distributed* database

- **DNS assigns hosts to a <u>tree-like directory hierarchy</u>**
  - each part of the hierarchy is called a "domain", each hierarchy level is assigned a label, called "domain name"
  - the Domain Name Tree <u>does NOT</u> reflect the physical network structure !!!
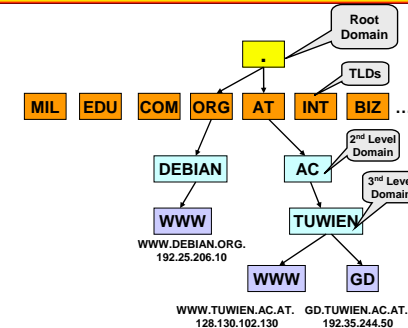
Tree of Names

Compare this DNS tree with a file directory tree of a common Operating Systems where **C:\at\ac\tuwien\www\ip_address.txt** is used to specify the location of the file ip_address.txt on the harddisk

Name Servers - DNS Resolver

- **the DNS tree is realized by**
  - <u>Name Servers</u>
- **each Name Server take cares**
  - for a subset of the DNS tree
  - so called "zones"
- **the physical location of name server**
  - has nothing to do with the DNS tree
- **if an IP host wants to resolve a symbolic name**
  - resolver software acting as DNS client will ask a DNS name server using the DNS protocol
  - IP address of name server either manually configured or known through DHCP or explicitly specified by the user
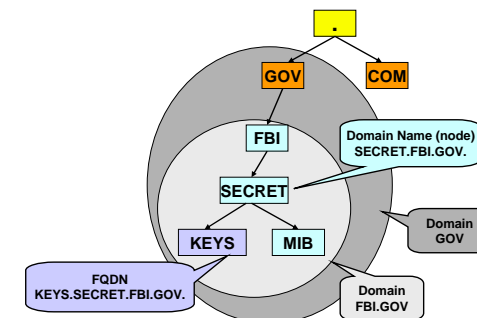
Conventions (1)

- Terminology**: a "Domain" ...**
  - is a complete <u>subtree</u>
    - everything under a particular point in the tree
  - <u>relates to the naming structure itself, not the way things are distributed</u>

- Terminology**: a "Domain Name" ...**
  - is the name of a <u>node</u> in the tree (domain, host, ...)
  - consists of all concatenated labels from the root to the current domain, listed from right to left, separated by dots
    - max 255 characters

Conventions (2)

- Terminology**: a "Label" ...**
  - is a component of the domain name
  - need only be unique at a particular point in the tree
    - that is, both "name.y.z" and "name.x.y.z" are allowed
    - max 63 characters
  - DNS is not case sensitive !
    - "www.nic.org" is the same as "WWW.NIC.ORG"
  - Due to SMTP restrictions, domain names may contain only characters of {a-z, A-Z, 0-9, "-"}
- Terminology**: a "Fully Qualified Domain Name"**
  - FQDN
  - concatenation of all labels of including trailing dot **". "**

Example for Terminology

Conventions (3)

- **hosts with _multiple_ network addresses can be assigned a _single_ domain name**
  - e.g. routers, servers with several network interfaces, ...
- **hosts with a _single_ IP address can be assigned _multiple_ domain names**
  - e.g. to differentiate several services: **www**.x.y.z, **ftp**.x.y.z, **mail**.x.y.z, ...

The Root Domain

- **the root of the DNS tree is denoted as a single dot "."**
  - each domain name without this root-dot is only a relative domain name
    - although, most applications do not follow this rule
    - but essential in BIND configuration files (master files)
  - otherwise it is a Fully Qualified Domain Name (FQDN) which exactly identifies a single host from all hosts in the world
- **the root is implemented by _several_ root-servers**
  - name server at the highest hierarchy level
- **below the root, a domain may be called top-level, second-level, third-level etc...**

Top Level Domains (RFC1591)

- **inside US: "generic domains"**
  - **com** - Commercial
  - **edu** - Educational
  - **org** - Non Profit Organizations (NPOs)
  - **net** - Networking providers
  - **mil** - US military
  - **gov** - US goverment
  - **int** - International organizations
- **outside US: two letter country code**
  - defined in ISO-3166
  - examples: **uk** (United Kingdom), **fr** (France), **us** (United States), **de** (Germany), **at** (Austria), **ax** (Antarctica)
  - Note: country code does not reflect real location !

Domain Name Registration

- **domain name registration is completely independent from IP address assignment**
- **where domain names can be registered:**
  - USA: InterNIC (www.internic.net)
  - Europe: RIPE (www.ripe.net)
  - Asia: APNIC (www.apnic.net)

IN-ADDR.ARPA (1)

- **special feature: the *in-addr.arpa domain***
  - used to support <u>gateway location</u>
  - enables <u>reverse lookups</u>: given an IP-address the associated hostname can be found
- **without the IN-ADDR.ARPA domain**
  - an *exhaustive search* in the domain space would be necessary to find any desired hostname
- **commonly used by**
  - WWW servers to log its users in a file
  - IRC servers that want to restrict their service inside a certain domain
    - E.g. a closed chat/discussion group exclusive for domains under IEEE.ORG

IN-ADDR.ARPA (2)

- **the domain in-addr.arpa is structured according to the IP address**
  - this special domain begins at "IN-ADDR.ARPA"
  - its substructure follows the Internet addressing structure

- **each domain name has up to 4 additional labels**
  - each label represents one octet of the IP address
    - expressed as character string for its decimal value ("0" - "255")
    - the reverse host/domain names are organized on byte boundaries

  - Note: labels are attached to the suffix in reverse order
    - e.g. data for internet address 216.32.74.50 is found at 50.74.32.216.IN-ADDR.ARPA
    - hosts have all four labels specified

IN-ADDR.ARPA (3)
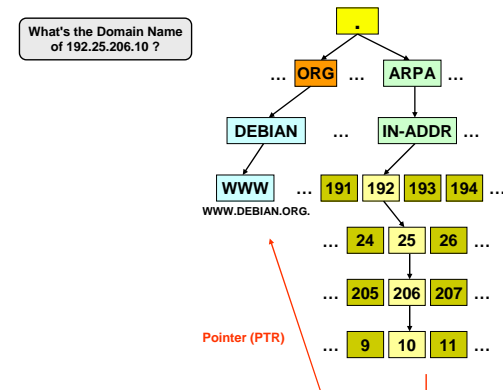
Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
  - Introduction
  - <u>Bind and DNS Servers</u>
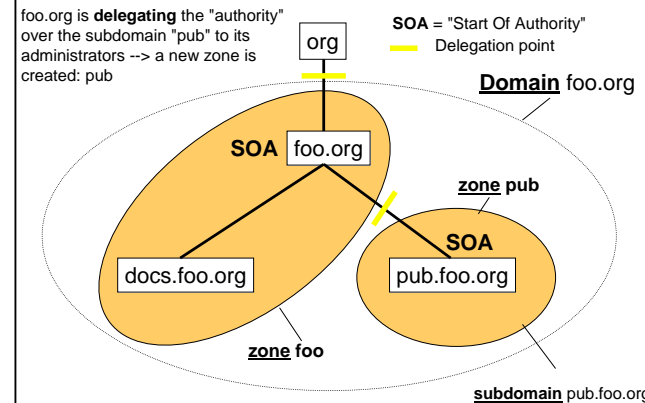 - Resource Records
  - DNS Protocol
- **SNMP**

What is BIND ?

- **the Berkeley Internet Name Domain (BIND)**
  – implemented by Paul Vixie as an Internet name server for BSD-derived systems
  – most widely used name server on the Internet
  – version numbers: 4 (old but still used), 8 and 9 (new)
- **BIND consists of**
  – a <u>name server</u> called <u>named</u> ("d" stands for "daemon")
  – a <u>resolver library</u> for client applications
    - The "resolver" is a collection of functions like gethostbyname(2) and gethostbyaddr(2)
- **technically, BIND and DNS deal primarily with <u>zones</u>**
  – a zone is a part of the domain space

What is a Zone ?

- **a <u>zone</u> is a "point of delegation"**
  – contains all names from this point downwards the domain-tree except those which are delegated to other zones (to other name servers)
  – a zone can span over a whole domain or just be part of it
- ***in other words:* a zone is a pruned domain !**
  – pruning occurs when zones are delegated
  – <u>zones relate to the way the database is partitioned and distributed</u>
- **a name server is *<u>authoritative</u>* over a domain**
  – if he keeps a *<u>master file</u>* (zone file) with information of that domain

Zones and SOA

foo.org is **delegating** the "authority" over the subdomain "pub" to its administrators --> a new zone is created: pub

**SOA** = "Start Of Authority"
▬ Delegation point

Delegation and Name Servers

Root Server responsible for root domain delegates authority for building symbols org. to NS below which holds the master file for zone org
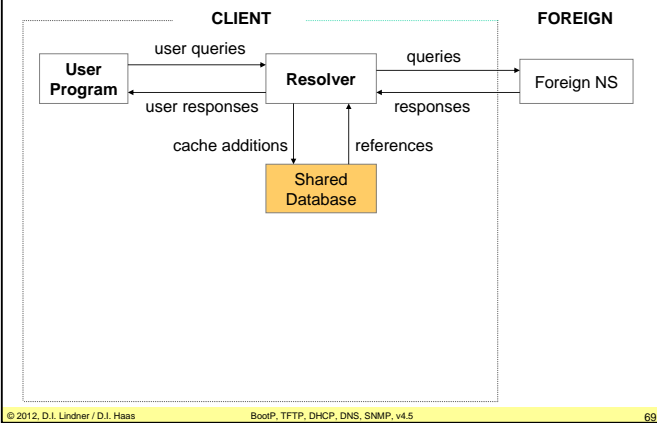
NS responsible for domain org delegates authority for building symbols foo.org. to NS below which holds the master file for zone foo

NS responsible for domain foo.org delegates authority for building symbols pub.foo.org. to NS below which holds the master file for zone pub
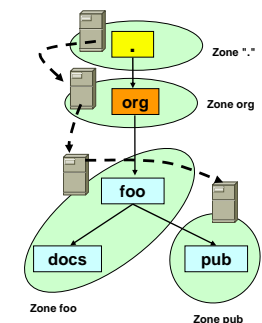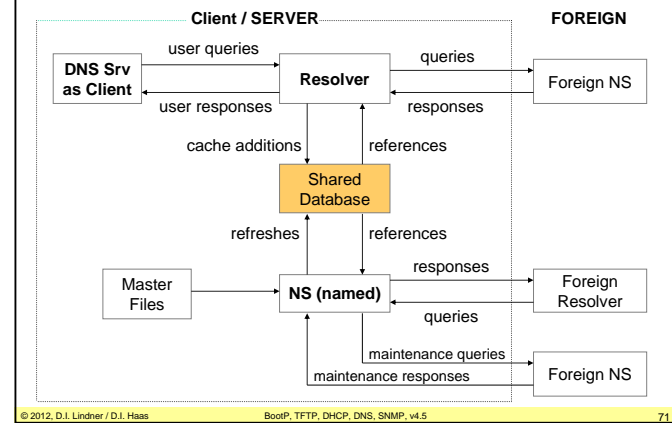
BIND Principles (Client)

BIND Principles (Server)
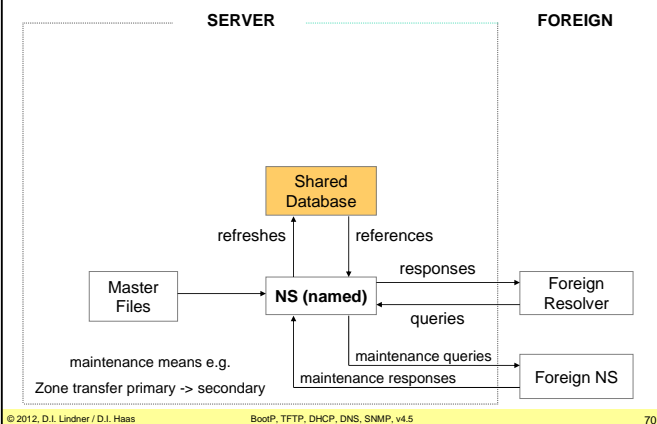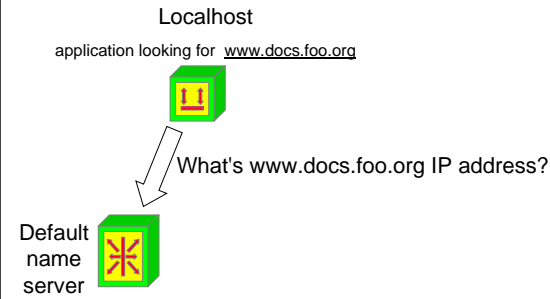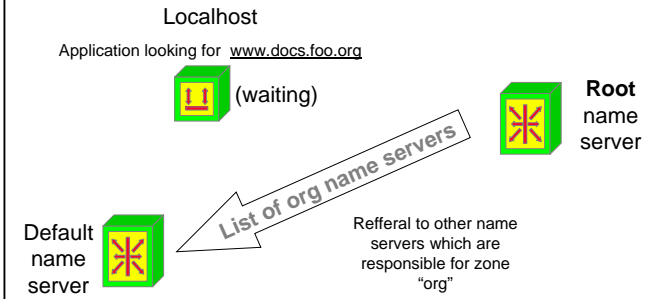
BIND Principles (Server complete)

BIND Principles

- **applications running on a client use the resolver to send *name resolution queries* to a name server**
  - each client-host requires a preconfigured IP address of one (or several) *default name server(s)*
- **a name server responses to this query after retrieving the requested data either**
  - by <u>recursive</u> queries -> the job is forwarded
  - by <u>iterative</u> queries -> the NS replies with a list of authoritative NSs to be queried by the client
  - from its <u>cache</u> -> the NS supplies non-authoritative data
  - or by its own zone data contained in its master file:
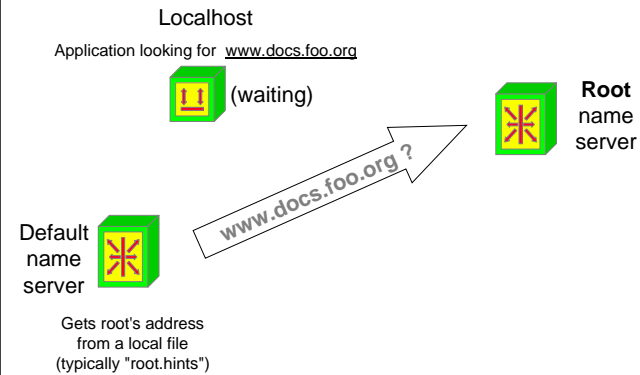   - the NS is authoritative for that requested zone

Recursive Query (1)

Localhost

application looking for www.docs.foo.org

What's www.docs.foo.org IP address?

Default
name
server

Iterative Queries (2)

Localhost

Application looking for www.docs.foo.org

(waiting)

Root name server

www.docs.foo.org ?

Default
name
server

Gets root's address
from a local file
(typically "root.hints")

Iterative Queries (3)

Localhost

Application looking for www.docs.foo.org

(waiting)

Root name server

List of org name servers

Default
name
server

Refferal to other name
servers which are
responsible for zone
"org"

Iterative Queries (4)

Localhost

Application looking for www.docs.foo.org

(waiting)

org name server

www.docs.foo.org ?

Default
name
server

Iterative Queries (5)

Localhost

Application looking for www.docs.foo.org

[icon] (waiting)

Default
name
server

org
name
server

List of foo name servers

Refferal to other name
servers which are
responsible for zone
"foo.org"

Iterative Queries (6)

Localhost

Application looking for www.docs.foo.org

[icon] (waiting)

Default
name
server

www.docs.foo.org ?

foo.org
name
server

Has authority of the
zone **foo.org** which
also includes
docs.foo.org

Iterative Queries (7)

Localhost

Application looking for www.docs.foo.org

[icon]

IP address of www.docs.foo.org

Default
name
server

IP address of www.docs.foo.org

foo.org
name
server

Now that response is
cached locally

Has authority of the
zone **foo.org** which
also includes
docs.foo.org

Root Hints

- **Since queries normally start at the root name servers, a name server has to know these address(es)**
- **This information is usually maintained in a "root.hints" file (currently 13 servers specified)**
- **The local name server queries these server one after each other until one of them replies**
- **The replying root server attaches an actual list of available root servers**
  - From this moment on, the local NS exclusively uses this list only

Master Files

- **The DNS database is made up of Master Files**
  - Contains mapping of symbols to IP addresses for the responsible part of the name tree (zone)
- **Each Master File is associated with a <u>domain</u>**
  - This domain is called the "origin" or the "owner"
    - Used symbol for this domain: "@"
    - Specified in the boot-up file with the *cache* or *primary* options
  - Within a master file other domain- and hostnames can be specified relative to the origin
  - Otherwise they are FQDNs and are specified with a trailing dot
    - Like "ws.docs.foo.org.**"**

Types of Name Servers (1)

- **Primary (master) name server**
  - Each zone must have exactly one primary NS
  - Has <u>own master files</u> about a zone ("authoritative")

- **Secondary (master) name servers**
  - Query a primary name server periodically for a "zone transfer", that is, each secondary name server stores a backup of the primary name server's master files
  - Have also authority over the zone of the primary
  - Are used for redundancy and load balancing purposes
  - Secondary NS are suggested by RFC 1035
  - Nowadays prefered term is slave name server

Types of Name Servers (2)

- **Caching only server**
  - **All** servers do cache -- but this one is not authoritative for any zone (except localhost)
  - Queries other servers who *have* authority
  - Data is kept in cache until the data expires (aging mechanism, TTL)
- **DNS client (or "remote server")**
  - Has no running named at all !!!
  - "remote server" is a confusing term; it means that this server *contacts* a remote server for hostname resolution
  - Technically it is no server at all !!!
  - Favour the term "DNS client", avoid "remote server"

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **<u>DNS</u>**
  - Introduction
  - Bind and DNS Servers
  - <u>Resource Records</u>
  - DNS Protocol
- **SNMP**

Resource Records

- **All data contained in a master file is split up into Resource Records (RRs)**
- **All DNS operations are formulated in terms of RRs (RFC 1035)**
  - Each query is answered with a copy of matching RRs !!!
  - RRs are the smallest unit of information available through DNS
- **RR format**
  - 5 fields, separated by spaces or tabs:

[DOMAIN] [TTL] [CLASS] TYPE RDATA

Resource Record Components (1)

- **DOMAIN**
  - Domain name to which the entry applies
  - If no domain name is given the RR applies to the domain of the previous RR
- **TTL**
  - Time To Live = time in seconds this RR is valid after it has been retrieved from the server
  - 8 digit decimal number
- **CLASS**
  - Address class: IN for Internet, CH for CHAOS, HS for Hesiod (MIT)
  - 2 bytes

Resource Record Components (2)

- **TYPE**
  - Describes the type of the RR
  - e.g. SOA, A, NS, PTR (see below)
  - 2 bytes
- **RDATA**
  - Contains the actual data of the RR
  - Its format depends on the type of the RR (see below)
  - Variable length

RR Type Values

Type	Value	Meaning
A	1	Host address
NS	2	Authoritative name server
CNAME	5	Canonical name for an alias
SOA	6	Marks the start of a zone of authority
WKS	11	Well known service description
PTR	12	Domain name pointer
HINFO	13	Host information
MINFO	14	Mailbox or mail list information
MX	15	Mail exchange
TXT	16	Text strings

Types of Resource Records (1)

- **SOA - Start of Authority RR**
  - Marks the beginning of a zone; typically seen as the first record in a master file
  - All records following the SOA RR contain authoritative information for the domain
  - Every master file included by a primary statement must contain an SOA record for this zone

  SOA RDATA fields:

  - MNAME (or "ORIGIN")
    - Canonical hostname of the primary server for this domain
    - Usually given as absolute name (FQDN)

SOA RDATA fields cont.

- RNAME (or "CONTACT")
  - E-Mail address of an administrator responsible for this domain
  - The "@" character must be replaced with a dot
- SERIAL
  - Version number of the zone file
  - Is used by secondary name servers to recognize changes of the zone file
  - Should be incremented when changes are applied to the zone
- REFRESH
  - 32 bit time interval in seconds that a secondary name server should wait between checking this SOA record
- RETRY
  - 32 bit time value in seconds that should elapse before a failed refresh should be retried by a secondary name server

SOA RDATA fields cont.

- EXPIRE
  - 32 bit time value in seconds after which this zone data should not be regarded as authoritative any longer
  - After this time a server may discard all zone data
  - Normally a very large period, e.g. 42 days
- MINIMUM
  - Minimum 32 bit TTL value in seconds
  - Is a lower bound on the TTL field for all RRs in a zone
  - Only used for normal responses (not zone transfers)

Types of Resource Records (2)

- **A - Address RR**
  - Most important -- associates an IP address with one canonical hostname
  - RDATA consists of a 32-bit IP address
  - Each host can have exactly as many A records as it has network interfaces
- **CNAME - Canonical Name RR**
  - Is like an alias or a symbolic link to a canonical hostname
  - RDATA contains the canonical name
- **PTR - POINTER RR**
  - Points to another location in the domain name space
  - RDATA contains the domain name

Types of Resource Records (3)

- **NS - Name Server RR**
  - Points to authoritative name server(s) of the given domain and to authoritative name server(s) of a subordinate zone
  - RDATA contains the FQDN of that name server
  - Using NS records a name server knows which name servers are responsible for a domain subdomains !
  - Might require an A record associating an address with that name ("glue record")
    - Only when the authoritative name server for a delegated zone "lives" in this zone
  - This way NS RRs hold the name space together

Types of Resource Records (4)

- **MX - Mail Exchanger RR**
  - Specifies a mail exchanger host for that domain
  - RDATA consists of PREFERENCE and EXCHANGE
    - A domain may have as many MX records as available mail exchange servers
    - Mail transport agents will try the server with lowest (16 bit integer) PREFERENCE value first, then the others in increasing order
    - EXCHANGE contains the host name of that mail exchanger
- **HINFO - Host Information RR**
  - Provides information of the hardware and software used by this host (e.g. utilized by the FTP protocol)
  - RDATA consists of CPU and OS fields
    - Prefer standard values specified in RFC-1010 and RFC-1340

Types of Resource Records (5)

- **WKS - Well Known Service RR**
  - Specifies a well known service supported by a particular protocol on a particular host
  - RDATA contains
    - ADDRESS (32 bit) IP Address
    - PROTOCOL (8 bit) IP protocol number
    - BIT MAP (variable length) indicates the TCP port number, e.g. the 26th bit set indicates port 25 - SMTP
- **LOC - Location (EXPERIMENTAL)**
  - Allows DNS to carry location information about hosts and networks (example application: xtraceroute)
  - RDATA contains latitude, longitude and altitude information fields

Example Configuration (1)
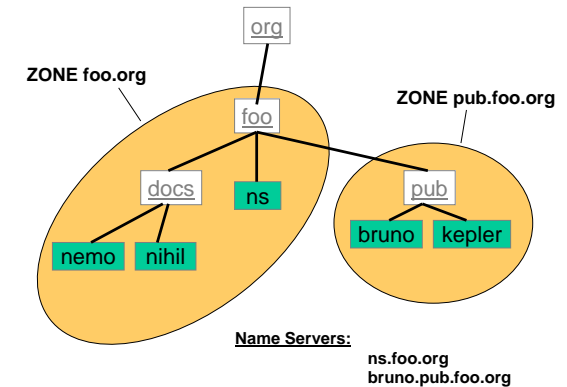
Example Configuration (2)

```
; zone file for the foo.org. zone
@              IN      SOA     ns.foo.org.  admin.nemo.docs.foo.org (
                               199912245        ;serial number
                       360000           ;refresh time
Records describing     3600             ;retry time
zone .foo.org. = @     3600000          ;expire time
                       3600             ;default TTL )
               IN      NS      ns.foo.org.
               IN      NS      ns.xyz.com.    ;secondary nameserver for @
               IN      MX      mail.foo.org.  ;mailserver for @
pub            IN      NS      bruno.pub.foo.org.
; glue records                                     Delegation for the
ns             IN      A       216.32.78.1         zone pub.foo.org.
bruno.pub      IN      A       216.32.78.99
; hosts in the zone foo.org
mail           IN      A       216.32.78.10
linus          IN      A       216.32.78.20
nemo.docs      IN      A       216.32.78.100
nihil.docs             IN      A       216.32.78.150
```

Example Configuration (3)

```
; zone file for the 78.32.216.in-addr.arpa domain
@              IN      SOA  ns.foo.org  admin.nemo.docs.foo.org.
                            (
                                    1034
                                    3600
                                    600
                                    3600000
                                    86400
                            )

               IN      NS      ns.foo.org.

1      IN      PTR     ns.foo.org.
10     IN      PTR     mail.foo.org.
20     IN      PTR     linus.foo.org.
99     IN      PTR     bruno.pub.foo.org.
100    IN      PTR     nemo.docs.foo.org.
150    IN      PTR     nihil.docs.foo.org.
```

Example Configuration (4)

```
; zone file for pub.foo.org
@              IN      SOA  bruno.pub.foo.org
     hostmaster.bruno.pub.foo.org.
                            ( 1034
                            3600
                            600
                            3600000
                            86400 )
; Name Servers
               IN      NS              bruno
               IN      NS              ns.foo.org.        ;secondary NS
; glue records
     bruno     IN      A               216.32.78.99
```

Example Configuration (5)

```
nameserver     IN      CNAME           bruno

; other hosts:
kepler         IN      A               216.32.22.50
               IN      MX              1  mail.foo.com
               IN      MX              2  picasso.art.net
               IN      MX              5  mail.ct.oberon.tuwien.ac.at
aristarch              IN      A               216.32.22.51
galilei        IN      A               216.32.22.52
               IN      HINFO           VAX-11/780  UNIX
               IN      WKS             216.32.22.52  TCP (telnet ftp
                                       netstat finger pop)
laplace        IN      A               216.32.34.2
               IN      HINFO           SUN  UNIX
; etc.....
```

BIND-8, BIND-9

- **New features:**
  - DNS Update (RFC 2136)
    - Authorized agents are allowed to update zone data by sending special update messages to add or delete RR
  - DNS Notify (RFC 1996)
    - Primary can notify the zone´s slaves when the serial number of the master file has incremented
  - Incremental zone transfer
    - Just the changes within a zone file are request and transfered
  - IP-address-based access control (= filters) for queries, zone transfers and updates
    - To increase or enable security
  - Many bug fixes and more secure

Diagnostic Tools

- **DIG - Domain Information Groper**
  - Send domain name query packets to name servers
  - Comand-line driven
  - Results are printed in a human-readable format
  - dig [@server] domain [<query-type>] [<query-class>] [+<query-option>] [-<dig-option>] [%comment]

- **NSLOOKUP**
  - Query Internet name servers interactively
  - More powerful utility as DIG

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
  - Introduction
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol
- **SNMP**

The "DNS Protocol"

- **DNS messages utilize TCP or UDP as transport protocol**
  - UDP for standard queries (need for performance)
  - TCP for zone transfers (need for reliability)
- **Well known port number 53 (server side)**
- **DNS messages using UDP are restricted to 512 bytes**
  - Longer messages are truncated and the TC bit is set in the header

Message Format

DNS messages have always the following 5 sections:

HEADER	Specifies which sections are present, query or response, etc
QUESTION	Contains the question for the NS
ANSWER	Contains **RRs** answering the question
AUTHORITY	Contains **RRs** pointing toward an authority
ADDITIONAL	Contains **RRs** holding additional information

Some sections (except HEADER) may be <u>empty</u> in certain cases

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   105

Header Section

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
IDENTIFICATION															

QR	OPCODE	AA	TC	RD	RA	Z	RCODE

QDCOUNT	(number of questions)
ANCOUNT	(number of answers)
NSCOUNT	(number of authority)
ARCOUNT	(number of additional)

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   106

Header Fields (1)

- **IDENTIFICATION**
  - 16 bit identifier assigned by the requesting program
  - the corresponding reply gets the same identifier
- **QR**
  - query = 0, response = 1
- **OPCODE**
  - Specifies the kind of query in this message
    - 0 .... standard query (QUERY)
    - 1 .... inverse query (IQUERY); IN-ADDR.ARPA !!!
    - 2 .... server status request (STATUS)
    - 3 -15 ... reserved

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   107

Header Fields (2)

- **AA**
  - Authoritative Answer
  - The responding NS is an authority for the domain name in the question section
  - If set, the data comes directly from a primary or secondary name server and not from a cache
- **TC**
  - TrunCation
  - Indicates that this message has been truncated (due to transmission channel's max message size)
- **RD**
  - Recursion Desired
  - The NS should solve the query recursively

© 2012, D.I. Lindner / D.I. Haas   BootP, TFTP, DHCP, DNS, SNMP, v4.5   108

Header Fields (3)

- **RA**
  - Recursion Available
  - May be set or cleared in a response
  - Indicates whether recursive queries are supported by the NS
- **Z**
  - Reserved
  - Must be zero

Header Fields (4)

- **RCODE**
  - Response Code
  - 0 ... *no error*
  - 1.... *format error* - the NS was not able to interpret the query
  - 2 ... *server failure* - the NS has problems
  - 3 ... *name error* - an authoritative NS signals that the requested domain does not exist
  - 4 ... *not implemented* - the NS does not support this kind of query
  - 5 ... *refused* - the NS refuses the required operation for policy reasons
  - 6-15 ... reserved for future use

Header Fields (5)

- **QDCOUNT**
  - Specifies the number of entries in the question section
- **ANCOUNT**
  - Specifies the number of RRs in the answer section
- **NSCOUNT**
  - Specifies the number of NS RRs in the authority records section
- **ARCOUNT**
  - Specifies the number of RRs in the additional records section

Question Section

The question section contains QDCOUNT entries, each of the following format:

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |

| QNAME | variable length |
| QTYPE |
| QCLASS |
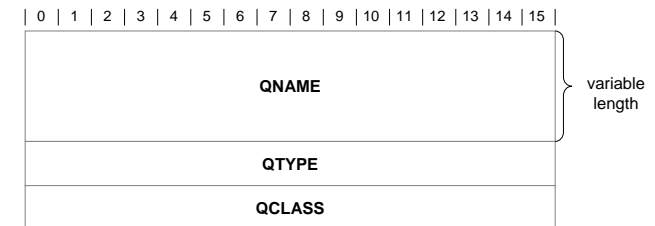
Question Fields

- **QNAME**
  - A domain name represented as a set of labels
    - See the domain name message format below
  - Can have an odd number of octets, no padding is used as reminder
- **QTYPE**
  - Type of query; values are a superset of the TYPE values in RRs
    - AXFR (252) request for a transfer of the entire zone
    - " * " (255) request for all records
- **QCLASS**
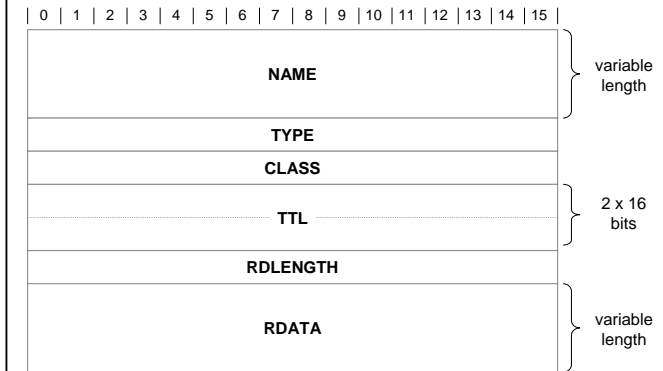 - Class of the query; values are a superset of the CLASS values in RRs (usually "IN" for Internet, " * " for any class)

Domain Names in Messages
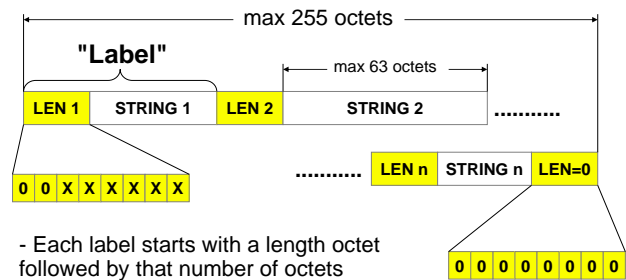
- Each label starts with a length octet followed by that number of octets

- The domain name is terminated with a zero length octet (= "null label" for the root)

Resource Record Format in Answers, Authorative and Additional Fields

Resource Record Fields (1)

- **NAME**
  - Domain name to which this RR refers
- **TYPE**
  - Specifies the meaning of the data in the RDATA field
  - e.g. A, CNAME, NS, SOA, PTR, ...
- **CLASS**
  - Specifies the class of the data in the RDATA field
- **TTL**
  - Specifies the duration this RR may be cached before it should be discarded
  - Zero values suggest that this RR should not be cached
  - 32 bit, time in seconds

Resource Record Fields (2)

- **RDLENGTH**
  - Specifies the length in octets of the RDATA field

- **RDATA**
  - Variable length string that specifies the resource
  - The format depends on the TYPE and CLASS field
    - E.g. if TYPE=A and CLASS=IN, then RDATA contains an IP address

Message Compression Example

30	**00**	**1**	A
32	**00**	**4**	D
34	O		C
36	S	**00**	**3**
38	F		O
40	O	**00**	**3**
42	O		R
44	G	**00**	**0**

A.DOCS.FOO.ORG

46	**00**	**1**	B
48	**11**	**32**	

B.DOCS.FOO.ORG

50	**11**	**37**

FOO.ORG

Message Compression

- **To reduce the size of messages DNS provides a simple compression method**
- **Repetitions of domain names can be replaced with a pointer to the previous occurance**
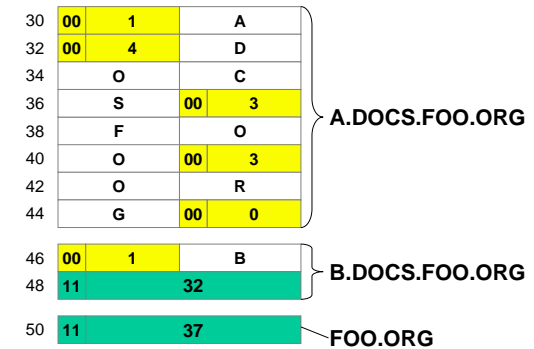 - Works even for part of domain names (list of labels)

Pointer format:

1	1	OFFSET

Helps to distinguish a pointer from a label

Specifies the distance from the start of the message (= from the first octet of the ID field)

Selected RFCs (1)

- **RFC 1034**
  - Domain Name Concept And Facilities
- **RFC 1035**
  - Domain Name Implementation and Specification
- **RFC 1101**
  - DNS Encoding Network Names And Other Types
- **RFC 1183**
  - New DNS RR Definitions
- **RFC 1591**
  - Domain Name System Structure And Delegation

Selected RFCs (2)

- **RFC 1664**
  - Using The Internet DNS To Distribute RFC1327 Mail Address Mapping Tables
- **RFC 1712**
  - DNS Encoding Of Geographical Location
- **RFC 1788**
  - ICMP Domain Name Messages
- **RFC 1794**
  - DNS Support For Load Balancing
- **RFC 1995**
  - Incremental Zone Transfers In DNS

Selected RFCs (3)

- **RFC 1996**
  - A Mechanism For Prompt Notification Of Zone Changes (DNS Notify)
- **RFC 2052**
  - A DNS RR For Specifying The Location Of Services (DNS SRV)
- **RFC 2065**
  - Domain Name System Security Extensions
- **RFC 2136**
  - Dynamic Updates In The Domain Name System (DNS Update)

Selected RFCs (4)

- **RFC 2308**
  - Negative Caching Of DNS Queries (DNS Ncache)
- **RFC 2535**
  - Domain Name System Security Extensions
- **RFC 2541**
  - DNS Security Operational Considerations
- **RFC 2606**
  - Reserved Top Level DNS Names
- **RFC 3007**
  - Secure Domain Name System Dynamic Update

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
- **SNMP**
  - Network Management Basics
  - SNMP
    - Basics
   - SMI
    - MIB
  - RMON
  - SNMPv2

What is it?

- **A network management should...**
  - automate the process of monitoring and adjusting the performance of a network
  - trigger alarms when special events occur
  - not burden the network
- **Two important implementations**
  - IETF Network Management based on SNMP
  - OSI Network Management Architecture
- **SNMP is the most important implementation**
  - Industry standard

Four Basic Elements

- **Management Process (Client)**
  - Central workstation
- **Agent Process (Server)**
  - Routers, hosts, ...
- **Management Information Base (MIB)**
  - The logical structure of this database is defined in the "Structure of Management Information" (SMI)
- **Communication Protocol**
  - Special PDU´s (e.g. SNMP)

Client/Server System
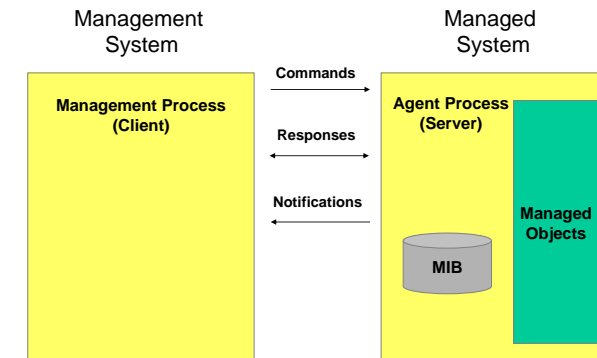
The MIB

- **The whole management information is organized as tree-structure**
  - Branches represent logical categories
  - Leaves contain information about objects
- **Tree can be locally or globally valid**
- **ISO administers one of the branches of the global tree**
  - iso(1)

Requirement for a SMI

- **Structure of Management Information (SMI)**
  - Specifies how information about managed objects is to be represented
- **This information representation is implemented using a restricted version of the ISO's Abstract Syntax Notation One (ASN.1)**
  - ASN.1 was designed to describe data types for the OSI presentation layer
  - Similar to a higher level programming language

ISO/OSI Network Management Model

- **ISO/IEC 7498-4**
  - OSI network management framework
- **ISO/IEC 9595 or CCITT X.710**
  - Common Management Information Service (CMIS)
  - Manages the OSI-MIB
- **ISO/IEC 9596 or CCITT X.711**
  - Common Management Information Protocol (CMIP)
  - Layer 7, connection oriented

- **Note: Also IEEE 802.1B LAN/MAN management standard uses ISO's CMIP**

IETF Management Framework

- **Late 1980s: IAB realized the demand for a general Internet management architecture**
- **Initially three proposals:**
  - High-level entity management system (HEMS)
  - OSI-based system using CMIS and CMIP
  - An extended version of the Simple Gateway Monitoring Protocol (SGMP)
    - Used in the early Arpanet (RFC 1028, historic)
- **IAB decided for a extended version of SGMP called Simple Network Management Protocol (SNMP)**

IETF Management Framework

- **SNMP was intended as short-time solution only (!)**
  - Should be replaced later by an OSI approach
- **So IAB formed another working group for "CMOT"**
  - CMIS/CMIP over TCP/IP = CMOT
  - RFC 1052 reflects these efforts
- **But this development could not keep up with time**
  - Too complex
  - No working stacks at that time
- **Hence SNMP became an industrial standard**

CMOT

- **CMIP over TCP/IP is known as CMOT**
  - RFC-1095
- **In 1990 the IAB recommended CMOT**
  - Although the IETF regards this work as "historic"
  - However, some vendors use CMOT
- **Main differences to SNMP:**
  - TCP instead of UDP
  - Connection-oriented application layer
  - CMOT's application layer is built on OSI services
  - MIB-II-OIM instead of SNMP's MIB-II

CMOT

- **CMOT is much more sophisticated**
  - CMIP object definitions are more comprehensive
- **In CMOT the management application relies on three OSI services:**
  - Common Management Information Service Element (CMISE)
  - Remote Operation Service Element (ROSE)
  - Association Control Service Element (ACSE)
- **And a Lightweight Presentation Protocol (LPP)**
  - As presentation layer

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
- **SNMP**
  - Network Management Basics
  - SNMP
    - Basics
    - SMI
    - MIB
  - RMON
  - SNMPv2

SNMP

- **What does SNMP ?**
  - Using SNMP, a NMS can set and retrieve variables of a MIB
- **Why "Simple" Network Management Protocol?**
  - Because the agents require only minimal software
  - Only the network management system (NMS, client side!) must provide enough CPU and memory resources
  - Only a small set of messages
    - Client: Get, GetNext, Set
   - Agents: Response, event notification (trap)

SNMP

- **Connectionless Communication**
  - Via UDP
    - port numbers 161 (Agent) and 162 (Client)
- **SNMP is described in RFC 1157**
  - IAB recommends the usage of the Internet MIB (RFC-1156) and at least one of the recommended management protocols, SNMP or CMOT
- **NMI acts as SNMP client**
  - Polls SNMP Agents periodically and collects performance statistics from them
- **Managed device is called SNMP Agent**
  - Runs SNMP server software
  - Can send so-called TRAPS to the NMI in case of an event (alarm)

Typical Agents

- **Typical SNMP agents:**
  - Wiring hubs
  - Network servers
  - Hosts with network interface cards
  - Internetworking devices, such as bridges and routers
  - Test equipment, such as monitors and analyzers
  - Uninterruptible Power Supplies (UPS)
- **Additionally, SNMP supplemental's are commonly supported, such as**
  - HTTP/HTML based systems
  - Web-based Enterprise Management (WBEM) solution
  - Java Management Application Programming Interface (JMAPI) proposed by SunSoft

SNMP Architecture
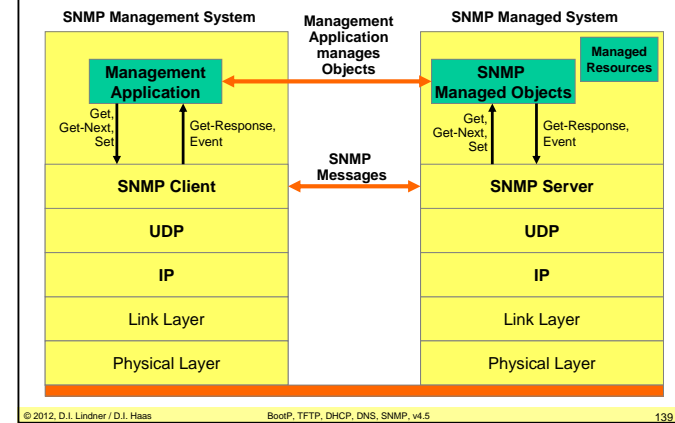
SNMP Message Format
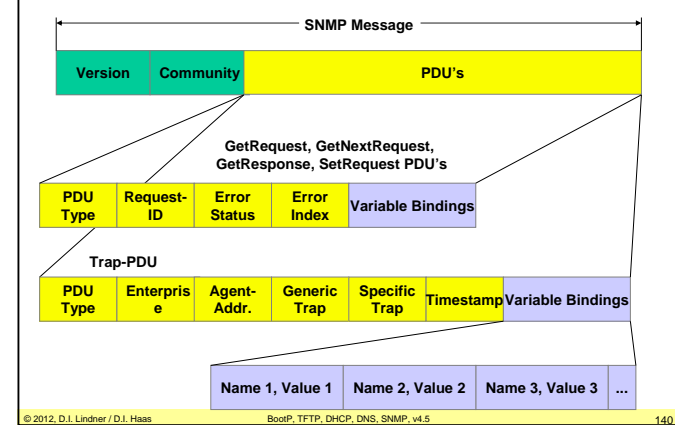
SNMP Header Entries

- **Version**
  - Specifies the protocol version (0..SNMPv1)
- **Community**
  - Contains a "community" password to protect the session
  - Note: cleartext!
    - Security hole if set commands are allowed
    - Hence many vendors disallow set Commands or provide proprietary cryptographic solutions
    - Solved in SNMPv2
- **Request-ID**
  - Identifies corresponding requests/responds

SNMP Header Entries

- **Error Status**
  - 0 (No Error), 1 (Too Big), 2 (No Such Name), 3 (Bad value), 4 (Read Only), 5 (General Error)
  - In Requests always set to zero
- **Error Index**
  - Points to the variable of the Request message that caused the error
  - In Requests always set to zero
- **Variable Bindings**
  - List of object names plus associated values
  - In Requests the values are always set to zero
- **Enterprise**
  - Type of management agent who generated the trap
  - Based on sysObjectID

SNMP Header Entries

- **Agent-Addr**
  - Address of management agent
- **Generic Trap**
  - 0 (Cold Start), 1 (Warm Start), 2 (Link Down), 3 (Link Up), 4 (Authentication Failure), 5 (EGP Neighbor Loss), 6 (Enterprise Specific)
- **Specific Trap**
  - Special vendor specific trap
- **Time-Stamp**
  - Contains system time at which the trap occured
  - Based on sysUpTime

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
- **SNMP**
  - Network Management Basics
  - SNMP
    - Basics
    - SMI
    - MIB
  - RMON
  - SNMPv2

SMI

- **Structure of Management Information (SMI)**
  – RFC 1155, RFC 1212, RFC 1215
- **SMI organizes, describes, and names information**
  – Thus providing accessibility
- **Each managed object must consist of**
  – a name
  – a syntax
 – and an encoding
- **A MIB contains these objects**

Managed Objects

- **Name**
  – Is also called Object Identifier (OID)
  – Uniquely identifies the object
- **Syntax**
  – Defines the data type (integer, string of bytes, etc.)
  – Described with the Abstract Syntax Notation One (ASN.1)
- **Encoding**
  – Describes how the information is transmitted between machines
  – Using so-called Basic Encoding Rules (BER) to specify the transfer syntax

SMI – Names

- **Each object must have a name**
  – An object is either a device
  – or a characteristic of a device
- **Name = sequence of integers separated by dots**
  – Also known as Object Identifier
  – Represents the tree structure of a MIB
- **Internet sub-tree uses prefix 1.3.6.1**
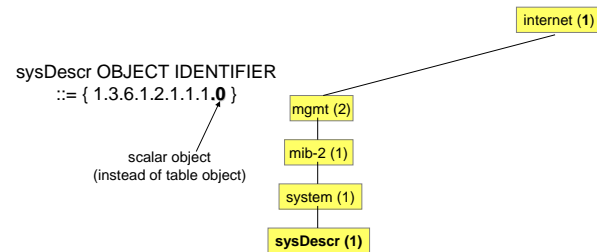 – Administered by IANA
  – RFC 1700

SNMP Related OID Subtree
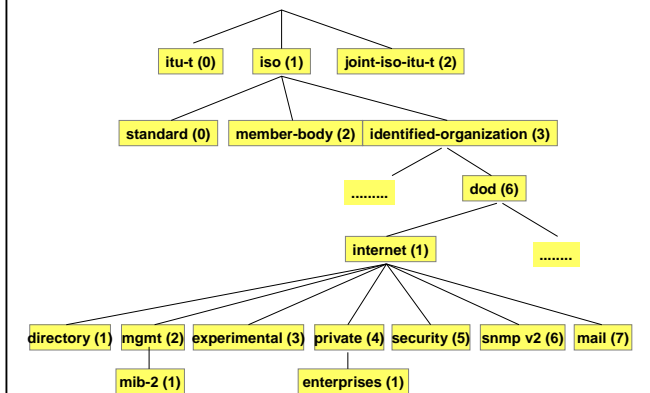
Name Example

ASN.1 Notation:

```
mgmt OBJECT IDENTIFIER ::= { internet 2 }
mib-2 OBJECT IDENTIFIER ::= { mgmt 1 }
system OBJECT IDENTIFIER ::= { mib-2 1 }
sysDescr OBJECT IDENTIFIER ::= { system 1 }
```

internet (**1**)

sysDescr OBJECT IDENTIFIER
::= { 1.3.6.1.2.1.1.1.**0** }

scalar object
(instead of table object)

mgmt (2)

mib-2 (1)

system (1)

sysDescr (1)

MIB Internet Branches

- **directory (1)**
  - was reserved for the OSI directory within the Internet
- **mgmt (2)**
  - contains all Internet Standard MIBs
- **experimental (3)**
  - is used for Internet experiments (IANA)
  - if certain objects are approved they move to an official branch
- **private (4)**
  - is used by vendors to register their own MIBs

SMI – Syntax

- **According ISO/OSI the ASN.1 is a presentation layer function**
  - For simplicity, SNMP uses only a subset of ASN.1
- **ASN.1 defines structured information in a machine-independent fashion**
  - Provides basic data types
  - All data types can be defined with this basic data types
- **ASN.1 describes:**
  - Types and values
  - Macros
  - Modules

ASN.1 – Types

- **Primitive data types (aka Simple Types)**
  - INTEGER, OCTET STRING, OBJECT IDENTIFIER, NULL
- **Subtypes**
  - E.g. INTEGER (0..255)
  - E.g. special cases of OCTET STRING
    - DisplayString: only printable ASCII characters
    - PhysAddress: for MAC addresses
    - octetBitstring: for bitstrings longer than 32 bits
- **Constructor types**
  - SEQUENCE, SEQUENCE OF
  - Define tables and rows within those tables

ASN.1 – Types

- **Defined types**
  - RFC 1213: NetworkAddress, IpAddress, Counter, Gauge, TimeTicks, and Opaque
- **Tagged types**
  - To differentiate between defined types
  - Types are defined using other defined types as basis and by adding additional information (tags)
  - Tags consist of class and number for the contained type
    - class universal: e.g. INTEGER, OCTET-STRING, ...
    - class application: e.g. IPAddress, Counter, ...
    - class context-specific: SNMP PDUs such as GetRequest, GetResponse, ...
    - class private: for enterprise specific applications

ASN.1 – Macros

- **Macros extend the ASN.1 language**
  - They are used to describe objects
- **Example for object "tcpInSegs"**

```
tcpInSegs OBJECT-TYPE
        SYNTAX  Counter
        ACCESS  read-only
        STATUS  mandatory
        ::= { tcp 10 }
```

  - Used Macro: OBJECT-TYPE
    - Is the most important macro (RFC 1213)
  - The object "tcpInSegs" represents a counter
  - This object can be accessed read-only under the name "tcp 10"
    - That is, tcpInSegs is the 10th object in the tcp group
  - This object is mandatory for all devices in mib-2.tcp

ASN.1 – Modules

- **Modules represent a collection of types, values and macros**
- **Structure of a module**
  - Module name
    - e.g. RMON-MIB
  - Body is enclosed with BEGIN and END
  - IMPORT-statement is used to collect types, values and macros from other modules
    - e.g. the type "Counter" from the module RFC1155-SMI
    - e.g. the macro "OBJECT-TYPE" from the module RFC-1212

Example Module

- **For example the RMON MIB module contains the following code:**

```
RMON-MIB DEFINITIONS ::= BEGIN
        IMPORTS
            Counter                 FROM RFC1155-SMI
            DisplayString           FROM RFC1158-MIB
            mib-2                   FROM RFC1213-MIB
            OBJECT-TYPE             FROM RFC-1212
            TRAP-TYPE               FROM RFC-1215;
    -- This is a comment line
    -- Remote Network Monitoring MIB
        rmon    OBJECT IDENTIFIER ::= { mib-2 16 }
        ...
    END
```
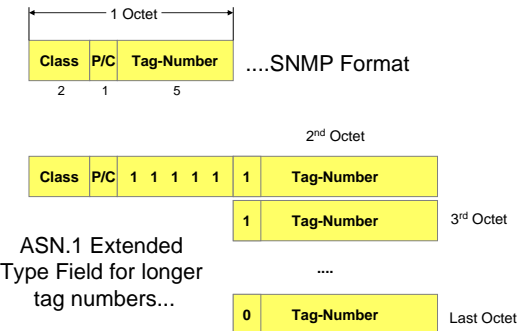
SMI – Encoding

- **Basic Encoding Rules (BER)**
  - Defines how to convert ASN.1 data types into bit patterns
  - Specified in ISO 8825-1
- **BER uses "Type-Length-Value" (TLV) coding**
  - Also other names used, such as
    - "Identifier-Length-Contents"
    - "Tag-Length-Value"
  - Each message is structured accordingly

BER – Type Field

- **Type field**
  - Contains the ASN.1 tag (class and number) to identify the data in the Value field
- **SNMP uses only one octet for the Type field**
  - Bits 1-5: tag numbers (0-30)
    - 2 .. Integer, 4 .. Octetstring, 5 ..Null, 6 .. Object Identifier, 16 .. Sequence
  - Bit 6: P/C bit (Primitive/Constructed)
  - Bits 7,8: class
    - 00..Universal, 01..Application, 10..Context Specific, 11..Private

© 2012, D.I. Lindner / D.I. Haas

Page Appendix 1 - 79

Type Field

....SNMP Format

ASN.1 Extended Type Field for longer tag numbers...

BER – Length and Value Fields

- **Length field**
  - Specifies the number of bytes in the Value field
  - Size of length field: 1-127
    - MSB = 0 means: 1 octet
    - MSB = 1 means: first octet specifies the number of octets of the length field
- **Value field**
  - 0 - $(2^{1008}-1)$ octets theoretically possible
  - Contains either an
    - Integer
    - ASCII character
    - OBJECT-IDENTIFIER

© 2012, D.I. Lindner / D.I. Haas

Page Appendix 1 - 80

Value – OBJECT-IDENTIFIER

- **OBJECT-IDENTIFIERs**
  - Are also encoded
  - Encoding trick for first (X) and second (Y) subidentifer
    - Encoded value = (40 * X) + Y
    - To save one octet
- **Example:**
  - { iso org(3) dod(6) internet(1) mgmt(2) mib-2 (1) 1 }
  - encoded as 0x 2B 06 01 02 01 01

Example BER for a SNMP Message

GetRequest für sysDescr (1.3.6.1.2.1.1.1)

```
30   29   02   01   00
sequence, len=41, integer, len=1, version=0
04   06   70   75   62   6C   69   63
string, len=6, communityname= PUBLIC
A0   1C   02   04   05   AE   56   02
GetReq., len=28, integer, len=4, request-id=05AE5602
02   01   00   02   01   00
integer, len=1, status=0, integer, len=1, error-index=0
30   0E   30   0C   06   08
sequence, len=14, sequence, len=12, object-id, len=8
2B   06   01   02   01   01   01   00
1.3 . 6 . 1 . 2 . 1 . 1 . 1 . 0
05   00
null, len=0
```

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
- **SNMP**
  - Network Management Basics
  - SNMP
    - Basics
    - SMI
   - MIB
  - RMON
  - SNMPv2

MIB

- **All Internet Standard MIBs are in the mgmt (2) sub-tree**
- **MIB-I: 8 objects**
  - System, Interfaces, Address Translation, IP, ICMP, TCP, UDP, and EGP
  - RFC 1156, May 1990
- **RFC 1212: Concise MIB**
  - Reason: several proprietary MIB's occur
  - Demand for consistent format for MIB modules
- **MIB-II: Replaced MIB-I**
  - RFC 1213, March 1991

MIB Entries

- **MIB objects**
  - Are described with the ASN.1 OBJECT-TYPE macro
- **MIB table entries**
  - Are also described using the OBJECT-TYPE macro
  - plus the SEQUENCE OF constructor type to define and attach the table at the tree
  - plus the INDEX statement to define the row
  - plus the SEQUENCE constructor type to define the columns of a row
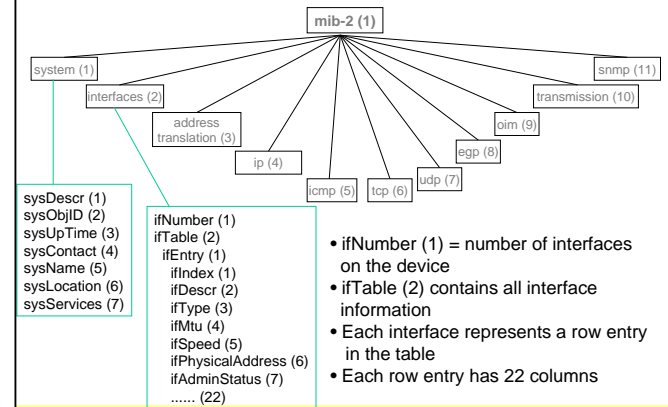
Example: ASN.1 Table Objects

```
udpTable  OBJECT-TYPE              udpEntry      OBJECT-TYPE
  SYNTAX SEQUENCE OF UdpEntry        SYNTAX  UdpEntry
  ACCESS not-accessible              ACCESS  not-accessible
  STATUS mandatory                   STATUS  mandatory
  DESCRIPTION                        DESCRIPTION
   "A table containing UDP listener     "Information about a particular
   information.                          current UDP listener."
  ::= { udp 5 }                        INDEX { udpLocalAddress, udpLocalPort }
                                     ::= { udpTable 1 }

udpLocalAddress  OBJECT-TYPE
        SYNTAX    IpAddress        UdpEntry ::=
        ACCESS    read-only          SEQUENCE {
        STATUS    mandatory            udpLocalAddress IpAddress,
        DESCRIPTION                    udpLocalPort INTEGER (0..65535)
  ::= { udpEntry 1 }                 }

udpLocalPort    OBJECT-TYPE
   SYNTAX  INTEGER (0..65535)       Note that the sequence name is the same
   ACCESS  read-only                as the row name except that it begins
   STATUS  mandatory                with an uppercase letter (Convention)
   DESCRIPTION
    "The local port number
    for this UDP listener."
  ::= { udpEntry 2 }
```

MIB-II Excerpt

- ifNumber (1) = number of interfaces on the device
- ifTable (2) contains all interface information
- Each interface represents a row entry in the table
- Each row entry has 22 columns

Variables of the "System" Group

- sysDescr (1): Display String, RO, M
  - Description of the hardware, operating system, etc
- sysObjID (2): Object Identifier, RO, M
  - Number-string representing the OID of the manufacturer's private MIB for this device
- sysUpTime (3): TimeTicks, RO, M
  - This Agent's last initialization time
- sysContact (4), sysName (5), sysLocation (6): Display Strings, RW, M
  - Can be utilized by the network manager
- sysServices (7): Integer, RO, M
  - A number specifying delivered services/layer

Variables of the "Interface" Group

- ifNumber (1): Integer, RO, M
  - Number of network interfaces
- ifTable (2): Sequence of ifEntry, NA, M
  - List of ifEntry (2.1) using the following entries:
- ifDescre (2.1.2): Display String RO, M
  - Information from manufacturer
- ifType (2.1.3): Integer, RO, M
  - 05 ... X.25 (RFC 877), 06 ... Ethernet, 07 ... 802.3,
  - 09 ... 802.5, 10 ... 802.6 (MAN), 15 ... FDDI,16 ... LAPD,
  - 17 ... Sdlc, 19 ... ES1, 20 ... Basic ISDN, 21 ... Primary ISDN,
  - 23 ... PPP, 28 ... Slip, 32 ... Frame Relay

Variables of the "Interface" Group

- ifMtu (2.1.4): Integer, RO, M
- ifPhysAddress (2.1.6): Integer, RO, M
  - Hardware Address (e.g. MAC-Address)
- ifAdminStatus (2.1.7): Integer, RW, M
  - 1 .. up, 2 .. down, 3 .. testing (desired state)
- ifOperStatus (2.1.8): Integer, RO, M
  - 1 .. up, 2 .. down, 3 .. testing (actual state)
- ifLastChange (2.1.9): Integer, RO, M
  - System-time when interface become active
- ifInOctets (2.1.10): Counter, RO, M
  - Total number of received octets

Variables of the "Interface" Group

- ifInDiscards (2.1.13): Counter, RO, M
  - Total number of rejected input packets (because of input buffer overflow)
- ifInErrors (2.1.14): Counter, RO, M
  - Total number of received damaged packets
- ifOutOctets (2.1.16): Counter, RO, M
  - Total number of sent octets
- ifOutDiscards (2.1.19): Counter, RO, M
  - Total number of rejected output packets (because of output buffer overflow)
- ifOutErrors (2.1.20): Counter, RO, M
  - Total number of not-sent packets (because of errors)

Meaning of the Remaining Groups

- **Group "Address Translation"**
  - Mapping between network layer address (IP-address) and physical address
- **Croup "IP"**
  - IpAddrTable ... assigns IP-addresses, subnet-masks, and broadcast-methods to physical network interfaces (RO)
  - IpRouteTable ... contains Routing-table of this device (RW)
  - Counter for all IP-packets:
    - ipInReceives, ipInDiscard, ipInHdrError, ipInAddrError,
    - ipForwDatagrams, ipInDelivers, ipOutRequests,
    - ipOutDiscards,ipOutNoRoutes, ipFragOk, ipFragFails, etc...

Meaning of the Remaining Groups

- **Group "ICMP"**
  - Contains a counter for several packet-types and events
- **Group "TCP"**
  - Contains a counter for segments, parameters (e.g. retransmission timer value), session information (local and remote socket) and associated states, etc ...
- **Group "UDP"**
  - Contains a counter for UDP datagram's and information about currently involved connection endpoints (socket information)
- **Group "EGB"**
  - Contains a counter for EGP-messages and a table of neighbor routers

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
- <u>**SNMP**</u>
  - Network Management Basics
  - SNMP
    - Basics
    - SMI
    - MIB
  - <u>RMON</u>
  - <u>SNMPv2</u>

RMON

- **Monitoring of specific MIB variables**
  - Requires continuous polling of these variables
  - Doesn't replace functionality of a protocol analysator
- **Solution: Remote Monitoring (RMON)**
  - Defines network monitoring functions and communications between NMS and remote monitors
  - RMON agent supervises local segment autonomously
    - Delivers only results of statistics to the NMS
    - E.g. Traffic Matrix
  - RMON-MIB is a sub-tree below MIB-II
    - Using identifier 16

Agenda

- **BootP**
- **DHCP**
- **TFTP**
- **DNS**
- <u>**SNMP**</u>
  - Network Management Basics
  - SNMP
    - Basics
    - SMI
    - MIB
  - RMON
  - <u>SNMPv2</u>

SNMPv1 Restrictions

- **Not suitable for large networks**
  - Polling efforts
  - Consider WAN links
- **Not efficient for carrying large data volumes**
  - Such as routing tables
- **Traps are not acknowledged**
  - Agent cannot be sure if its alarm reached NMS
- **Authentication is transmitted in plain-text Community strings**
- **No manager-to-manager communication**
- **Solution: SNMPv2**

SNMPv2 – New PDUs

- **GetBulkRequest**
  - For transmission of large data volumes such as routing tables
- **InformRequest**
  - For MIB-communication between network managers (client applications)
  - Hereby supporting multiple management stations
- **Report**
  - Not yet defined (?)
- **Trap PDU structure conforms to the format of the other PDU's**

SNMPv2 – Features

- **SNMPv1 was only designed to be transported via UDP and IP**
- **SNMPv2 defines several "Transport Mappings" in order to utilize other transport protocols**
  - OSI Connectionless Transport Service (CLTS, RFC 1418)
  - IPX (RFC 1420)
  - AppleTalk DDP (RFC 1419)