Application Protocols for Administration

BootP, TFTP, DHCP, DNS

Agenda

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

Datenkommunikation 384.081 - SS 2007

L12 - BootP, TFTP, DHCP, DNS

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

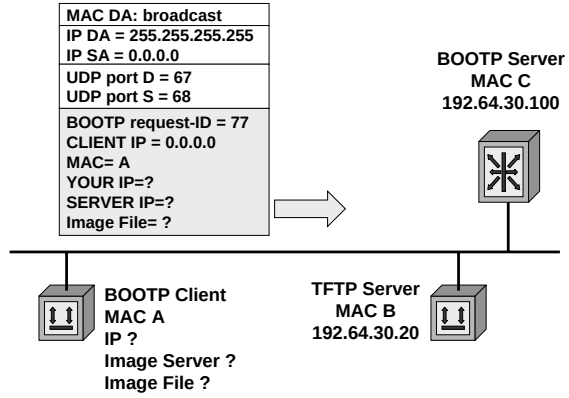
BOOTP-Principles

- **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**

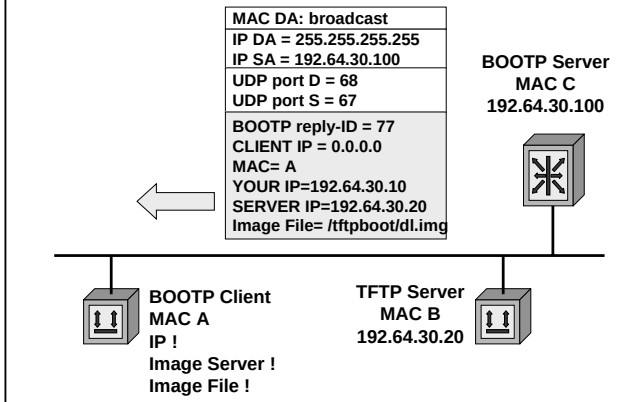
BOOTP-Principles

- **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



Bootstrap 2



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

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 !!!)			

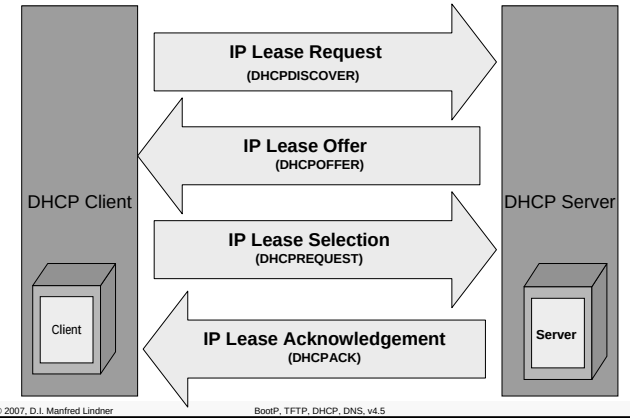
DHCP Message Types in Option Field

- 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



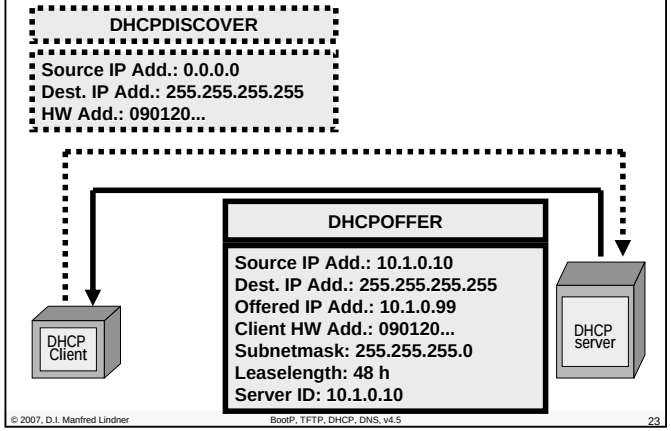
IP Lease Request

- **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



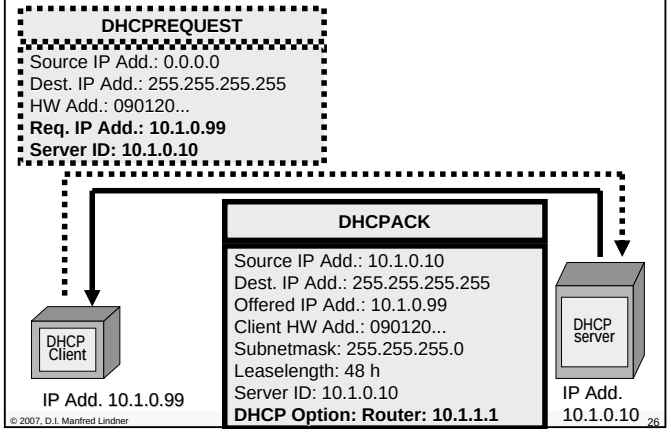
IP Lease Selection

- **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



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

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 (4)	BLOCK#

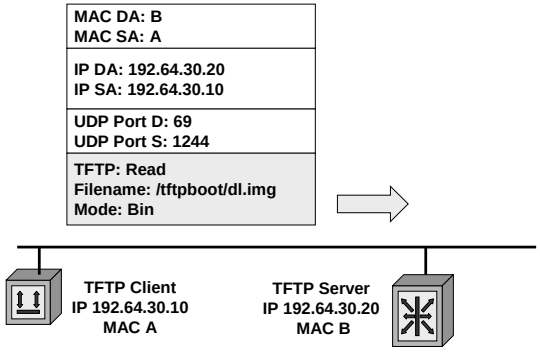
Type 4

- **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**

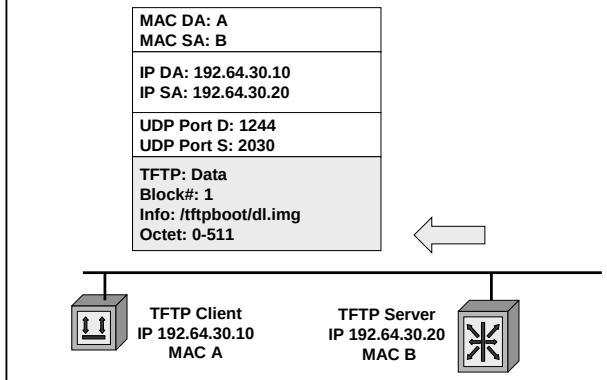
TFTP Protocol Description

- 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

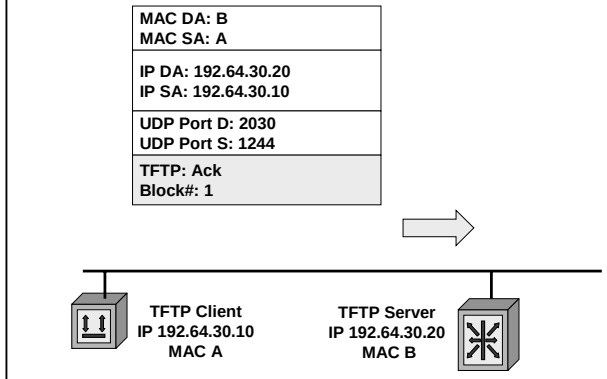
TFTP (1)



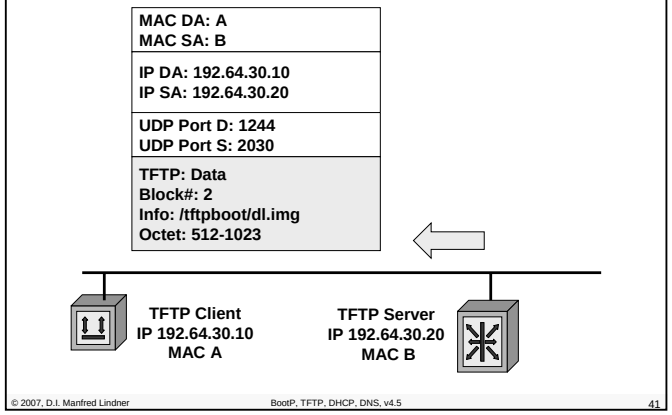
TFTP (2)



TFTP (3)

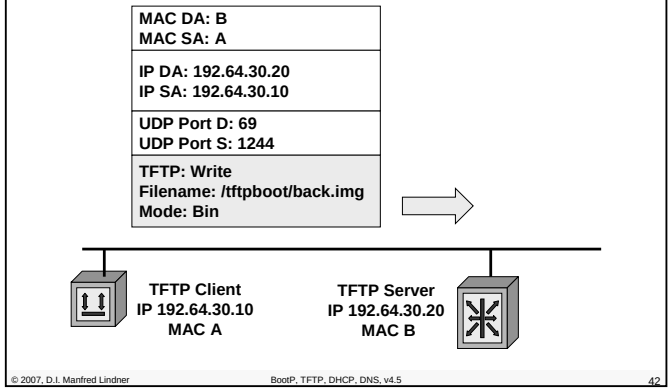


TFTP (4)



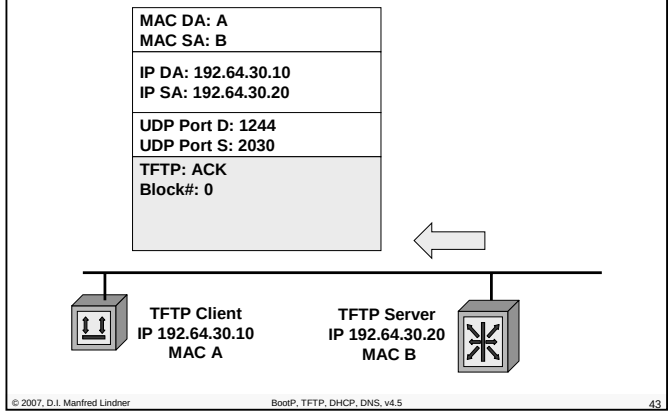
 BootP, TFTP, DHCP, DNS, v4.5 41

TFTP (1) Write



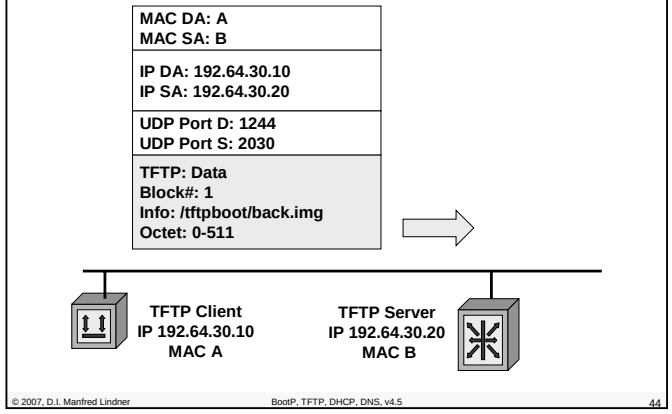
 BootP, TFTP, DHCP, DNS, v4.5 42

TFTP (2) Write



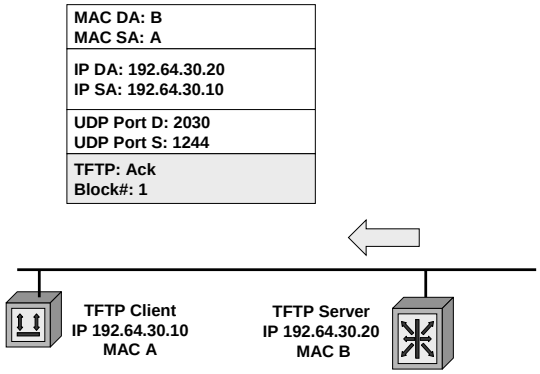
 BootP, TFTP, DHCP, DNS, v4.5 43

TFTP (3) Write



 BootP, TFTP, DHCP, DNS, v4.5 44

TFTP (4) Write



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

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 Domain Name System (DNS) 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)

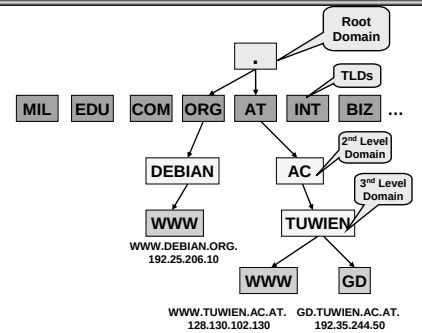
Mnemonic Approach

- **Problem: 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 directory for such a huge number of hosts?
- **Solution:**
  - hierarchy 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 human readable 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 tree-like directory hierarchy**
  - each part of the hierarchy is called a "domain", each hierarchy level is assigned a label, called "domain name"
  - the Domain Name Tree does NOT 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**
  - Name Servers
- **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

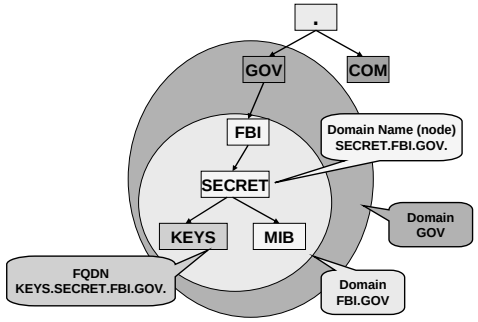
Conventions (1)

- Terminology**: a "Domain" ...**
  - is a complete subtree
    - everything under a particular point in the tree
  - relates to the naming structure itself, not the way things are distributed
- Terminology**: a "Domain Name" ...**
  - is the name of a node 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)

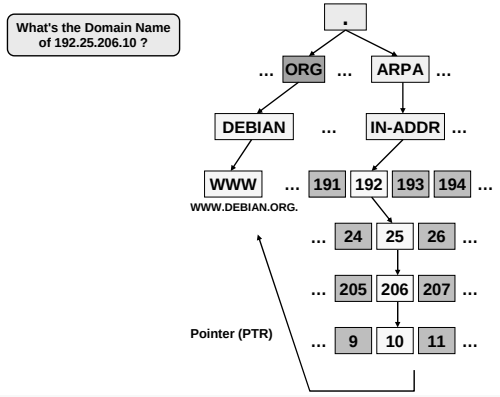
IN-ADDR.ARPA (1)

- **special feature: the *in-addr.arpa domain***
  - used to support gateway location
  - enables reverse lookups: 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
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol

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 name server called named ("d" stands for "daemon")
  - a resolver library for client applications
    - The "resolver" is a collection of functions like gethostbyname(2) and gethostbyaddr(2)
- **technically, BIND and DNS deal primarily with zones**
  - a zone is a part of the domain space

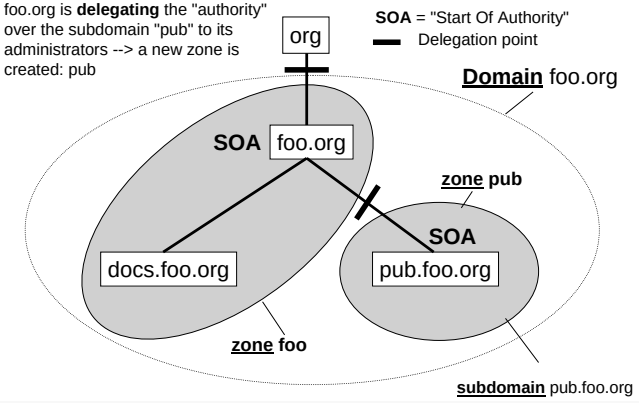
What is a Zone ?

- **a zone 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
  - zones relate to the way the database is partitioned and distributed
- **a name server is *authoritative* over a domain**
  - if he keeps a *master file* (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

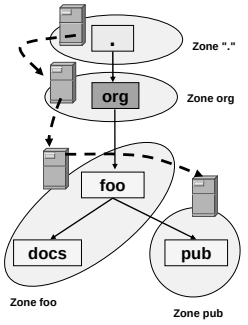


Delegation and Name Servers

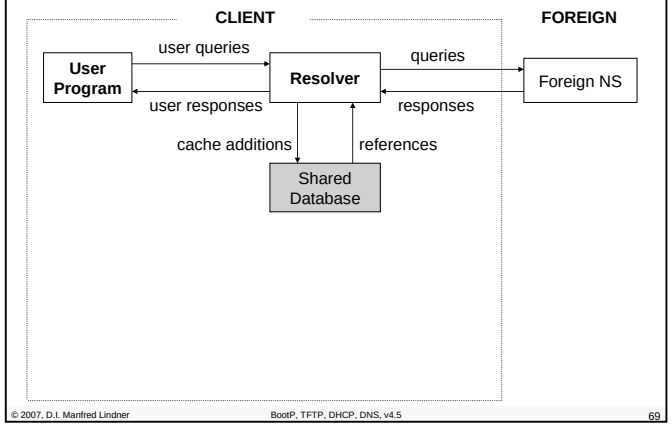
Root Server responsible for root domain delegates authority for building symbols org. to NS below which holds the master file for zone org

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

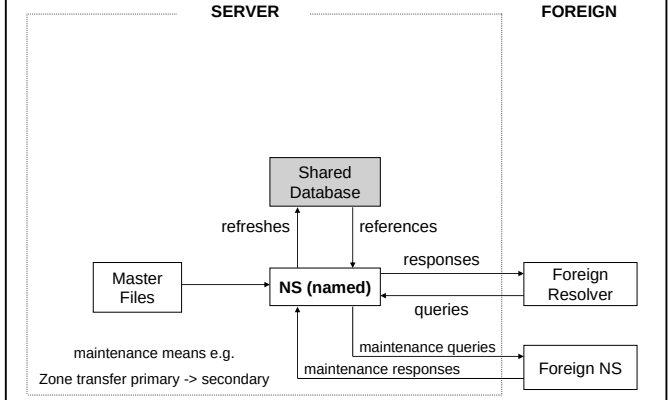
NS responsible for domain foo.org delegates authority for building symbols pub.foo.org. to NS below which holds the master file for zone pub



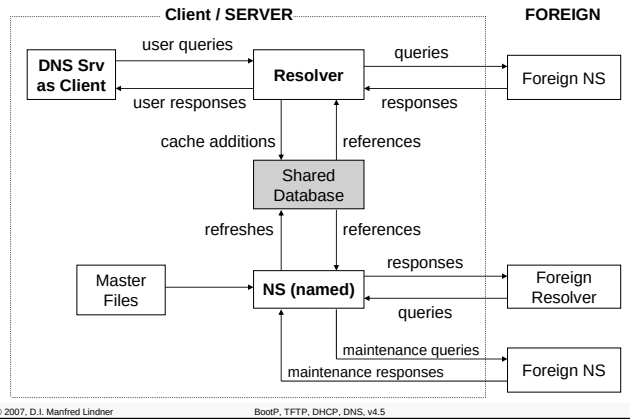
BIND Principles (Client)



BIND Principles (Server)



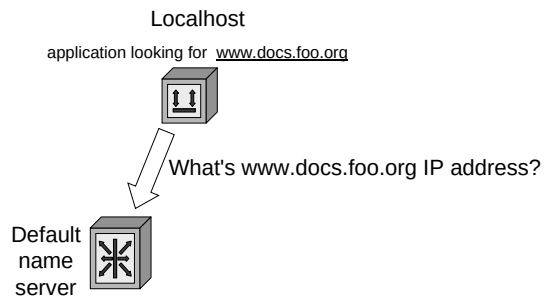
BIND Principles (Server complete)



BIND Principles

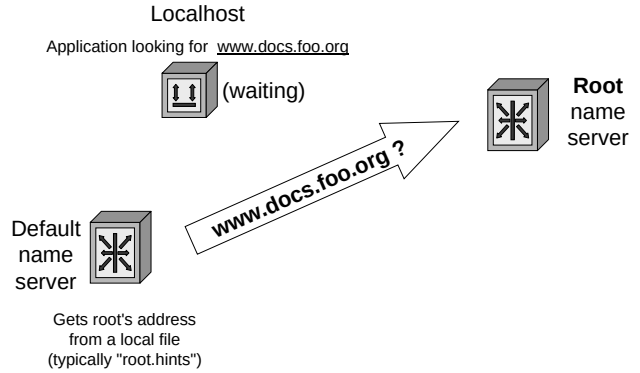
- **applications running on a client use the resolver to send *name resolution queries* to a name server**
  - each client-host requires a preconfigured IP address of one (or several) *default name server(s)*
- **a name server responses to this query after retrieving the requested data either**
  - by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



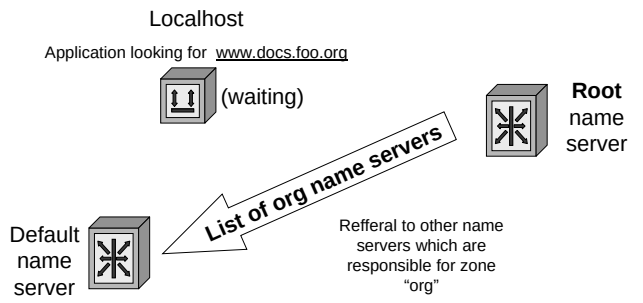
 BootP, TFTP, DHCP, DNS, v4.5 73

Iterative Queries (2)



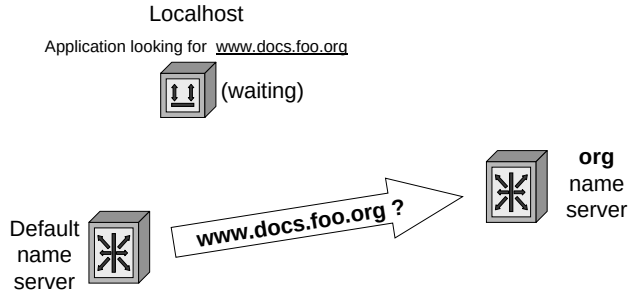
 BootP, TFTP, DHCP, DNS, v4.5 74

Iterative Queries (3)



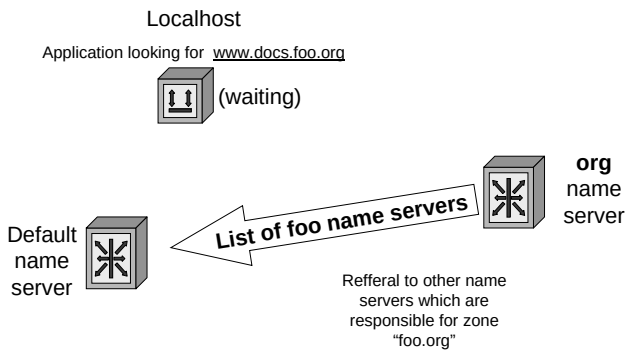
 BootP, TFTP, DHCP, DNS, v4.5 75

Iterative Queries (4)

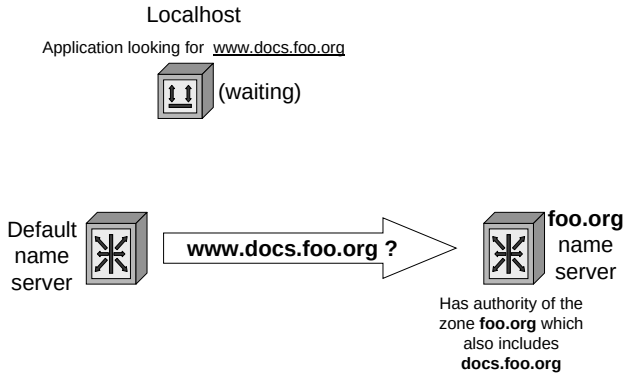


 BootP, TFTP, DHCP, DNS, v4.5 76

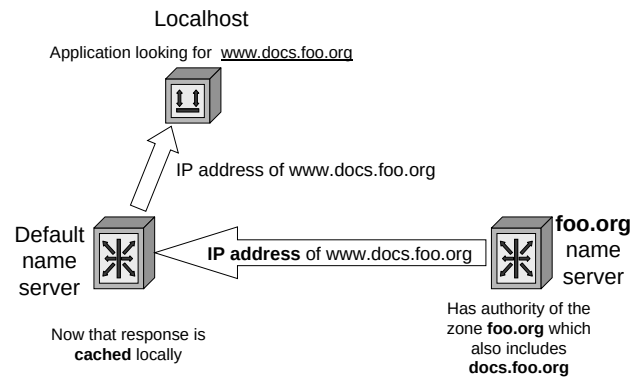
Iterative Queries (5)



Iterative Queries (6)



Iterative Queries (7)



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 domain**
  - 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 own master files 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**
- **DNS**
  - Introduction
  - Bind and DNS Servers
  - Resource Records
  - DNS Protocol

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

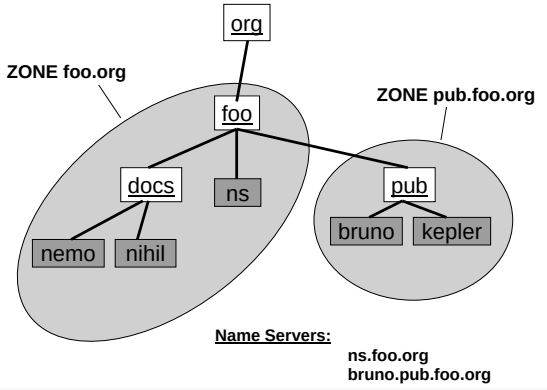
Types of Resource Records (4)

- **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
                        3600            ;retry time
                        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
ns              IN      A       216.32.78.1
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
```

Records describing zone .foo.org. = @

Delegation for the zone pub.foo.org.

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

Datenkommunikation 384.081 - SS 2007

L12 - BootP, TFTP, DHCP, DNS

Agenda

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

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 empty in certain cases

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)															

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

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

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)

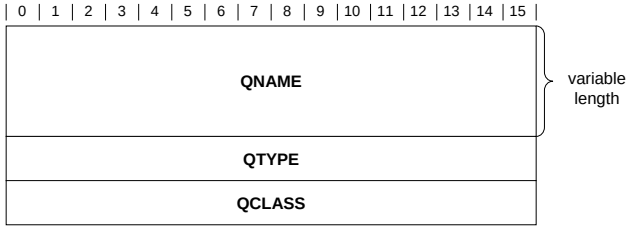
- **RCODE**
  - Response Code
  - 0 ... *no error*
  - 1.... *format error* - the NS was not able to interpret the query
  - 2 ... *server failure* - the NS has problems
  - 3 ... *name error* - an authoritative NS signals that the requested domain does not exist
  - 4 ... *not implemented* - the NS 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

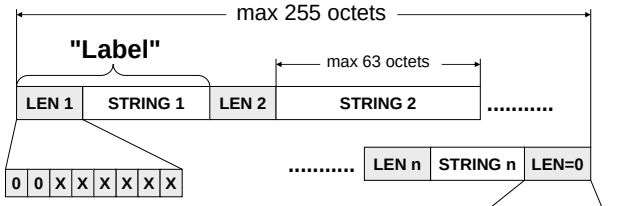
The question section contains QDCOUNT entries, each of the following format:



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)

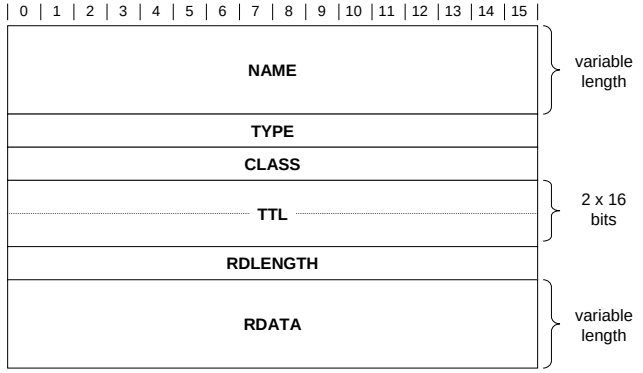
Domain Names in Messages



- 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

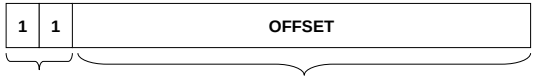
Resource Record Fields (2)

- **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

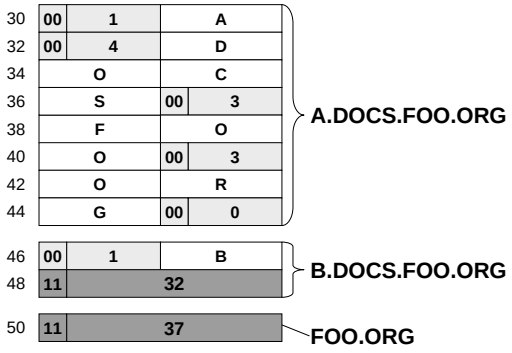
Message Compression

- **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:



Message Compression Example



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