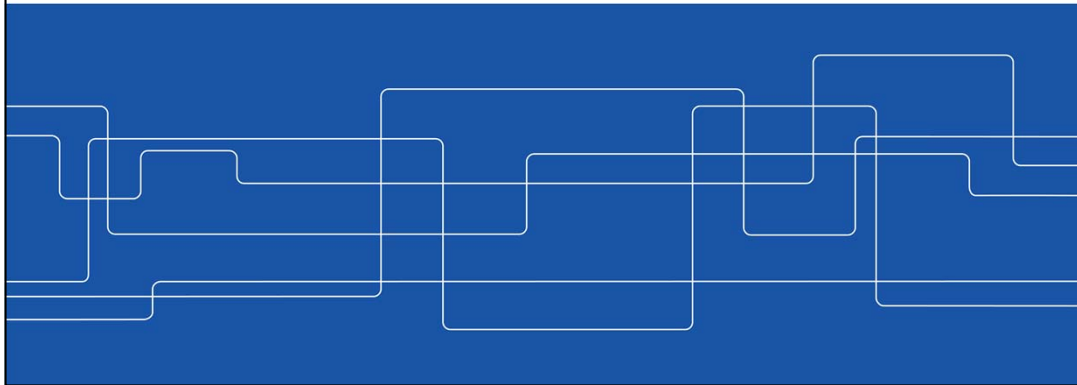




# IK1550 & IK1552 Internetworking/Internetteknik

prof. Gerald Q. Maguire Jr. <http://web.ict.kth.se/~maguire>

School of Information and Communication Technology (ICT), KTH Royal Institute of Technology  
IK1550/IK1552 Spring 2014, Period 4 2014.03.26 © 2014 G. Q. Maguire Jr. All rights reserved.





## Module 4: UDP and friends

Lecture notes of G. Q. Maguire Jr.

For use in conjunction with James F. Kurose and Keith W. Ross, *Computer Networking: A Top-Down Approach*, Fifth Edition, Pearson, 2010.



## Outline

- UDP
- Socket API
- BOOTP
- DHCP
- DNS, DDNS



## Transport layer protocols

The transport layer is responsible for end-to-end delivery of entire messages

- uses Protocol Port and/or Port Number to demultiplex the incoming packet so that it can be delivered to a specific process
- Segmentation and Reassembly  
Divides a message into transmittable segments and reassemble them at the receive
- Connection control - for connection-oriented transport protocols
- End-to-end **Flow** Control (in contrast to link level flow control)
- End-to-end **Error** Control (in contrast to link level error control)





## Main Transport layer protocols

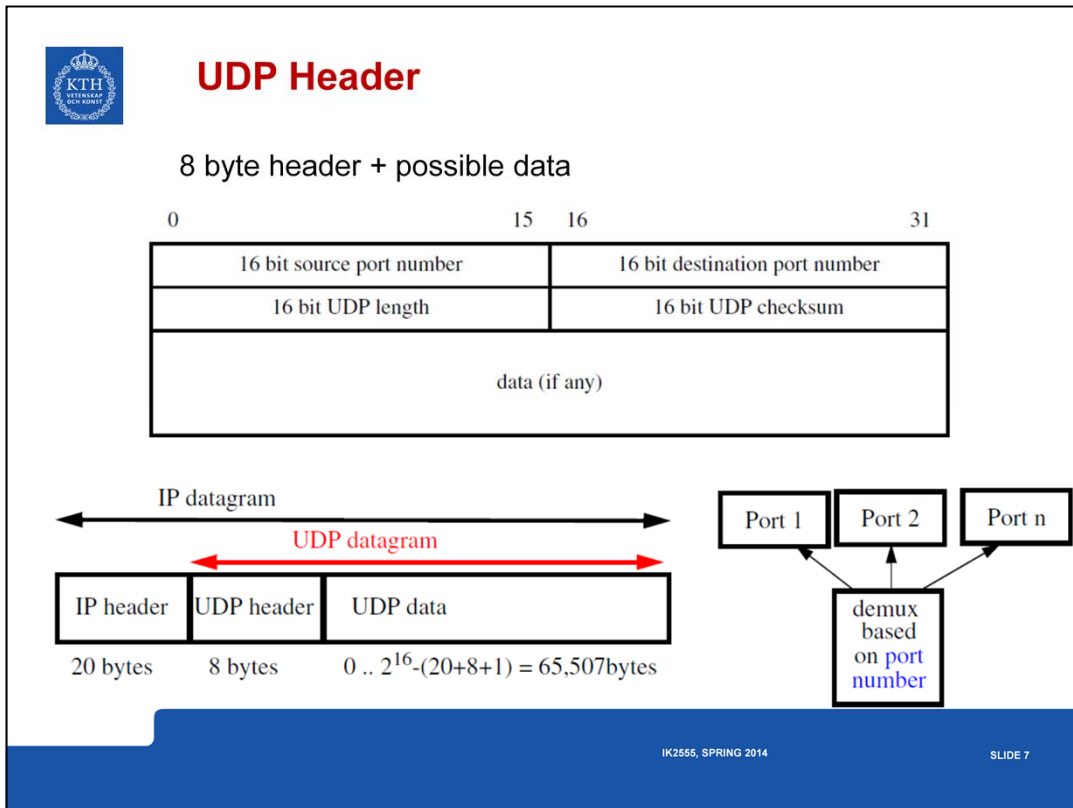
Three main transport layer protocols:

- User Datagram Protocol (UDP) <<< today's topic  
Connectionless **unreliable** service
- Transmission Control Protocol (TCP)  
Connection-oriented **reliable** stream service
- Stream Control Transmission Protocol (STCP)  
a modern transmission protocol with many facilities  
which the user can chose from



## User Datagram Protocol (UDP)

- Datagram-oriented transport layer protocol
- Provides connectionless unreliable service
- No reliability guarantee
- Checksum covers both header and data, end-to-end, but optional
  - if you care about your data you should be doing end-to-end checksums or using an even stronger error detection (e.g., MD5).
- An UDP datagram is silently discarded if checksum is in error.
  - No error message is generated
- Lots of UDP traffic is only sent locally
  - thus the reliability is comparable to the error rate on the local links. (see Stevens, Vol. 1, figure 11.5, pg. 147 for comparison of Ethernet, IP, UDP, and TCP checksum errors)
- Each output operation results in one UDP datagram, which causes one IP datagram to be sent
- Applications which use UDP: DNS, TFTP, BOOTP, DHCP, SNMP, NFS, VoIP, etc.
  - An advantage of UDP is that it is a base to build your own protocols on
  - Especially if you don't need reliability and in order delivery of lots of data



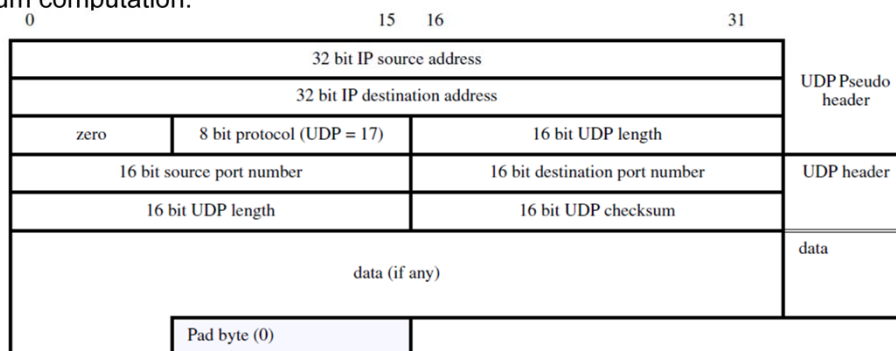


## UDP Checksum and Pseudo-Header

UDP checksum covers more info than is present in the UDP datagram alone: **pseudo-header** and **pad byte (0)** {to even number of 16 bit words}.

Propose: to verify the UDP datagram reached its correct destination: **right port number at the right IP address**.

Pseudo-header and pad byte are **not transmitted** with the UDP datagram, only used for checksum computation.





## Reserved and Available UDP Port Numbers

	keyword	UNIX keyword	Description
0	reserved		
7	ECHO	echo	Echo
9	DISCARD	discard	Discard == sink null
11	USERS	systat	Active users
13	DAYTIME	daytime	Daytime
15	-	netstat	Network status program
17	QUOTE	qotd	Quote of the day
19	CHARGEN	chargen	Character generator
37	TIME	time	Time server
39	RLP	rlp	Resource Location Protocol
42	NAMESERVER	name	Host Name Server
43	NICNAME	whois	Who is
53	DOMAIN	domain	Domain Name Server
67	BOOTPS	bootps	Bootstrap Protocol Server
68	BOOTPC	bootpc	Bootstrap Protocol Client
69	TFTP	tftp	Trivial File Transfer Protocol
88	KERBEROS	kerberos5	Kerberos v5 kdc
111	SUNRPC	sunrpc	SUN Remote Procedure Call (portmap)
123	NTP	ntp	Network Time Protocol
137	netbios_ns	netbios_ns	NetBIOS name service
138	netbios_dgm	netbios_dgm	NetBIOS Datagram Service
139	netbios_ssn	netbios_ssn	NetBIOS Session Service
161	snmp		Simple Network Management Protocol Agent
162	snmp-trap		Simple Network Management Protocol Traps
512	biff		mail notification
513	who		remote who and uptime
514	syslog		remote system logging
517	talk		conversation
520	route		routing information protocol
525	timed		remote clock synchronization
533	netwall		Emergency broadcasting
750	kerberos	kerberos	Kerberos (server)
6000 + display number		X11 server	
7000		X11 font server	



## Port numbers in three groups

Range	Purpose
0 .. 1023	System (Well-Known) Ports†
1024 .. 49151	User (Registered) Ports
49152 .. 65535	Dynamic and/or Private Ports

†Roughly 300 well know port numbers remain unassigned and 38 reserved

Roughly 26k registered port numbers remain unassigned and 9 reserved


‘For the purpose of providing services to **unknown** callers, a service contact port is defined. This list specifies the port used by the server process as its contact port. The contact port is sometimes called the "well-known port".’

<http://www.iana.org/assignments/port-numbers>

Linux chooses the local port to use for TCP and UDP traffic from this range:

```
$ cat /proc/sys/net/ipv4/ip_local_port_range
```

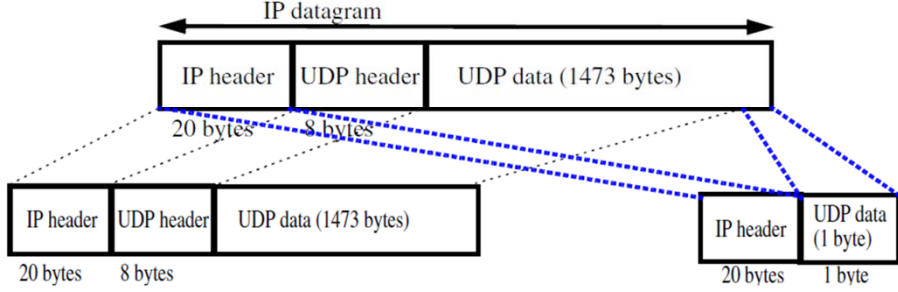
```
32768 61000
```

 **MTU and Datagram Fragmentation**

If datagram size > MTU, perform fragmentation.

- At sending host or at intermediate router (IPv4).
- Reassembled only at final destination.

Example: 1501 (20 + 8 + 1473 data) on Ethernet (MTU=1500):



Note there is no UDP header in the second fragment.  
Therefore, a frequent operation is to compute the path MTU before sending anything else. (see RFC 1191 for the table of common MTUs)

IK2555, SPRING 2014 SLIDE 11

J. C. Mogul and S. E. Deering, 'Path MTU discovery', *Internet Request for Comments*, vol. RFC 1191 (Draft Standard), Nov. 1990 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1191.txt>

J. McCann, S. Deering, and J. Mogul, 'Path MTU Discovery for IP version 6', *Internet Request for Comments*, vol. RFC 1981 (Draft Standard), Aug. 1996 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1981.txt>

B. Black and K. Kompella, 'Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol', *Internet Request for Comments*, vol. RFC 3988 (Experimental), Jan. 2005 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc3988.txt>

M. Mathis and J. Heffner, 'Packetization Layer Path MTU Discovery', *Internet Request for Comments*, vol. RFC 4821 (Proposed Standard), Mar. 2007 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc4821.txt>



## Fragmentation Required

If datagram size > MTU, DF (Don't Fragment) in IP header is on, then the router sends ICMP Unreachable Error.

⇒ this can be used to find Path MTU.





## Interaction between UDP and ARP

With ARP cache empty, send a UDP datagram with 8192 bytes onto an Ethernet

- 8192 bytes > ethernet MTU, therefore 6 fragments are created by IP
- if ARP cache is empty, first fragment causes ARP request to be sent
- This leads to two timing questions:

1. Are the remaining fragments sent before the ARP reply is received?
2. What does ARP do with multiple packets to the same destination while waiting for a reply?

### Example under BSD

Bsdi% arp -a ARP cache is empty

Bsdi% sock -u -i -n1 -w8192 svr4 discard

```

1      0.0      arp who-has svr4 tell bsdi
2      0.001234 (0.0012)  arp who-has svr4 tell bsdi
3      0.001941 (0.0007)  arp who-has svr4 tell bsdi
4      0.002775 (0.0008)  arp who-has svr4 tell bsdi
5      0.003495 (0.0007)  arp who-has svr4 tell bsdi
6      0.004319 (0.0008)  arp who-has svr4 tell bsdi
7      0.008772 (0.0045)  arp reply svr4 is-at 0:0:c0:c2:9b:26
8      0.009911 (0.0011)  arp reply svr4 is-at 0:0:c0:c2:9b:26
9      0.011127 (0.0012)  bsdi > svr4: (frag 10863:800@7400)
10     0.011255 (0.0001)  arp reply svr4 is-at 0:0:c0:c2:9b:26
11     0.012562 (0.0013)  arp reply svr4 is-at 0:0:c0:c2:9b:26
12     0.013458 (0.0009)  arp reply svr4 is-at 0:0:c0:c2:9b:26
13     0.014526 (0.0011)  arp reply svr4 is-at 0:0:c0:c2:9b:26
14     0.015583 (0.0011)  arp reply svr4 is-at 0:0:c0:c2:9b:26

```



## on a BSDI system:

- each of the additional (5) fragments caused an ARP request to be generated  
this violates the Host Requirements RFC - which tries to prevent [ARP flooding](#) by limiting the maximum rate to 1 per second
- when the ARP reply is received the last fragment is sent  
Host Requirements RFC says that ARP should save at least one packet and this should be the latest packet
- unexplained anomaly: the System Vr4 system sent 7 ARP replies back!
- no ICMP "time exceeded during reassembly" message is sent
  - BSD derived systems - never generate this error! It does set the timer internally and discard the fragments, but never sends an ICMP error.
  - fragment 0 (which contains the UDP header) was not received - so there is no way to know which process sent the fragment; thus unless fragment 0 is received - you are not required to send an ICMP "time exceeded during reassembly" error.

[Not just a fluke](#) (i.e., a rare event)

- The same error occurs even if you don't have fragmentation - simply sending multiple UDP datagrams rapidly when there is no ARP entry is sufficient!
- NFS sends UDP datagrams whose length just exceeds 8192 bytes
- NFS will timeout and resend
- however, there will always be this behavior - if the ARP cache has no entry for this destination!



## Still a problem?

UDP with 8192 payload to echo port as seen on SuSE 9.2 linux 2.6.8-24:

No.	Time	Source	Destination	Protocol	Info
37	3.020002	172.16.33.16	Broadcast	ARP	Who has 172.16.33.5? Tell 172.16.33.16
38	3.021385	172.16.33.5	172.16.33.16	ARP	172.16.33.5 is at 00:40:8c:24:37:f4
39	3.021422	172.16.33.16	172.16.33.5	IP	Fragmented IP protocol (proto=UDP 0x11, off=4440)
40	3.021452	172.16.33.16	172.16.33.5	IP	Fragmented IP protocol (proto=UDP 0x11, off=5920)
41	3.021480	172.16.33.16	172.16.33.5	IP	Fragmented IP protocol (proto=UDP 0x11, off=7400)

3.021385-3.020002=.001383 sec.  $\Rightarrow$  1.383ms for the ARP reply

All but the last 3 fragments are dropped! Including the initial echo request packet -- so in the fragments that do arrive you don't know who they are for -- because the first fragment was lost!



## With an even larger UDP packet

I removed the arp cache entry with: `/sbin/arp -i eth1 -d 172.16.33.5`

When sending 65500 bytes of UDP payload -- it loses many packets (in fact all but the last 3 fragments)!!!

No. Time Source Destination Protocol Info

```
36 4.342158 172.16.33.16 Broadcast ARP Who has 172.16.33.5? Tell 172.16.33.16
37 4.342875 172.16.33.5 172.16.33.16 ARP 172.16.33.5 is at 00:40:8c:24:37:f4
38 4.342906 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=62160)
39 4.342932 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=63640)
40 4.342986 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=65120)
```

**With the entry in the ARP cache get:**

No. Time Source Destination Protocol Info

```
35 5.118063 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=1480)
36 5.118095 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=2960)
37 5.118115 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=4440)
38 5.118214 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=5920)
39 5.118328 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=7400)
....
75 5.122787 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=60680)
76 5.122877 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=62160)
77 5.122999 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=63640)
78 5.123122 172.16.33.16 172.16.33.5 IP Fragmented IP protocol (proto=UDP 0x11, off=65120)
```

The initial UDP Echo request is still lost! The key parameter is `/proc/sys/net/ipv4/neigh/ethX/unres_qlen` where X is the interface (i.e., eth0, eth1, ...)  
-- the default value is 3.



## Maximum UDP Datagram size

theoretical limit: 65,535 bytes - due to (IP's) 16-bit total length field  
with 20 bytes of IP header + 8 bytes of UDP header  $\Rightarrow$  65,507 bytes of user data

two limits:

- sockets API limits size of send and receive buffer; generally 8 kbytes, but you can call a routine to change this (modern linux kernels auto tune this buffer size)
- TCP/IP implementation - Stevens found various limits to the sizes - even with loopback interface (see Stevens, Vol. 1, pg. 159)

Hosts are required to handle at least 576 byte IP datagrams, thus lots of protocols limit themselves to 512 bytes or less of data:

DNS, TFTP, BOOTP, and SNMP



## Datagram truncation

What if the application is not prepared to read the datagram of the size sent?

Implementation dependent:

- traditional Berkeley: silently truncate
- 4.3BSD and Reno: **can** notify the application that the data was truncated
- SVR4: excess data returned in subsequent reads - application is not told that this all comes from one datagram
- TLI: sets a flag that more data is available, subsequent reads return the rest of the datagram



## Socket API

```
int socket(int domain, int type, int protocol);
    creates an endpoint description that you can use to send and receive network traffic
int bind(int sockfd, struct sockaddr *my_addr, socklen_t addrlen);
    binds a socket to the local address and port: my_address
int connect(int sockfd, const struct sockaddr *serv_addr, socklen_t addrlen);
    connect the local socket to a remote socket
int listen(int s, int backlog);
    indicates socket is willing to accept connections & limits the queue of pending connections
int accept(int s, struct sockaddr *addr, socklen_t *addrlen);
    accepted the first connection request from the queue and connects it to the socket
connection oriented sending and receiving:
    ssize_t send(int s, const void *buf, size_t len, int flags);
    ssize_t recv(int s, void *buf, size_t len, int flags);
datagram sending and receiving:
    ssize_t sendto(int s, const void *buf, size_t len, int flags, const struct sockaddr *to, socklen_t
    tolen);
    ssize_t recvfrom(int s, void *buf, size_t len, int flags, struct sockaddr *from, socklen_t
    *fromlen);
int close(int fd) and int shutdown(int s, int how) - end it all!
```



## Windows Socket API

Have to start up the Windows TCP/IP stack:

```
int WSAStartup( _In_ WORD wVersionRequested,  
              _Out_ LPWSADATA lpWSAData);
```

To cleanly shutdown the stack:

```
WSACleanup();
```

There are additional WSA functions, for example for asynchronous I/O





## Learning about Socket programming

For examples of using the socket API and networking programming see :

- Richard Stevens, *UNIX Network Programming, Volume 1, Second Edition: Networking APIs: Sockets and XTI*, Prentice Hall, 1998, ISBN 0-13-490012-X  
<http://www.kohala.com/start/unpv12e.html>
- Brian "Beej" Hall, "Beej's Guide to Network Programming: Using Internet Sockets", Jorgensen Publishing, October 20, 2011, 142 pages  
<http://beej.us/guide/bgnet/>

Two additional socket functions for controlling various properties of sockets are:

- int `getsockopt`(int s, int level, int optname, void \*optval, socklen\_t \*optlen);
- int `setsockopt`(int s, int level, int optname, const void \*optval, socklen\_t optlen);

For example, the options `SO_SNDBUF` and `SO_RCVBUF` - control the size of the sending buffer and the receiver buffer.



## Simple UDP client

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define bigBufferSize 8192
#define destination_host "172.16.33.5"
main (int argc, char **argv);
{
  int client_socket_fd;
  struct sockaddr_in server_addr;
  char bigBuffer[bigBufferSize];
  int sendto_flags=0;

  /* Socket to client, server */
  /* server's address */
  /* buffer of data to send as payload */

  /* create a UDP socket */
  if ((client_socket_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
    perror("Unable to open socket"); exit(1);
  }
  /* initialize the server address structure */
  memset((char*)&server_addr, 0, sizeof(server_addr));
  server_addr.sin_family=AF_INET;
  server_addr.sin_port=htons(9);
  /* 9 is the UDP port number for Discard */
  if (inet_aton(destination_host, (struct sockaddr*)&server_addr.sin_addr) == 0) {
    fprintf(stderr, "could not get an address for: %s", destination_host); exit(1);
  }
  if ((sendto(client_socket_fd, bigBuffer, bigBufferSize,
    sendto_flags, (struct sockaddr*)&server_addr, sizeof(server_addr))) == -1) {
    perror("Unable to send to socket"); close(client_socket_fd); exit(1);
  }
  close(client_socket_fd);
  /* close the socket */
  exit(0);
}

```

IK2555, SPRING 2014

SLIDE 22



## UDP server design

Stevens, Vol, 1, pp. 162-167 discusses how to program a UDP server

You can often determine what IP address the request was sent to (i.e., the destination address):

- for example: thus ignoring datagrams sent to a broadcast address

You can limit a server to a given incoming IP address:

- thus limiting requests to a given interface

You can limit a server to a given foreign IP address and port:

- only accepting requests from a given foreign IP address and port #

Multiple recipients per port (for implementations with multicasting support)

- setting `SO_REUSEADDR` socket option  $\Rightarrow$  each process gets a copy of the incoming datagram

Note: limited size input queue to each UDP port, can result in silent discards without an ICMP message being sent back (since `OS discarded`, not the network!)



## UDP listener example

```

#include <stdio.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>
#define bigBufferSize 8192
#define my_port 52000
#define destination_host "127.0.0.1"
main(int argc, char **argv)
{
    int client_socket_fd; /* Socket to client, server */
    struct sockaddr_in client_addr; /* client's address */
    struct sockaddr_in other_addr; /* other party's address */
    int other_addr_len;
    char bigBuffer[bigBufferSize];
    int sendto_flags=0;

    /* create a UDP socket */
    if ((client_socket_fd = socket(AF_INET, SOCK_DGRAM, IPPROTO_UDP)) == -1) {
        perror("Unable to open socket"); exit(1); }
    /* initialize address structure */
    memset((char*)&client_addr, 0, sizeof(client_addr));
    client_addr.sin_family=AF_INET;
    client_addr.sin_port=htons(my_port);
    client_addr.sin_addr.s_addr = htonl(INADDR_ANY);
    if (bind(client_socket_fd, (struct sockaddr*)&client_addr, sizeof(client_addr))== -1) {
        close(client_socket_fd); exit(1); }
    if ((recvfrom(client_socket_fd, bigBuffer, bigBufferSize, sendto_flags, (struct sockaddr*)&other_addr, &other_addr_len))
    == -1) {
        perror("Unable to receive from socket"); close(client_socket_fd); exit(1); }
    printf("Received packet from %s:%d\nData: %s\nString length=%d\n", inet_ntoa(other_addr.sin_addr),
    ntohs(other_addr.sin_port), bigBuffer, strlen(bigBuffer));
    close(client_socket_fd); exit(0);}

```

IK2555, SPRING 2014

SLIDE 24



## Changed the client

Changing the following:

```
#define destination_host "127.0.0.1"  
#define my_port 52000  
server_addr.sin_port=htons(my_port);
```

Adding some content to the bigBuffer:

```
sprintf(bigBuffer, "This is a simple test string to be sent to the other party\n");
```

Sending only as much of the buffer as necessary:

```
if ((sendto(client_socket_fd, bigBuffer, strlen(bigBuffer),  
sendto_flags, (struct sockaddr*)&server_addr, sizeof(server_addr))) == -1) {...}
```

Results in the listener outputting:

```
Received packet from 127.0.0.1:1260  
Data: This is a simple test string to be sent to the other party  
String length=59
```



## Building a UDP packet from scratch

/\* simple example of building a UDP packet from scratch, based on the program: PingPong - 970621 by Willy TARREAU  
<tarreau@aemiaif.ibp.fr>

[http://www.insecure.org/splits/inetd.internal\\_udp\\_ports.DOS.attack.html](http://www.insecure.org/splits/inetd.internal_udp_ports.DOS.attack.html)

As this program uses RAW sockets, you must be root to run it \*/

```
#include <netinet.h>
#include <netinet/ip.h>
#include <netinet/udp.h>
#include <netdb.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include <ctype.h>
#include <errno.h>
struct sockaddr addrfrom;
struct sockaddr addrto;
int s;
u_char outpack[65536];
struct iphdr *ip;
struct udphdr *udp;
main(int argc, char **argv) {
    struct sockaddr_in *from;
    struct sockaddr_in *to;
    struct protoent *proto;
    int i;
    char *src,*dest;
    int srctp, destp;
    int packetsize,datasize;
    if (argc!=5) {printf(stderr,"Usage: %s src_addr src_port dst_addr dst_port\n", argv[0]);
        printf(stderr,"src_addr and dst_addr must be given as IP addresses (xxx.xxx.xxx.xxx)\n");
        exit(2);}
    src=argv[1]; srctp=atoi(argv[2]); dest=argv[3]; destp=atoi(argv[4]);
```



```

if (!(proto = getprotobyname("raw"))) {perror("getprotobyname(raw));exit(2);}

if ((s = socket(AF_INET, SOCK_RAW, proto->p_proto)) < 0) {perror("socket");exit(2);}
memset(&addrfrom, 0, sizeof(struct sockaddr));
from = (struct sockaddr_in *)&addrfrom;
from->sin_family = AF_INET;
from->sin_port=htons(srcp);
if (!inet_aton(src, &from->sin_addr)) {fprintf(stderr,"Incorrect address for 'from': %s\n",src);exit(2); }

memset(&addrto, 0, sizeof(struct sockaddr));
to = (struct sockaddr_in *)&addrto;
to->sin_family = AF_INET;
to->sin_port=htons(destp);
if (!inet_aton(dest, &to->sin_addr)) {fprintf(stderr,"Incorrect address for 'to': %s\n",dest);exit(2); }

packetsize=0; /* build a UDP packet from scratch */
ip=(struct iphdr *)outpack; /* IPv4 */
ip->version=4; /* IP header length: 5 words */
ip->ihl=5; /* no special type of service */
ip->tos=0; /* no ID */
ip->id=0; /* not a fragment - so there is no offset */
ip->frag_off=0; /* TTL = 64 */
ip->ttl=0x40;
if (!(proto = getprotobyname("udp"))) {perror("getprotobyname(udp)"); exit(2);}
ip->protocol=proto->p_proto; /* null checksum, will be automatically computed by the kernel */
ip->check=0; /* set source and destination addresses */
ip->saddr=from->sin_addr.s_addr; /* end of ip header */
ip->daddr=to->sin_addr.s_addr;

```



```
packetsize+=ip->ihl<<2;
/* udp header */
udp=(struct udphdr*)((int)outpack + (int)(ip->ihl<<2));
udp->source=htons(srcp);
udp->dest=htons(destp);
udp->check=0;
packetsize+=sizeof(struct udphdr);
/* ignore UDP checksum */
/* end of udp header */
/* add data to UDP payload if you want: */

for (datasize=0;datasize<8;datasize++) {
    outpack[packetsize+datasize]='A'+datasize;
}
packetsize+=datasize;
udp->len=htons(sizeof(struct udphdr)+datasize);
ip->tot_len=htons(packetsize);

if (sendto(s, (char *)outpack, packetsize, 0, &addrto, sizeof(struct sockaddr))!=-1) {perror("sendto");
exit(2);}
printf("packet sent !\n");
close(s);
exit(0);}
```

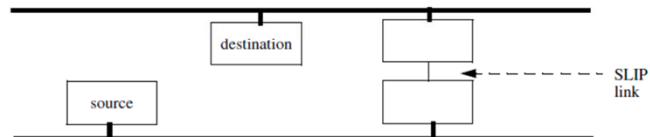




## ICMP Source Quench Error

Since UDP has no flow control, a node could receive datagrams faster than it can process them. In this situation the host **may** send an ICMP source quench.

Note: “**may** be generated” - it is **not** required to generate this error  
Stevens (Vol. 1, pp. 160-161) gives the example of sending 100 1024-byte datagrams from a machine on an Ethernet via a router and SLIP line to another machine:



simplified from Stevens, Vol. 1, inside cover

- SLIP link is ~1000 times slower than the ethernet
- 26 datagrams are transmitted, then a source quench is sent for each successive datagram



- the router gets all 100 packets, before the first has been sent across the link!  
the Router Requirements RFC - says that routers should not generate source quench errors, since it just consumes network bandwidth and it is an ineffective and unfair fix for congestion
  - In any case, the sending program never responded to the source quench errors!
    - BSD implementations ignore received source quenches if the protocol is UDP
    - the program finished before the source quench was received!
- Thus if you want reliability you have to build it in and do end-to-end flow control, error checking, and use (and thus wait for) acknowledgements.



## No error control

Since UDP has no error control, the sender has to take responsibility for sending the datagram again if this datagram must be delivered.

But how does the sender know if a datagram was successfully delivered?

- Unless the receiver sends a reply (or does some action due to receiving a given datagram) the sender will **not** know!  
[loss]
- Note that without some additional mechanism - the sender doesn't know if the datagram was delivered multiple times!  
[duplicates]

If you want reliability you have to build your own protocol on top of UDP to achieve it. This includes deciding on your own retransmission scheme, timeouts, etc.



## BOOTP: Bootstrap Protocol (RFC 951)

Although you can figure out who you are, i.e., your IP address, via RARP - many machines want more information.

BOOTP requests and answer are sent via UDP (port 67 server; port 68 client)

- so it is easy to make a user space server
- the client (who wants the answer) need not have a full TCP/IP, it can simply send what **looks** like a UDP datagram with a BOOTP request (see Stevens, Vol. 1, figure 16.2, pg. 2).

Opcode (1=request, 2=reply)	hardware type(1=ethernet)	hardware address length (6 for ethernet)	hop count
transaction ID			
number of seconds		unused	
client IP address			
your IP address			
server IP address			
gateway IP address			
client hardware address (16 bytes of space)			
server hostname (64 bytes)			
Boot file name (128 bytes)			
Vendor specific information (64 bytes)			

IK2555, SPRING 2014

SLIDE 32

W. J. Croft and J. Gilmore, 'Bootstrap Protocol', *Internet Request for Comments*, vol. RFC 951 (Draft Standard), Sep. 1985 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc951.txt>



## BOOTP (continued)

When a request is sent as an IP datagram:

- if client does not know its IP address it uses 0.0.0.0
- if it does not know the server's address it uses 255.255.255.255
- if the client does not get a reply, it tries again in **about 2** sec.



## Vendor specific information (RFC 1497 and RFC1533)

if this area is used the first 4 bytes are: IP address 99.130.83.99 this is called the “**magic cookie**”, Then the rest of the area is a list of items, possibly including:

- Pad (tag=0);
- Subnet mask (tag=1);
- Time offset (tag=2);
- List of IP addresses of Gateways (tag=3);
- Time server's IP address (tag=4);
- Name Server (tag=5);
- Domain Name Server (tag=6);
- LOG server (tag=7); ...
- LPR server (tag=9); ...
- this Host's name (tag=12);
- Boot file size (tag=13); ...
- Domain name (tag=15); ...
- End (tag=255)

IK2555, SPRING 2014

SLIDE 34

W. J. Croft and J. Gilmore, 'Bootstrap Protocol', *Internet Request for Comments*, vol. RFC 951 (Draft Standard), Sep. 1985 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc951.txt>

P. A. Prindeville, 'BOOTP vendor information extensions', *Internet Request for Comments*, vol. RFC 1048, Feb. 1988 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1048.txt>

J. K. Reynolds, 'BOOTP vendor information extensions', *Internet Request for Comments*, vol. RFC 1084, Dec. 1988 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1084.txt>

J. Reynolds, 'BOOTP Vendor Information Extensions', *Internet Request for Comments*, vol. RFC 1395 (Draft Standard), Jan. 1993 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1395.txt>

J. Reynolds, 'BOOTP Vendor Information Extensions', *Internet Request for Comments*, vol. RFC 1497 (Draft Standard), Aug. 1993 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1497.txt>

S. Alexander and R. Droms, 'DHCP Options and BOOTP Vendor Extensions', *Internet Request for Comments*, vol. RFC 1533 (Proposed Standard), Oct. 1993 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1533.txt>



## DHCP: Dynamic Host Configuration Protocol (RFC 1531)

Extends the Vendor specific options area by 312 bytes.

This protocol is designed to make it easier to allocate (and reallocate) addresses for clients. DHCP defines:

- **Requested IP Address** - used in client request (DHCPDISCOVER) to request that a particular IP address (tag=50)
- **IP Address Lease Time** - used in a client request (DHCPDISCOVER or DHCPREQUEST) to request a lease time for the IP address. In a server reply (DHCPOFFER), specific lease time offered. (tag=51)
- **Option Overload** - used to indicate that the DHCP "sname" or "file" fields are being overloaded by using them to carry DHCP options. A DHCP server inserts this option if the returned parameters will exceed the usual space allotted for options, i.e., it uses the sname and file fields for another purpose! (tag=52)

IK2555, SPRING 2014

SLIDE 35

R. Droms, 'Dynamic Host Configuration Protocol', *Internet Request for Comments*, vol. RFC 1531 (Proposed Standard), Oct. 1993 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1531.txt>



## DHCP (continued)

- **DHCP Message Type** - the type of the DHCP message (tag=53)
- **Server Identifier** - used in DHCPOFFER and DHCPREQUEST (optionally in DHCPACK and DHCPNAK) messages. Servers include this in the DHCPOFFER to allow the client to distinguish **between** lease offers. DHCP clients indicate which of several lease offers is being accepted by including this in a DHCPREQUEST message. (tag=54)
- **Parameter Request List** - used by a DHCP client to request values for specified configuration parameters. The client **may** list options in order of preference. The DHCP server **must** try to insert the requested options in the order requested by the client. (tag=55)





## DHCP (continued)

- **Message** - used by a server to provide an error message to client in a DHCPNAK message in the event of a failure. A client may use this in a DHCPDECLINE message to indicate the reason **why** the client declined the offered parameters. (tag=56)
- **Maximum DHCP Message Size** - specifies the maximum length DHCP message that it is willing to accept. A client may use the maximum DHCP message size option in DHCPDISCOVER or DHCPREQUEST messages, but should not use the option in DHCPDECLINE messages. (tag=57)
- **Renewal (T1) Time Value** - specifies the time interval from address assignment until the client transitions to the RENEWING state. (tag=58)
- **Rebinding (T2) Time Value** - specifies the time interval from address assignment until the client transitions to the REBINDING state. (tag=59)



## DHCP (continued)

- **Class-identifier** - used by DHCP clients to optionally identify the type and configuration of a DHCP client. Vendors and sites may choose to define specific class identifiers to convey particular configuration or other identification information about a client. Servers not equipped to interpret the class-specific information sent by a client **must** ignore it (although it may be reported). (tag=60)
- **Client-identifier** - used by DHCP clients to specify their unique identifier. DHCP servers use this value to index their database of address bindings. This value is expected to be unique for all clients in an administrative domain. (tag=61)



## DHCP's importance

- allows reuse of address, which avoids having to tie up addresses for systems which are not currently connected to the Internet
- avoids user configuration of IP address (avoids mistakes and effort)
- allows recycling of an IP address when devices are scrapped
- ...

### How big a problem is manual configuration?

A large site (such as DuPont Co. - a large chemical company) has > 70k IP addressable devices; or consider what happens if each of the > 1.4M Wal-Mart employees has an IP device

Product	Vendor	URL
Network Registrar	Cisco	<a href="http://www.cisco.com">http://www.cisco.com</a>
NetID	Nortel Networks	<a href="http://www.nortelnetworks.com">http://www.nortelnetworks.com</a>
Meta IP	CheckPoint	<a href="http://www.checkpoint.com">http://www.checkpoint.com</a>
VitalQIP	Alcatel-Lucent	<a href="http://enterprise.alcatel-lucent.com">http://enterprise.alcatel-lucent.com</a>



## DHCP performance problems

Most implementations of DHCP do a duplicate address detection (DAAD) test **after** they have picked an address to assign.

An alternative approach to speed up the DHCP process does the duplicate address detection process in the background (in advance) so that you will have a set of recently tested addresses to hand out:

Jon-Olov Vatn and Gerald Q. Maguire Jr., "The effect of using co-located care-of addresses on macro handover latency", Fourteenth Nordic Tele-traffic Seminar (NTS 14), August 18 - 20, 1998, Lyngby, Denmark.

<http://www.it.kth.se/~vatn/research/nts14-coloc.pdf>

The result is that a DHCP request can be answered in less than 100ms.



## Example of dhcpd.conf

```
### Managed by Linuxconf, you may edit by hand.
### Comments may not be fully preserved by linuxconf.
server-identifier dhcptest1;
default-lease-time 1000;
max-lease-time 2000;
option domain-name "3ctechnologies.se";
option domain-name-servers 130.237.12.2;
option host-name "s1.3ctechnologies.se";
option routers 130.237.12.2;
option subnet-mask 255.255.255.0;
subnet 130.237.12.0 netmask 255.255.255.0 {
    range 130.237.12.3 130.237.12.200;
    default-lease-time 1000;
    max-lease-time 2000;
}
subnet 130.237.11.0 netmask 255.255.255.0 {
    range 130.237.11.3 130.237.11.254;
    default-lease-time 1000;
    max-lease-time 2000;
}
```



## DHCP and DNS

- There is no dynamic host name assignment yet.
- Interaction between DHCP and DNS is needed.

For example: once a host is assigned an IP address the DNS should be updated dynamically:

- If the host hasn't got a name: it should assign a name along the IP address assignment (no DNS update is needed).
- If the host has already a name: the DNS should be dynamically updated once the host has gotten a new IP address from DHCP.

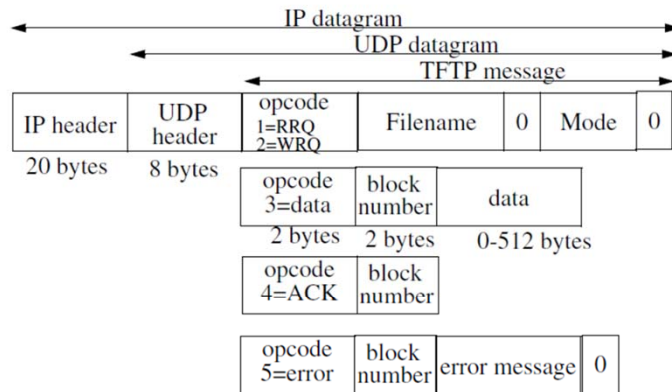
The IETF's Dynamic Host Configuration (dhc) Working group <http://www.ietf.org/html.charters/dhc-charter.html> is working on addressing the issues concerning interaction between DHCP and DNS.



## Trivial File Transfer Protocol (TFTP)

TFTP uses UDP (unlike FTP which uses TCP)

- simple and small
- requires only UDP, IP, and a device driver - easily fits in ROM
- a stop-and-wait protocol
- lost packets detected by timeout and retransmission
- Two operations:
  - Read Request (RRQ)
  - Write Request (WRQ) - for security reasons the file must already exist
- The TFTP server ("tftpd") is generally run setrooted (i.e., it only has access to its own directory) and with a special **user** and **group** ID - since there is no password or other protection of the access to files via TFTP!
- TFTP request is sent to the well known port number (69/udp)
- TFTP server uses an unused ephemeral port for its replies since a TFTP transfer can last for quite some time - it uses another port; thus freeing up the well known port for other requests

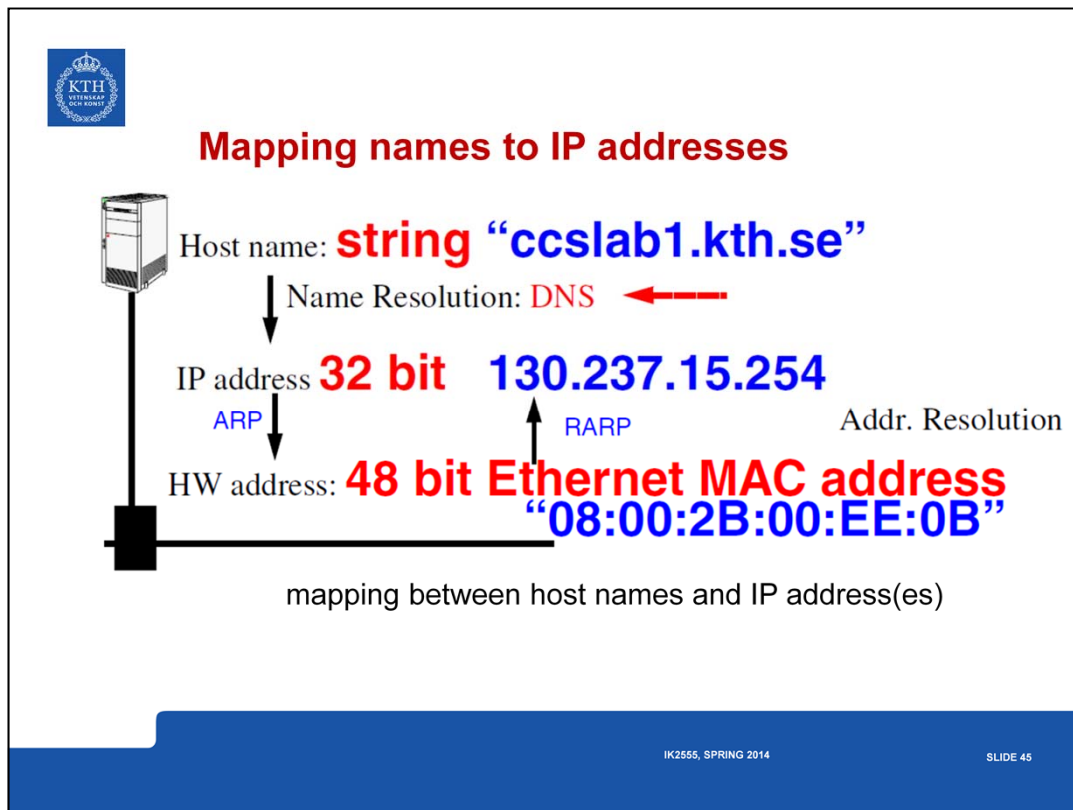


TFTP messages (see Stevens, Vol. 1, figure 15.1, pg. 210)

Filename and Mode (“netascii” or “octet”) are both N bytes sequences terminated by a null byte.

Widely used for bootstrapping diskless systems (such as X terminals) and for dumping the configuration of routers (this is where the write request is used)







## DNS: Domain Name Service (RFC 1034, RFC 1035)

- To make the network more user friendly
- Distributed database (with caching) providing:
  - hostname  $\Rightarrow$  IP address, IP address  $\Rightarrow$  hostname
  - mailbox  $\Rightarrow$  mail server
  - ...
- applications call a “resolver”
  - gethostbyname: hostname  $\Rightarrow$  IP address
  - gethostbyaddr: IP address  $\Rightarrow$  hostname
- Resolver’s contact name servers (see “/etc/resolv.conf”)
- DNS names:
  - domain name: list of labels from a root, i.e., www.imit.kth.se
  - Fully Qualified Name (FQDN): a domain name ending in “.” - there are no further labels
  - leaves are managed locally through delegation of authority (to a zone) **not** centrally; this allows scaling
  - if a name server does not know the answer it asks other name servers
  - every name server **must** know how to contact a root server
- Uses UDP (for query) and TCP (zone transfer and large record query)

IK2555, SPRING 2014

SLIDE 46

P. V. Mockapetris, ‘Domain names - concepts and facilities’, *Internet Request for Comments*, vol. RFC 1034 (INTERNET STANDARD), Nov. 1987 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1034.txt>

P. V. Mockapetris, ‘Domain names - implementation and specification’, *Internet Request for Comments*, vol. RFC 1035 (INTERNET STANDARD), Nov. 1987 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1035.txt>



## Zones

A zone is a subtree of the DNS tree which is managed separately.

Each zone must have multiple name servers:

- a **primary name server** for the zone
  - gets its data from disk files (or other stable store)
  - must know the IP address of one or more **root servers**
- one or more **secondary name servers** for the zone
  - get their data by doing a **zone transfer** from a primary
  - generally query their primary server every ~3 hours

To find a server you may have to walk the tree up to the root or possibly from the root down (but the later is **not friendly**).



## DNS Message format

0	16	31
Identification	Parameters	
Number of Questions	Number of Answers	
Number of authority	Number of Additional	
Question section		
...		
Answer section		
...		
Additional Information section		
...		



## DNS Parameters field

Bit or Parameter field	Meaning
0	Operation: 0=Query, 1=Response
1-4	Query type: 0=standard, 1=Inverse
5	Set if answer is authoritative
6	Set if answer is truncate
7	Set if answer is desired
8	Set if answer is available
9-11	reserved
12-15	Response Type: 0=No error, 1=Format error in query, 2=Server failure, 3=Name does not exist



## Internet's top level domains

(see Stevens, Vol. 1, figure 14.2, pg. 189)

Domain	Description
com	commercial organizations
edu	educational organizations
gov	other U.S. government organizations (see RFC 1811 for policies)
int	international organizations
mil	U.S. Military
net	networks
org	other organizations
arpa	special domain for address to name mappings, e.g., 5.215.237.130.in-addr.arpa
ae	United Arab Emirates
...	
se	Sweden
zw	Zimbabwe

Lots of interest in having subdomains of "com"

- companies registering product names, etc. - in some cases asking for 10s of addresses
- who gets to use a given name? problems with registered trade marks, who registered the name first, ... [How much is a name worth?]



## New top level domains

There is a proposed new set of top level domains and an increase in the number of entities which can assign domain names.

Generic Top Level Domains (gTLDs), November 16, 2000:

.aero	for the entire aviation community
.biz	for business purpose
.coop	for cooperatives
.info	unrestricted
.museum	for museums
.name	for personal names
.pro	for professionals

CORE (Council of Registrars) - operational organization composed of authorized Registrars for managing allocations under gTLDs.

WIPO provides arbitration concerning names:  
<http://arbitr.wipo.int/domains/gtld/newgtld.html>



## Domain registrars

Internet Corporation for Assigned Names and Numbers (ICANN) Accredited Registrars, the full list is at

<http://www.icann.org/registrars/accredited-list.html>

Even more registrars are on their way to being accredited and operating!





## Country Code Top-Level Domains (CCTLDs)

<http://www.iana.org/cctld/cctld-whois.htm>

For Sweden (SE) SE-DOM, the NIC is: <https://www.iis.se/>

Contact .SE Registry: [registry@iis.se](mailto:registry@iis.se).

### Visiting address, .SE

[Ringvägen 100](#), Stockholm, Sweden

### Postal address, .SE

.SE  
Box 7399  
SE-103 91 Stockholm  
Sweden

### Phone and fax, .SE

Phone: +46-8 452 35 00  
Fax: +46-8 452 35 02

Swedish organization number: 802405-0190

VAT number: SE802405019001

E-mail address: [info@iis.se](mailto:info@iis.se)



## Resource Records (RR)

See Stevens, Vol. 1, figure 14.2, pg. 201 (augmented with additional entries)

Record type	Description
A	an IP address. Defined in RFC 1035
AAAA	an IPv6 address. Defined in RFC 1886
PTR	pointer record in the in-addr.arpa format. Defined in RFC 1035.
CNAME	canonical name $\equiv$ alias (in the format of a domain name). Defined in RFC 1035.
HINFO	Host information. Defined in RFC 1035.
MX	Mail eXchange record. Defined in RFC 1035.
NS	authoritative Name Server (gives authoritative name server for this domain). Defined in RFC 1035.
TXT	other attributes. Defined in RFC 1035.
AFSDB	AFS Data Base location. Defined in RFC 1183.
ISDN	ISDN. Defined in RFC 1183.
KEY	Public key. Defined in RFC 2065.
KX	Key Exchanger. Defined in RFC 2230.
LOC	Location. Defined in RFC 1876.
MG	mail group member. Defined in RFC 1035.
MINFO	mailbox or mail list information. Defined in RFC 1035.
MR	mail rename domain name. Defined in RFC 1035.
NULL	null RR. Defined in RFC 1035.
NS	Name Server. Defined in RFC 1035.

IK2555, SPRING 2014

SLIDE 54

P. V. Mockapetris, 'Domain names - implementation and specification', *Internet Request for Comments*, vol. RFC 1035 (INTERNET STANDARD), Nov. 1987 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1035.txt>

S. Thomson and C. Huitema, 'DNS Extensions to support IP version 6', *Internet Request for Comments*, vol. RFC 1886 (Proposed Standard), Dec. 1995 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1886.txt>

C. F. Everhart, L. A. Mamakos, R. Ullmann, and P. V. Mockapetris, 'New DNS RR Definitions', *Internet Request for Comments*, vol. RFC 1183 (Experimental), Oct. 1990 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1183.txt>

R. Allbery, 'DNS SRV Resource Records for AFS', *Internet Request for Comments*, vol. RFC 5864 (Proposed Standard), Apr. 2010 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc5864.txt>

D. Eastlake 3rd, 'Domain Name System (DNS) IANA Considerations', *Internet Request for Comments*, vol. RFC 6895 (Best Current Practice), Apr. 2013 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc6895.txt>

D. Eastlake 3rd and C. Kaufman, 'Domain Name System Security Extensions', *Internet Request for Comments*, vol. RFC 2065 (Proposed Standard), Jan. 1997 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2065.txt>

D. Eastlake 3rd, 'Domain Name System Security Extensions', *Internet Request for Comments*, vol. RFC 2535 (Proposed Standard), Mar. 1999 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2535.txt>

R. Atkinson, 'Key Exchange Delegation Record for the DNS', *Internet Request for*

*Comments*, vol. RFC 2230 (Informational), Nov. 1997 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2230.txt>

C. Davis, P. Vixie, T. Goodwin, and I. Dickinson, 'A Means for Expressing Location Information in the Domain Name System', *Internet Request for Comments*, vol. RFC 1876 (Experimental), Jan. 1996 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1876.txt>



See Stevens, Vol. 1, figure 14.2, pg . 201 (augmented with additional entries)

#### Record type Description

NSAP	Network service access point address. Defined in RFC 1348. Redefined in RFC 1637. Redefined in RFC 1706.
NXT	Next. Defined in RFC 2065.
PX	Pointer to X.400/RFC822 information. Defined in RFC 1664.
RP	Responsible Person. Defined in RFC 1183.
RT	Route Through. Defined in RFC 1183.
SIG	Cryptographic signature. Defined in RFC 2065.
SOA	Start Of Authority. Defined in RFC 1035.
SRV	Server. DNS Server resource record -- RFC 2052, for use with DDNS.
TXT	Text. Defined in RFC 1035.
WKS	Well-Known Service. Defined in RFC 1035.
X25	X25. Defined in RFC 1183.

Note that an number of the RR types above are for experimental use.

Name of an organization:

ISI.EDU. PTR 0.0.9.128.IN-ADDR.ARPA.



## Network names

Conventions:

- it.kth.se includes all the computers in the KTH/SU IT-University
- kth.se includes all the computers at KTH
- ...



As resource records (on 2014.03.26):  
 nslookup  
 > set querytype=any

```
> kth.se
;; Truncated, retrying in TCP mode.
Server:          130.237.216.10
Address:         130.237.216.10#53

kth.se          rdata_51 = # 13 0100000208B7B89C09813F6C7A
kth.se          rdata_46 = TYPE51 8 2 0 20140403180343 20140304174436 61440 kth.se.
15eoFafsKI6bKghnfiXutDOCALjQHsQajll/d51qGqcSubzqLjJ6R2yv0 Zff38+MZMDDfytwkT9/VpWJHuSx+6AhMkZyVqXrGaCrhy7x0BXGt1kkG
Q2FBJOMvI9map7pehDondsew13215v2HZ6Y5AcyJ6NoxFce8TwsYTpEo
PEppK+Mwouy4614IF+GA2ub9b70IU2VDfYQTToLEcpSsBkklg1kuniZEg
AV3pUHZCabR7YjmDuZiZAXG2pCk6rizWX9QNjauNtBpiA7OFZTNmQX62
4JLaTkMvGpQKTESkJcSGnSwZD18UY3A/TiV/36dj2M17hriLa8BjtrP qO9Row==
kth.se

      origin = a.ns.kth.se
      mail addr = hostmaster.kth.se
      serial = 2014033054
      refresh = 14400
      retry = 900
      expire = 604800
      minimum = 86400
kth.se          nameserver = a.ns.kth.se.
kth.se          nameserver = nic2.lth.se.
kth.se          nameserver = ns2.chalmers.se.
kth.se          nameserver = b.ns.kth.se.
kth.se          mail exchanger = 20 mx-alt1.kth.se.
kth.se          mail exchanger = 30 tarbaby.junkemailfilter.com.
kth.se          mail exchanger = 10 mx.kth.se.

kth.se          afsdb = 1 anden.e.kth.se.
kth.se          afsdb = 1 sonen.e.kth.se.
kth.se          afsdb = 1 fadern.e.kth.se.
Name:           kth.se
Address: 130.237.32.143
kth.se          text = "5 - Tel. +46 8 790 60 00"
kth.se          text = "3 - SE-100 44 STOCKHOLM"
kth.se          text = "1 - Royal Inst of Technology"
kth.se          text = "2 - Kungliga Tekniska HJgskolan"
kth.se          text = "4 - SWEDEN"
kth.se          rdata_48 = 257 3 8
```



kth.se (continued) – for DNSSEC

kth.se rdata\_48 = 257 3 8 AwEAAeqLuvNBVEmjzUQCkRLHcDbmOI83rOI8wmwj+wlwQ3l+Tw2MsMJ  
 9ot1J1U6p6yRjHys3V4gC5itRUKuU4waOqCgFSXMRq+EvFk80HJEN+ /PRpKgRCU537mQcDgZnBukOa4cYydXBMQ8FLMh2yfmP2Ddk1iBNTXOPS  
 lJCpIZyeJ6/hh+HURZeFed+H9CrCzovsn19JhJh+YB5UJZYOWmsnSX hgmtdm3Y YFVU5NRXD7HQadqL5D8gQPdWEXdCDB0hVvkZB7Bt1+7q  
 1pCucNUFFWnxSgmKofPg83ZELvuuGxatAH+S/Z5GJXpBvGkIBDDfLr OmaqrKdciJM=

kth.se rdata\_48 = 256 3 8 AwEAAelwmjXgoN8dlcWEy2QN40VU42hdciL Y5dMDvimmwup/9NlkYlo  
 nNwRWo8gyvr53Ql+K/TindQK0og4PQRKrguYLHv42BwMsump9kcvhzy Hy+HNSqffWYfXFIK+3GpofCIXTSTD1uMlgUx9vA4WmvejJKCd/U/R  
 D0LBT7B+Zcl8VqNhpV4VlycUokgMHSdV06IKNVIIXKUIYkDK8F8JK1 XSPv45MSQ7zchNv33DZeO0i9anVxq/rAF8fdn3yb8dfiKud18jrm1wG  
 bWvT56jsjDMFh0Skf53JXBsGCZFj/defyU0002zTjDx09Hpz/nLX3V s19lBgtf8+s=

kth.se rdata\_65534 = # 5 0856710001

kth.se rdata\_46 = DNSKEY 8 2 7200 20140414140000 20140315130000 22129 kth.se  
 Nca6AROKLx2TU+bywVVBHJVbyNvXGbaDOAXs5C+V+O/AltikDz2dRi CRgrBF/sXRzL8lgyZJ1mdOtP0sS0WbE3fNzbjEenfRHDCFBxUWa9t2  
 I83y31DB1ClkVHvhg8A3mmjITclqPbatBqEcqQwFTBSFEYfguy3+qG uRvngod05PnNqryn4O9TEETUmF2qECaXOP22xUQCZTMEOMrKv6iD7  
 XKL85ucyTN6dxxU/Nv7Q6yqWmumY0M8+uhYlcfYw/bfy74wXcmB6ff9 1q+8xNbzjldHARn4h/dE4PGfk81ciun2m3fEPps7zvK5FkJZHQsfm UMHS3g==

kth.se rdata\_46 = DNSKEY 8 2 7200 20140414140000 20140315130000 61440 kth.se UuZy1y4rwnKPCd+YecCCF7OstayL3dMj54OCGN7A/7H+JIMV8Yd7 AIOi86m6D4A4M0qPmG1R0QqRfRhb7Elib1YQmhyC3nINe8Y4  
 AulJ58F8KwEiHTCONCIUErZ2EFPZvH1TMkzssC388bub6C4V4 HJ3kmsVwvCCl9guc0e1Y7otTKa1YVqjpe1HKR9GZ/E2HwWwbsJ Vxp+g54S630nbnRUtak9mKsXEF/PhyAHHEK7G134QXasmV3UJ  
 2hmF53c5sMODD10rOyPpzn2wv9159pKCCZC0Woed1Qc9sUJ aTfC9m=

kth.se rdata\_46 = TYPE65534 8 2 0 20140414 130840 20140315130005 61440 kth.se wyOrds8thTLF0sAKoPAXqG2Nl3ZVXpmdFdyUJ0ve2R9RneV7REmth bhtESc7mthGYJrEDoXo9xqOmLIT+8z+2m3Vjmmk80Ri508xG  
 9wWYPUhJTeaU/JKucv1MCRwv0h9FyUjH2Xcc04X2Xv0849h3j OLUSk000zape0JLJTFsaWbhzqCqP/WJNTh8b7w1K4S849XW gKcXCTNlUicm7M4QTCpTnEzXK3365d4U038Kkqj0rOJ0V8  
 zzzmNKLQf8mnsYLUe4x4zVw152n0e+Kraz99Pymh+9gkuth FjgZQ2=

kth.se rdata\_46 = SOA 8 2 1800 201404260451132 201403200351132 61440 kth.se Hpt4TWJHTvA66VPVgGdJZHzrak4OGoYezak8NnGs527e4ZE cSFHOFJ0r4U+GqmJL+MQJLxMKV1xE6M0CvxE+KagbyYigP9gZw  
 Meed9GmhpRG1WKR0SVOZCDRAZ0JSMa0zUJQp9CLFah78N83 3WKGd9m=

kth.se rdata\_46 = TXT 8 2 1800 20140424045412 20140325045214 61440 kth.se CzcQahZNB20ZKtB3fCRXvAtss4AGaT0WqL1HKzabShvdU93 o4WjORPBWwUp+H3LmNyg+59cA55zvdtmALV30ovdm0X4uQRD2K96  
 ac6MKrY3CpS89f9cuVRdHCanYrAtEPF3H2CLF8+776+9pX RPyuKPGJngwtgU3K4MdbliqUJfmgqbdYf1MBJDCJRPZNKzNB NoY10F++haz2Ump2QVJkcof61f8BngzpeZfA3YnywnsHaeyf  
 k0xwTfVWUJfTf9pKedH+hd8ahKGS8MT2zucX0TfwGCMmYkMl 2v87g=

kth.se rdata\_46 = A 8 2 60 20140419103918 20140320094725 61440 kth.se 1T8d8k8mG8iV88hUCu60vM5MII03EagJF6EgDCAQR32Eew1W8x uhDvY358n1PbF5H3zZaTTeaUjVWJlRvsl42oq68MEH0U0Up  
 mChjW3EGFDF85OCb4S0b4H+K01D2R2P8P8ND248QFP9+ 0M204eyhHTm9G3o0rjns9C0WUJUCYHkca+daekUov8sU8cCKE Z9l8E3k1x2Q8HWTLDc3ms2wp98FXyYnR0v5Mh1X1WMEzstVoyzZ  
 N2g8NpY1bP9v0NVM55Umi81UAD0JkUW58e7SLxvq9ZBegy7u EAweSQ=

kth.se rdata\_46 = AFSDB 8 2 1800 20140424060154 20140325050711 61440 kth.se csOH8Hw65DCbHdMEKCRS3Cm4Q57ZBwJBLNPKW8lCHHU gCv05PKBf1ehnd4rcDCBa+3ke4nJR22T3C771Y9lIL7xBR/4  
 C18yHqQHRf9y0vAND5C8Mq8BL5388e5C6fHqRhb1Y 89u58Hbz9z4528p8eY1Yn4848B8hZc+SOVWmM1Tm815 cmRDJL0S8k8pNjuz8005YwMk8L468+KqgANRULC0C0vU  
 sTgEW1AwZ0RgmhnmM+h98wKcWxncmz7DMMWHLJUBNSCXJfP89 aT1TQ=

kth.se rdata\_46 = MX 8 2 60 20140419103918 20140320094725 61440 kth.se mN5okAwUq4MhBJyANiaco2vNv3F1+eEDvYQTHSLUHYvOB9j IdmM6OYvnlEaccSOxRfLgUxDTXJJC0005p1085K26FTJc1w  
 s0TMCW7k8eG8u5S81jjsuMwKd8qM70UJKA7psSMEAGD zOZZSD0ER0DCa6OMdg71IgzvH8275LpsGqWxNtmNofirKUy0c dH8ISeW528p8SVDfMVL81NGALvK+r1qQgYemKXkUvFsdLxaxXf  
 Wk6GSA3Pz2UjXfPmwdADrS88K8P8LzCuf8v0H8jJlU7r SgZ8e=

kth.se rdata\_46 = NS 8 2 1800 20140403201036 20140304193741 61440 kth.se S25mGKBUMAF05G+Yn5HMunHJwZV5GSL2mmPyrGRF04oJ3bmj5+0Tq kthQ7MwYpLu8XmLgU6XN8601mq+H5ZJNhbPvBUJGauQo4D  
 jYwXk3uRDTClachGdG5U9gtrJkx3R6EK9C0c3JdMh74yvcvY 8h+QhNASgzv3QUJUC78EM6C2eul5KFq35qk5RwWJ16h9s10V 4hDbKY36Tdc++B4q8btchdJ2NGO3Z+Mv96GyCyFS/VVong/QFhwEz  
 D3rm188P8KwEhFHP8m8lRlZv9z9F8kAwWz++K1743Z KJGCa=

kth.se has AAAA address 2001.680.1.1200.82ed208f

kth.se rdata\_46 = AAAA 8 2 60 20140404010055 20140326000109 61440 kth.se vas11G1asyFAS1TfVfV85h0Tckv8Qm1f8/Bc2h8A4P8d8vLoo ng288aoOa0d0vz0mTm86T86J848DDU+D8EN4eRFjwH9  
 QWU5RFpjeWwQ7CZJap8pKvYyCzETr6T0YJ4CE83u9611W52x Vem+spH2e0UYMyUAKZ2EIVVXbbipEN5Zw91TR8SPC+fyAzY dukZVm8Sv8V5MULPp1ymH8m30s1LcMQp8u86D8E0k1e6VQk8  
 17vasmkl0FNFRQ/SowP1qH8fODUJlglV+2VvcQ1LFC3L0LU 3VcgZ=



## > set querytype=MX

```
> kth.se
Server:          130.237.216.10
Address:        130.237.216.10#53
```

### Non-authoritative answer:

```
kth.se mail exchanger = 20 mx-alt1.kth.se.
kth.se mail exchanger = 30 tarbaby.junkemailfilter.com.
kth.se mail exchanger = 10 mx.kth.se.
```

weight name

### Authoritative answers can be found from:

```
kth.se nameserver = ns2.chalmers.se.
kth.se nameserver = nic2.lth.se.
kth.se nameserver = a.ns.kth.se.
kth.se nameserver = b.ns.kth.se.
mx.kth.se internet address = 130.237.48.70
mx.kth.se internet address = 130.237.32.10
mx.kth.se internet address = 130.237.48.48
mx.kth.se has AAAA address 2001:6b0:1:1300:d685:64ff:fe5f:abec
mx.kth.se has AAAA address 2001:6b0:1:1200:d685:64ff:fe5f:5afc
mx.kth.se has AAAA address 2001:6b0:1:1300:6ab5:99ff:fecc:79ec
mx-alt1.kth.se internet address = 130.237.48.70
mx-alt1.kth.se internet address = 130.237.32.10
mx-alt1.kth.se internet address = 130.237.48.48
a.ns.kth.se internet address = 130.237.72.246
b.ns.kth.se internet address = 130.237.72.250
b.ns.kth.se has AAAA address 2001:6b0:1::250
```





## Host names and info

How to give your host a name?

see RFC 1178: Choosing a Name for Your Computer

Internet Addresses: A second address for your host?

to have multiple addresses for you computer, see section on “ifconfig”

### Hostinfo (HINFO)

```
> set querytype=HINFO
```

IK2555, SPRING 2014

SLIDE 60

D. Libes, 'Choosing a name for your computer', *Internet Request for Comments*, vol. RFC 1178 (Informational), Aug. 1990 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1178.txt>



## Storing other attributes - TXT records

The general syntax is:

`<owner> <class> <ttl> TXT "<attribute name>=<attribute value>"`

Examples:

`host.widgets.com IN TXT "printer=lpr5"`

`sam.widgets.com IN TXT "favorite drink=orange juice"`

For more information see:

RFC 1464: *Using the Domain Name System To Store Arbitrary String Attributes*

IK2555, SPRING 2014

SLIDE 61

R. Rosenbaum, 'Using the Domain Name System To Store Arbitrary String Attributes', *Internet Request for Comments*, vol. RFC 1464 (Experimental), May 1993 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc1464.txt>



## Configuring DNS

Configuring the BIND resolver

`/etc/resolv.conf`

Configuring the BIND nameserver (named)

`/etc/named.boot` or `/etc/named.conf`

Configuring the nameserver database files (zone files)

<code>named.hosts</code>	the zone file that maps hostnames to IP addresses
<code>named.rev</code>	the zone file that maps IP addresses to hostnames



## Root servers <http://www.root-servers.org/>

	Operator	IP address(es)	Home ASN	Locations
A	Verisign, Inc.	198.41.0.4 2001:503:BA3E::2:30	19836	Frankfurt, DE Hong Kong, HK Los Angeles, US New York, US
B	Information Sciences Institute	192.228.79.201 2001:478:65::53		
C	Cogent Communications	192.33.4.12	2149	Bratislava, SK Chicago, US Frankfurt, DE Herndon, US Los Angeles, US Madrid, ES New York, US Paris, FR
D	University of Maryland	199.7.91.13 2001:500:2D::D	27	College Park, US Dulles, US
E	NASA Ames Research Center	192.203.230.10	297	Atlanta, US Brussels, BE Cape Town, ZA Chicago, US London, UK Los Angeles, US Mountain View, US New York, US San Paulo, BR Seattle, US Sydney, AU Tokyo, JP



## Root servers (continued)

	Operator	IP address(es)	Home ASN	Locations
F	Internet Systems Consortium, Inc.	192.5.5.241 2001:500:2f::f	3557	Amsterdam, NL Atlanta, US Auckland, NZ Barcelona, ES Beijing, CN Brisbane, AU Buenos Aires, AR Cairo, EG Caracas, VE Chennai, IN Chicago, US Dar es Salaam, TZ Dhaka, BD Dubai, AE Frankfurt, DE Hong Kong, HK Jakarta, ID Johannesburg, ZA Karachi, PK Kyiv, UA Lagos, NG Lisbon, PT London, UK Los Angeles, US Madrid, ES Monterrey, MX Moscow, RU Nairobi, KE New York, US Osaka, JP Oslo, NO Ottawa, Canada Palo Alto, US Panama, PA Paris, FR Phnom Penh, KH Podgorica, ME Port au Prince, HT Prague, CZ Quito, EC Rome, IT San Jose, US San Jose, CR Santiago de Chile, CL Sao Paulo, BR Seoul, KR Singapore, SG St. Maarten, AN Suva, Fiji Taipei, TW Tel Aviv, IL Torino, IT Toronto, CA Ulan Bator, MN Warsaw, PL

IK2555, SPRING 2014

SLIDE 64



## Root servers (continued)

	Operator	IP address(es)	Home ASN	Locations
G	U.S. DOD Network Information Center	192.112.36.4	5927	Columbus, US Fussa, JP Honolulu, US Naples, IT San Antonio, US Stuttgart-Vaihingen, DE
H	U.S. Army Research Lab	128.63.2.53 2001:500:1::803f:235	13	Aberdeen Proving Ground, US San Diego, US
I	Netnod (formerly Autonomica)	192.36.148.17 2001:7fe::53	29216	Amsterdam, NL Ankara, TR Bangkok, TH Beijing, CN Brussels, BE Bucharest, RO Chicago, US Colombo, LK Doha, QA Frankfurt, DE Geneva, CH Helsinki, FI Hong Kong, HK Jakarta, ID Johannesburg, ZA Karachi, PK Kiev, UA Kuala Lumpur, MY London, UK Lulea, SE Manama, BH Manila, PH Miami, US Milan, IT Mumbai, IN Oslo, NO Palo Alto, US Paris, FR Perth, AU Porto Alegre, BR Singapore, SG Stockholm, SE Taipei, TW Tallinn, EE Thimphu, BT Tokyo, JP Ulaanbaatar, MN Vienna, AT Washington, US Wellington, NZ Yerevan, AM

IK2555, SPRING 2014

SLIDE 65



## Root servers (continued)

	Operator	IP address(es)	Home ASN	Locations
J	Verisign, Inc.	192.58.128.30 2001:503:C27::2:30	26415	Amsterdam, NL Ashburn, US Atlanta, US Barcelona, ES Beijing, CN Brasilia, BR Brussels, BE Bucharest, RO Buenos Aires, AR Cairo, EG Cape Town, ZA Chicago, US Dakar, SN Dallas, US Denver, US Dhaka, BD Dublin, IE Dulles, US Edinburgh, UK Frankfurt, DE Fribourg, CH Guam, US Helsinki, FI Hong Kong, HK Honolulu, US Kaunas, LT Kuala Lumpur, MY Lisbon, PT Ljubljana, SI London, UK Luxembourg City, LU Lyon, FR Madrid, ES Manila, PH Miami, US Milan, IT Montreal, CA Moscow, RU Mumbai, IN Nairobi, KE New Castle, US New York, US Oslo, NO Palo Alto, US Paris, FR Perth, AU Prague, CZ Reykjavik, IS Riga, LV Rome, IT San Jose, CR San Jose, US San Juan, PR Seattle, US Seoul, KR Singapore, SG Sofia, BG Stockholm, SE Sydney, AU Taipei, TW Tallinn, EE Tokyo, JP Toronto, CA Turin, IT Vancouver, CA Warsaw, PL Wellington, NZ Willemstad, CW Zagreb, HR

IK2555, SPRING 2014

SLIDE 66



## Root servers (continued)

	Operator	IP address(es)	Home ASN	Locations
K	RIPE NCC	193.0.14.129 2001:7fd::1	25152	Abu Dhabi, AE Amsterdam, NL Athens, GR Brisbane, AU Budapest, HU Delhi, IN Doha, QA Frankfurt, DE Geneva, CH Helsinki, FI London, UK Miami, US Milan, IT Novosibirsk, RU Poznan, PL Reykjavik, IS Tokyo, JP
L	ICANN	199.7.83.42 2001:500:3::42	20144	145 sites
M	WIDE Project	202.12.27.33 2001:dc3::35	7500	Osaka, JP IPv6 enabled Global Instance (1 site) Paris, FR San Francisco, US Seoul, KR Tokyo, JP





## Load leveling [Abley 2005]

For example, f.root-servers.net has a **single** IP address (192.5.5.241), but requests sent to 192.5.5.241 are routed to *different* nameservers

- This routing can depend on:
- where the request is made from
- what the load on each of these names servers is

Note: this is transparent to the host which sent the request to F  
ISC uses Hierarchical Anycast routing to do this, with some of the servers being:

- large, redundant, ... installations serving the *global* internet
- small installations serving a *local* region
- 28 nodes as of Feb. 2005; 55 in 2014

Joe Abley, f.root-servers.net, NZNOG 2005, February 2005, Hamilton,  
NZ <http://www.isc.org/pubs/pres/NZNOG/2005/F%20Root%20Server.pdf>



## F root nameserver nodes

**IPv6 enabled Global:** Atlanta, US; Chicago, US; New York, US; Palo Alto, US; ; San Jose, US

**IPv6 enabled Local:** Amsterdam, NL; Auckland, NZ; Barcelona, ES; Beijing, CN; Brisbane, AU; Caracas, VE; Dar es Salaam, TZ; Dhaka, BD; Frankfurt, DE; Hong Kong, HK; Jakarta, ID; Johannesburg, ZA; Kyiv, UA; Lisbon, PT; London, UK; Los Angeles, US; Moscow, RU; Oslo, NO; Paris, FR; Phnom Penh, KH; Prague, CZ; Quito, EC; Santiago de Chile, CL; Sao Paulo, BR; Seoul, KR; Singapore, SG; St. Maarten, AN; Toronto, CA

**IPv4 enable Local:** Buenos Aires, AR; Cairo, EG Chennai, IN; Dubai, AE Karachi, PK; Lagos, NG; Madrid, ES; Monterrey, MX; Nairobi, KE; Osaka, JP; Ottawa, Canada; Panama, PA; Podgorica, ME; Port au Prince, HT; Rome, IT; San Jose, CR; Suva, Fiji; Taipei, TW; Tel Aviv, IL; Torino, IT; Ulan Bator, MN; Warsaw, PL



## Where is f.root-servers.net ?

traceroute to f.root-servers.net (192.5.5.241), 30 hops max, 40 byte  
(times in ms)

1	net212-b.ict.kth.se (130.237.212.129)	1.840	1.876	1.892
2	cn6-vss1-p2p.gw.kth.se (130.237.211.118)	1.146	1.231	1.307
3				
4	kth-br2.sunet.se (193.11.0.197)	0.444	0.455	0.446
5	m1tug-ae4.sunet.se (130.242.85.209)	0.613	22.889	0.643
6	t1tug-ae1-v1.sunet.se (130.242.83.41)	0.681	0.720	0.685
7	se-tug.nordu.net (109.105.102.17)	0.574	0.621	0.602
8	se-fre.nordu.net (109.105.97.1)	0.658	0.726	0.692
9	10gigabitethernet1-3.core1.sto1.he.net (194.68.128.187)	6.495	6.964	7.203



## Where is f.root-servers.net ? (continued)

traceroute to f.root-servers.net (192.5.5.241), 30 hops max, 40 byte  
(times in ms)

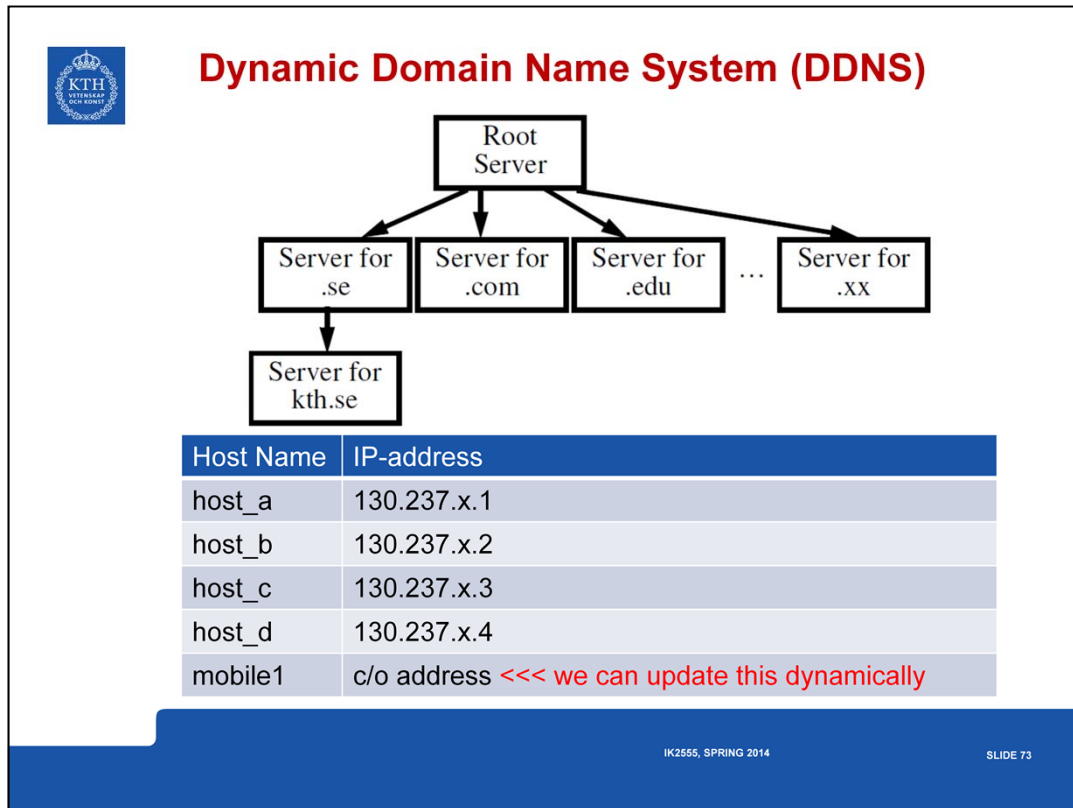
10	10ge3-1.core1.cph1.he.net (184.105.223.206)	13.062	12.985	13.034
11	10ge16-1.core1.fra1.he.net (184.105.223.201)	32.656	31.347	27.261
12	100ge5-2.core1.par2.he.net (72.52.92.13)	36.154	36.163	36.165
13	10ge15-1.core1.ash1.he.net (184.105.213.93)	109.742	107.485	113.367
14	10ge1-2.core1.atl1.he.net (184.105.213.110)	138.077	145.210	145.277
15	isc.gige-g2-1.core1.atl1.he.net (216.66.0.50)	182.818	182.580	182.714
16	f.root-servers.net (192.5.5.241)	176.932	176.897	176.831



## Where is i.root-servers.net ?

traceroute to i.root-servers.net (192.36.148.17), 30 hops max, 40 byte packets

		(times in ms)		
1	net212-b.ict.kth.se (130.237.212.129)	26.460	26.455	26.469
2	cn6-vss1-p2p.gw.kth.se (130.237.211.118)	0.495	0.566	0.641
3	br1g-cn6-p2p.gw.kth.se (130.237.0.1)	0.412	0.428	0.419
4	kth-br2.sunet.se (193.11.0.197)	0.449	0.460	0.452
5	m1tug-ae4.sunet.se (130.242.85.209)	0.624	0.643	0.636
6	t1tug-ae1-v1.sunet.se (130.242.83.41)	0.698	3.712	3.705
7	10ge-2-3-1500.outer-b-gw.sth.netnod.se (194.68.128.172)	1.064	1.191	1.265
8	peering.r1.sth.dnsnode.net (194.146.105.187)	0.824	0.829	0.819
9	i.root-servers.net (192.36.148.17)	0.876	0.881	0.877





## DDNS

RFC 2136: Dynamic Updates in the Domain Name System (DNS UPDATE)

- add or delete resource records

RFC 2052: A DNS RR for specifying the location of services (DNS SRV)

- When a SRV-cognizant web-browser wants to retrieve

`http://www.asdf.com/`

it does a lookup of

`http.tcp.www.asdf.com`

RFC 2535: Domain Name System Security Extensions (DNSSec)

IK2555, SPRING 2014

SLIDE 74

P. Vixie, S. Thomson, Y. Rekhter, and J. Bound, 'Dynamic Updates in the Domain Name System (DNS UPDATE)', *Internet Request for Comments*, vol. RFC 2136 (Proposed Standard), Apr. 1997 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2136.txt>

A. Gulbrandsen and P. Vixie, 'A DNS RR for specifying the location of services (DNS SRV)', *Internet Request for Comments*, vol. RFC 2052 (Experimental), Oct. 1996 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2052.txt>

D. Eastlake 3rd, 'Domain Name System Security Extensions', *Internet Request for Comments*, vol. RFC 2535 (Proposed Standard), Mar. 1999 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc2535.txt>



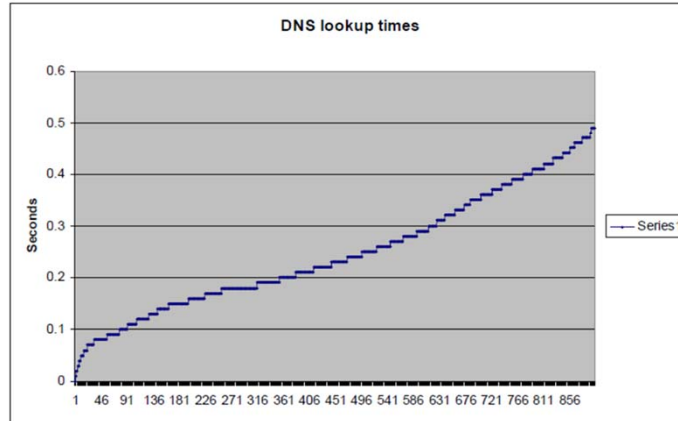
- Denial of service (DoS) and Distributed DoS (DDoS) attacks against a DNS server  
See for example:  
Root server attack on 6 February 2007, Factsheet, ICANN, 1 March 2007  
[http://www.icann.org/announcements/factsheet-dns-attack-08mar07\\_v1.1.pdf](http://www.icann.org/announcements/factsheet-dns-attack-08mar07_v1.1.pdf)
- DNS cache poisoning
- DNS spoofing
- ...





## DNS performance

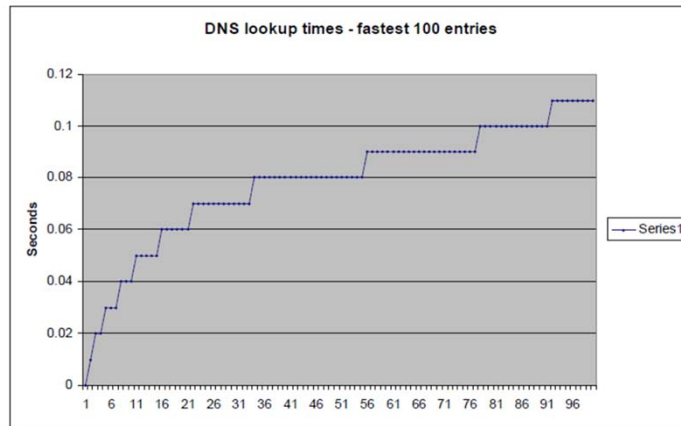
From [www.dnsserverlist.org](http://www.dnsserverlist.org) - 900 servers query times (2010.03.21)





## DNS performance - top 100

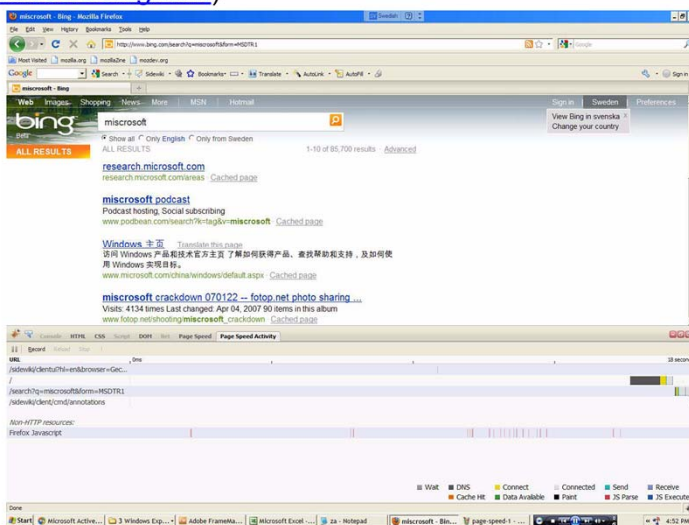
From [www.dnsserverlist.org](http://www.dnsserverlist.org) - top 100 servers query times (2010.03.21)





## Web performance

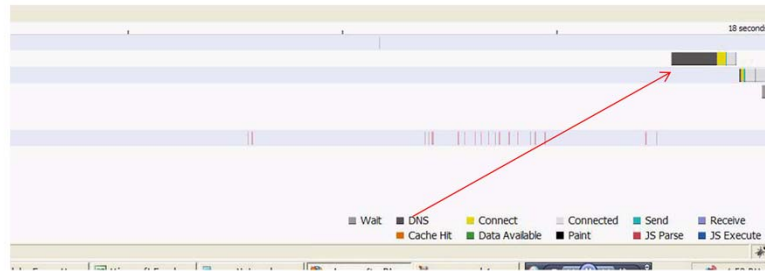
Using Page Speed (<http://code.google.com/speed/page-speed/docs/using.htm>)





## Web performance - continued

Using Page Speed (<http://code.google.com/speed/page-speed/docs/using.htm>)





## Public/Commercial DNS services

A number of companies are providing DNS servers to speed up DNS lookups:

- Google Public DNS  
<http://code.google.com/speed/public-dns/>
- OpenDNS  
<http://www.opendns.com/>
- Scrubit - offering "clean, reliable, and speedy DNS service"  
<http://www.scrubit.com/>
- dnsadvantage  
<http://www.dnsadvantage.com/>
- ...



## Summary

This module we have discussed:

- UDP
- BOOTP
- DHCP
- DNS, DDNS



# ¿Questions?

IK1550, SPRING 2014

SLIDE 82