

Slide 1

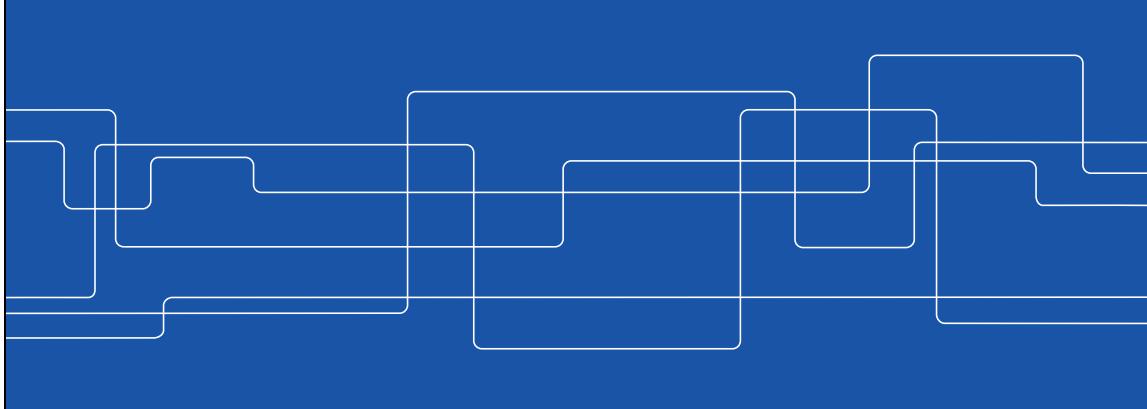


# IK1552

## Internetworking/Internetes teknik

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

School of Electrical Engineering and Computer Science (EECS), KTH Royal Institute of Technology  
IK1552 Spring 2019, Period 4 2019.03.13 © 2019 G. Q. Maguire Jr. All rights reserved.



Slide 2



## Module 2: IP Basics: Routing, ARP, and RARP

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*.

IK1552

SPRING 2019

SLIDE 2

Slide 3



## IP Basics Outline

- IP Routing: Delivery and Routing of IP packets
- Address Resolution: ARP and RARP

IK1552

SPRING 2019

SLIDE 3

Slide 4



## Connection-oriented vs. Connectionless

### Connection-Oriented Services

- Network layer first establishes a connection between a source and a destination
- Packets are sent along this connection
- Route is decided **once** at the time the connection is established
- Routers/switches in connection-oriented networks are **stateful**

### Connectionless Services

- Network layer can process each packet **independently**
- A route lookup is performed for **each** packet
- IP is connectionless
- IP routers are **stateless**

Of course reality is (much) more complex. To gain performance IP routers dynamically create state (in caches) as there is frequently **correlation** between packets (i.e., if you just did a route lookup for destination B, there is a non-zero probability that another packet which will arrive shortly might also be headed to destination B).

Slide 5



## Routing

The internet protocols are based on moving packets from a source to a destination with each hop making a routing decision.

Two components to routing:

- **packet forwarding** - Routing Mechanism: search the routing table and decide which interface to send a packet out.
  - A matching host address? If no,
  - A matching network address? (using longest match) If no,
  - Default entry.
- **computing routes** - Routing Policy: rules that decide which routes should be added into the routing table.

Traditionally most of the complexity was in the later (i.e., computing routes) while packet forwarding was very straight forward -- this is no longer true due to Quality of Service (QoS).

Routers vs. hosts -- a node can be both

- Routers forward IP packets
- Hosts generate or sink IP packets

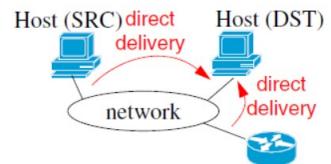
Slide 6



## Direct vs. indirect Delivery

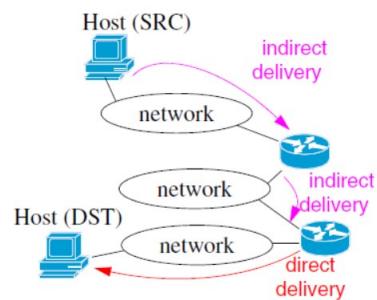
### Direct delivery

- The final destination is (directly) connected to the same physical network as the sender
- IP destination address and local interface have the same netmask
- Map destination IP address to destination physical address via **ARP**



### Indirect delivery

- From router to router (note: the last delivery is always direct!)
- Destination address is used for a **routing lookup** in a **routing table**: Routing



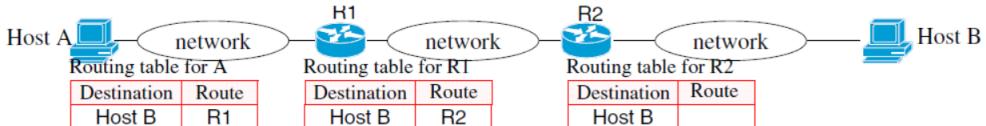
SRC = Source, DST = Destination

## Slide 7

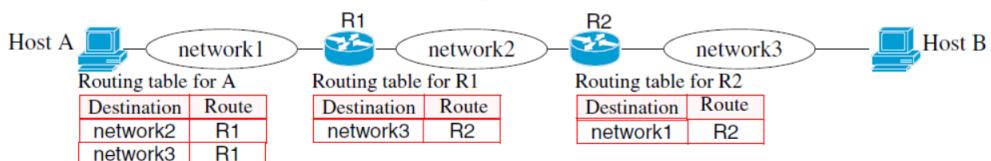


## Forwarding

- Next-Hop method - routing table holds only the address of the next hop

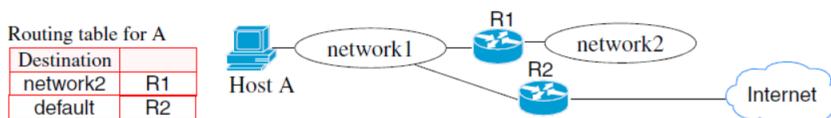


- Network-specific method - routing table entries are for networks

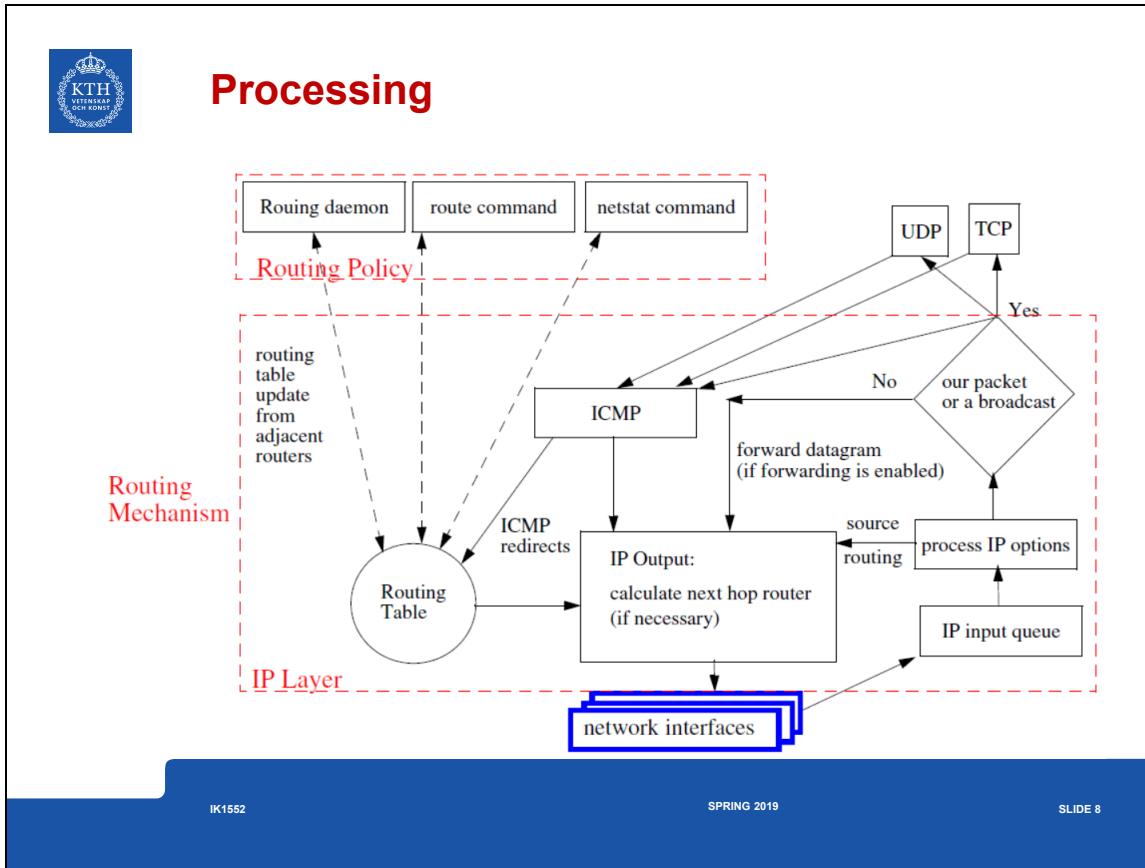


- Host-specific method - per host routes

- Default method - specifies a default route (normally network address 0.0.0.0)



## Slide 8

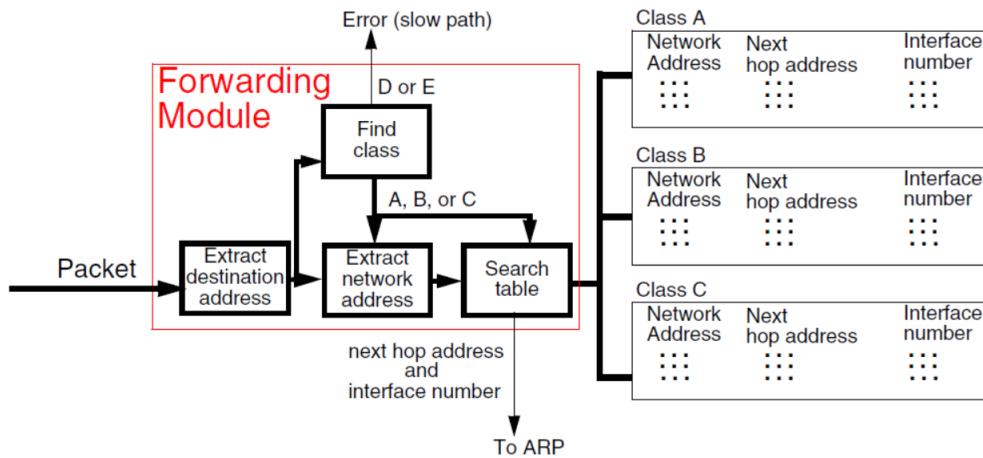


Slide 9



## Forwarding module

A simplified view of forwarding using classful address *without* subnetting:



The bulk of the forwarding effort is **searching** the tables (as most of the rest of the processing is simple logical bit operations).

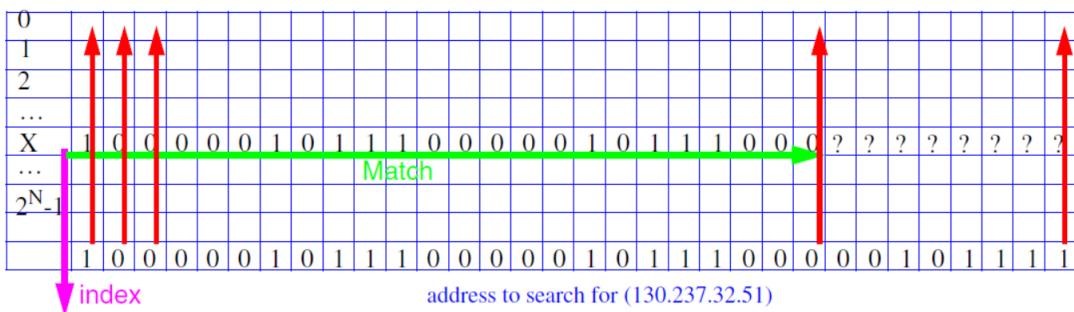
Slide 10



## Routing Table Search - Classless

Match destination with longest prefixes first

- Software algorithms: tree, binary trees, tries (different data structures) [Ravikumar 2002]
- Hardware support: Content Addressable Memory (CAM)
  - Ternary CAM (TCAM) - [Renesas],[Yu 2004] - for searching and pattern matching
  - Ternary because each cell has three states: 0, 1, or ? (aka "don't care")
  - Performance an associate search (i.e., in **parallel**), returns the index (X) of the first match
  - 4-8 ns/search (i.e., 125M - 250M lookups per second) [2019: clock speeds are >2GHz]



Note that the entries in the table have to be sorted - to achieve the longest match being at the smallest index.

[Ravikumar 2002] V. C. Ravikumar, R. Mahapatra, and J. C. Liu, 'Modified LC-trie based efficient routing lookup', presented at the 10th IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, 2002. MASCOTS 2002, 2002, pp. 177–182, DOI:10.1109/MASCOT.2002.1167075  
<http://faculty.cs.tamu.edu/rabi/Publications/Mascot-final-proceeding.pdf>

[Renesas] Renesas Technology Corp. TCAM description. Formerly available from  
[http://www.renesas.com/fmwk.jsp?cnt=tcam\\_series\\_landing.jsp&fp=/applications/network/memory/tcam/](http://www.renesas.com/fmwk.jsp?cnt=tcam_series_landing.jsp&fp=/applications/network/memory/tcam/)

[Yu 2004] Fany Yu, Randy H. Katz, and T. V. Lakshman, "Gigabit Rate Multiple-Pattern Matching with TCAM", University of California at Berkeley, Computer Science department, January 2004, [http://sahara.cs.berkeley.edu/jan2004-retreat/slides/Fang\\_retreat.ppt](http://sahara.cs.berkeley.edu/jan2004-retreat/slides/Fang_retreat.ppt)

Slide 11



## Fast forwarding

Mikael Degermark, Andrej Brodnik, Svante Carlsson, and Stephen Pink, "Small Forwarding Tables for Fast Routing Lookups", in Proceedings of the ACM SIGCOMM'97. {basis for Effnet AB}

IP routing lookups must find routing entry with longest matching prefix.

Networking community *assumed* it was impossible to do IP routing lookups in software fast enough to support gigabit speeds - but they were wrong!

Paper presents a forwarding table data structure designed for quick routing lookups.

- Such forwarding tables are small enough to fit in the cache of a conventional general purpose processor.
- The forwarding tables are very small, a large routing table with 40,000 routing entries can be compacted to a forwarding table of 150-160 Kbytes.
- With the table in cache, a 200 MHz Pentium Pro or 333 MHz Alpha 21164 can perform >2 million lookups per second.
- A lookup typically requires less than 100 instructions on an Alpha, using eight memory references accessing a total of 14 bytes.

∴ Full routing lookup of each IP packet at gigabit speeds without special hardware

IK1552

SPRING 2019

SLIDE 11

M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, 'Small Forwarding Tables for Fast Routing Lookups', in *Proceedings of the ACM SIGCOMM '97 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, New York, NY, USA, 1997, pp. 3–14 [Online]. Available: <http://doi.acm.org/10.1145/263105.263133>

M. Degermark, A. Brodnik, S. Carlsson, and S. Pink, 'Small Forwarding Tables for Fast Routing Lookups', *SIGCOMM Comput. Commun. Rev.*, vol. 27, no. 4, pp. 3–14, Oct. 1997. DOI:10.1145/263109.263133

<ftp://cdt.luth.se/micke/sigcomm97-lookup.ps.Z>

Slide 12



## Routing Tables

Aggregate IP addresses (i.e., exploit CIDR)

- more specific networks (with longer prefixes)
  - less specific networks (with shorter prefixes)
- ⇒ smaller routing tables

If each routing domain exports (i.e., tells others) only a small set of prefixes, this makes it easier for other routers to send traffic to it

✗ Unfortunately this requires clever address assignments

Some mechanisms lead to increased fragmentation

- Due to limited availability of addresses long prefixes (particularly /24) are scattered geographically
- Increasingly sites are connected to multiple ISPs (for redundancy) i.e., Multihoming, thus they have addresses from several different subnetworks

In 2005, routing tables had ~157,975 entries [APNIC] (of which a large fraction are /24 prefixes) with a growth rate of “18,000 entries per year”[Huston 2005]. As of 3 March 2016: 595,692 prefixes, 334,379 CIDR aggregated.

There are a limited number of prefixes for Class A + B + C networks (2,113,664). If the longest prefixes which a backbone router had to deal with were /24, then a table with 16,777,216 entries would be sufficient (even without aggregation) - each entry only needs to store the outgoing port number! This would allow a direct lookup in a memory of ~26 Mbytes - with up to 256 outgoing ports.

IK1552

SPRING 2019

SLIDE 12

APNIC, Routing Table Report 04:00 +10GMT Sat 19 Mar, 2005, North American Network Operators Group, Weekly Routing Table Report, From: Routing Table Analysis, Mar 18 13:10:37 2005, "This is an automated weekly mailing describing the state of the Internet Routing Table as seen from APNIC's router in Japan. Daily listings are sent to [bgp-stats@lists.apnic.net](mailto:bgp-stats@lists.apnic.net)" <http://www.merit.edu/mail.archives/nanog/2005-03/msg00401.html>

Geoff Huston, Routing Table Status Report, Policy SIG, APNIC19, Kyoto, Japan, Feb 24 2005

CIDR REPORT for 3 Mar 16,  
 This report was generated at Thu Mar 3 09:15:14 2016 AEST.  
<http://www.cidr-report.org/as2.0/>

Slide 13



## Routing table

Flags	Destination IP address	Next-hop Router IP address	point to local interface to use	Refcnt	Use	PMTU...
UGH	140.252.13.65	140.252.13.35	emd0	ddd	ddd	ddd
U	140.252.13.32	140.252.13.34	emd0	ddd	ddd	ddd
UG	default	140.252.13.33	emd0			
UH	127.0.0.1	127.0.0.1	lo0			

where *ddd* is some numeric value.

display the routing table with "netstat -rn"  
 "r" is for routing table  
 "n" asks for numeric IP addresses rather than name

Flags:

- U route is **Up**
- G route is to a **Gateway**
- H route is to a **Host**
- D route was **Discovered** by a redirect
- M route was **Modified** by a redirect

Slide 14



## Host vs. router - two behaviors

- Hosts generate or sink IP packets
- Routers forward IP packets

Thus it is possible for a device to be both a host **and** a router.

**Unless** a host is **explicitly** configured as a router, it is **not** supposed to forward IP datagrams. The default behavior must be **never forward**.

In linux the variable which controls this is:

/proc/sys/net/ipv4/ip\_forward

- If this variable is set to **1**, then the node **will** perform IP forwarding.
- If this variable is set to **0**, then the node **will not** perform IP forwarding.

Slide 15



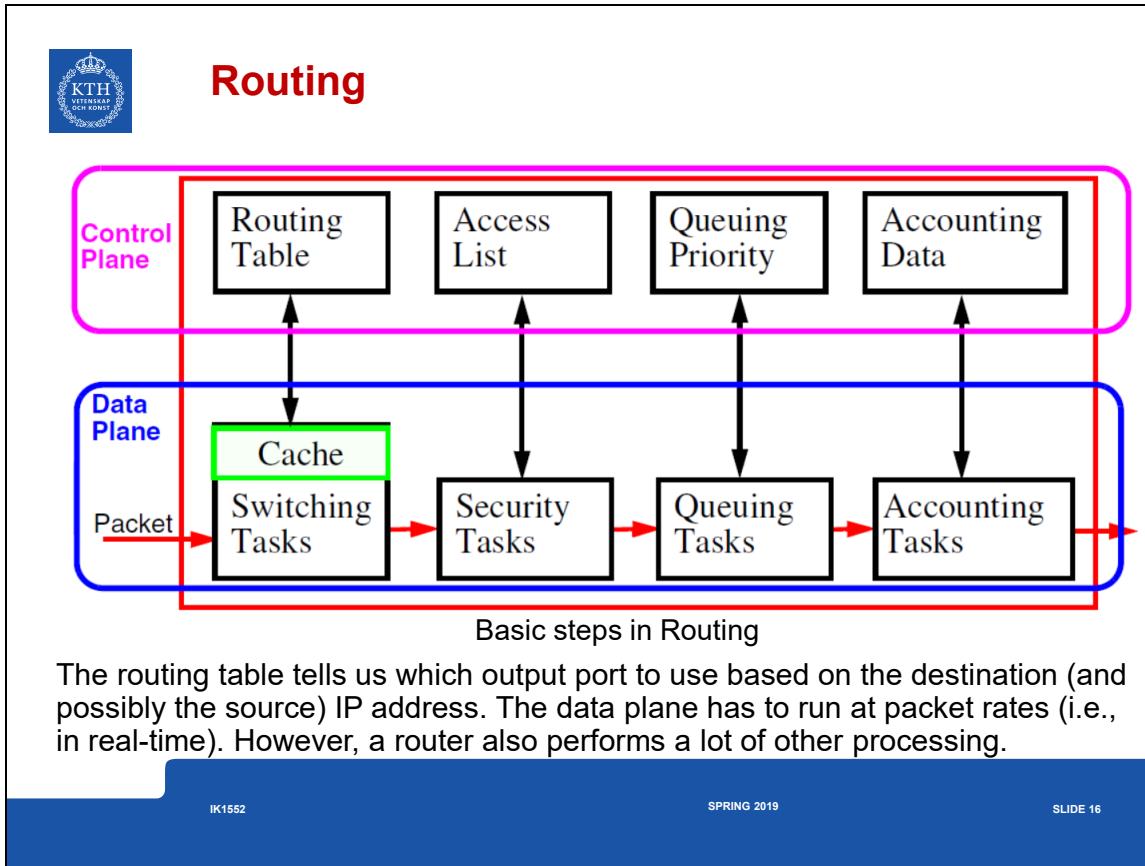
## Host routing

A host either:

- knows a route - manually **configured** [i.e., "Static routes"]
  - from the interface (for directly connected networks) or manually via the "route" command
- or **learns of a route** [i.e., "Dynamic routes"]
  - Simplest method of learning a route:  
The host sends a packet via the default route and is told via an ICMP Redirect of a better route
  - or the host hears an ICMP router advertisement (perhaps in response to its ICMP router solicitation message)
    - routers (**almost**) periodically broadcast or multicast advertisements of their existence and desire to provide routing service
    - format of ICMP router advertisement packet shown in Forouzan figure 9.18 on page 226
    - advertisements typically every 450..600 seconds
    - advertisements have a stated lifetime (typically 30 minutes)
  - or the host learns via a dynamic routing protocol.
  - or uses a **default** route.

On booting hosts send ~3 ICMP router solicitation messages (~3 seconds apart) to find a default router. This allows for dynamic discovery of the default router.

Slide 16



Slide 17



## Combining layers

Many devices now combine processing of several layers:

- Switch/Routers: combine layers 2+3
- Devices combining layers 3+4 are appearing - which extract “flows” based on looking at transport layer port numbers in addition to network addresses.

Slide 18



## ARP and RARP

Address resolution (logical  $\Leftrightarrow$  physical addresses):

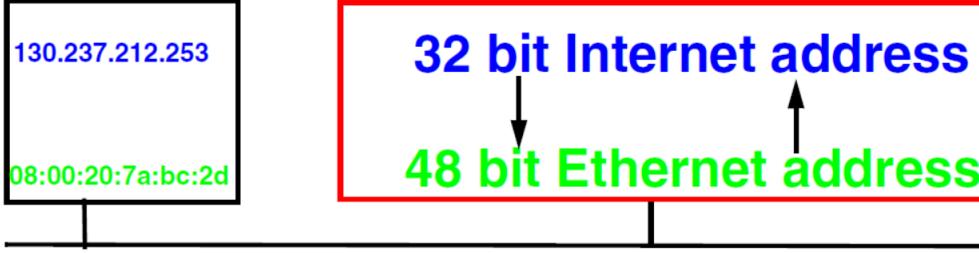
- Mapping IP addresses  $\Rightarrow$  link layer (MAC) addresses via **Address Resolution Protocol (ARP)**
- Mapping link layer (MAC) addresses  $\Rightarrow$  IP addresses via **Reverse Address Resolution Protocol (RARP)**

Slide 19

 **What to do with a new computer?**

We will assume that the computer has an Ethernet interface:

oscar.it.kth.se      ?hostname?      A new computer

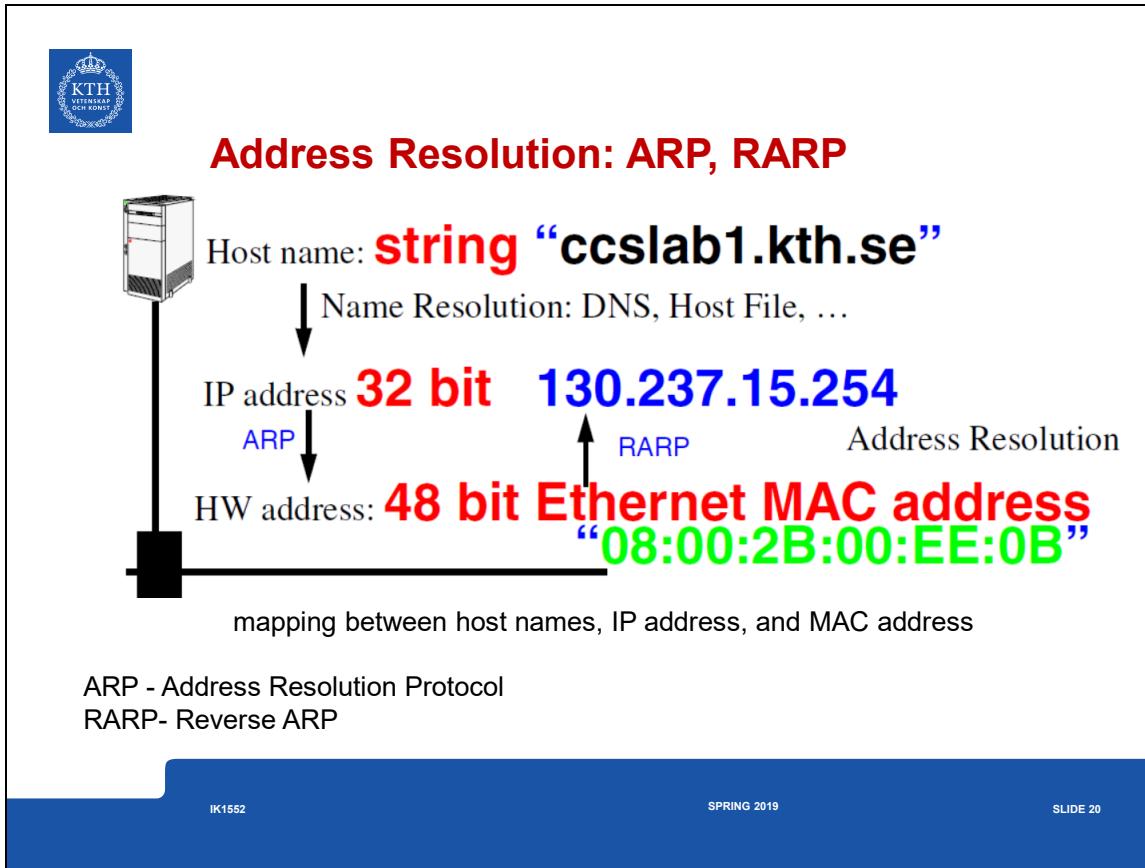


The diagram illustrates the mapping between a host name, IP address, and MAC address. It shows a host name "oscar.it.kth.se" and a question mark "?hostname?" above a red-bordered box containing "32 bit Internet address". Below this box is another red-bordered box containing "48 bit Ethernet address". To the left of the boxes is an IP address "130.237.212.253" and a MAC address "08:00:20:7a:bc:2d". Arrows point from the host name/question mark to the top box, and from the IP/MAC addresses to the bottom box. A horizontal line at the bottom is labeled "Name and IP Address needed!".

**Direct mapping** - requires no I/O, just a computation; hard to maintain; and requires stable storage (since you have to store the mappings somewhere) or  
**Dynamic Binding** - easier to maintain; but has a delay while messages are exchanged

IK1552      SPRING 2019      SLIDE 19

Slide 20



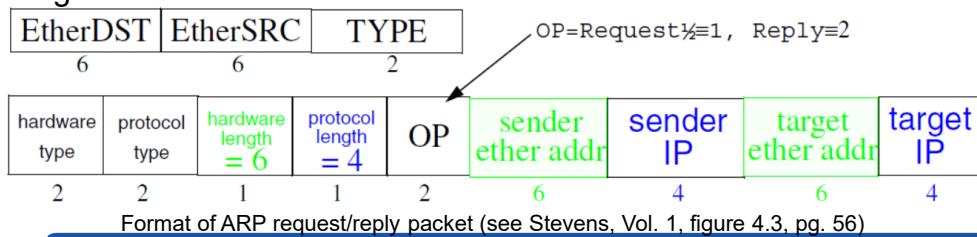
Slide 21



## ARP ≡ Address Resolution Protocol (RFC826)

Address Resolution Protocol (ARP) - allows a host to find the physical address of a target host **on the same network**, given only target's IP address.

- Sending host (source) wants to send an IP datagram, but does not know the corresponding ethernet address
- ARP request - broadcast to every host on the network (i.e., EtherDST=0xFFFFFFFFFFFF), TYPE=0x0806
- Destination host: "It is my address!" and sends an ARP reply
- Source host - receives the unicast ARP reply, and now uses it to send the IP datagram



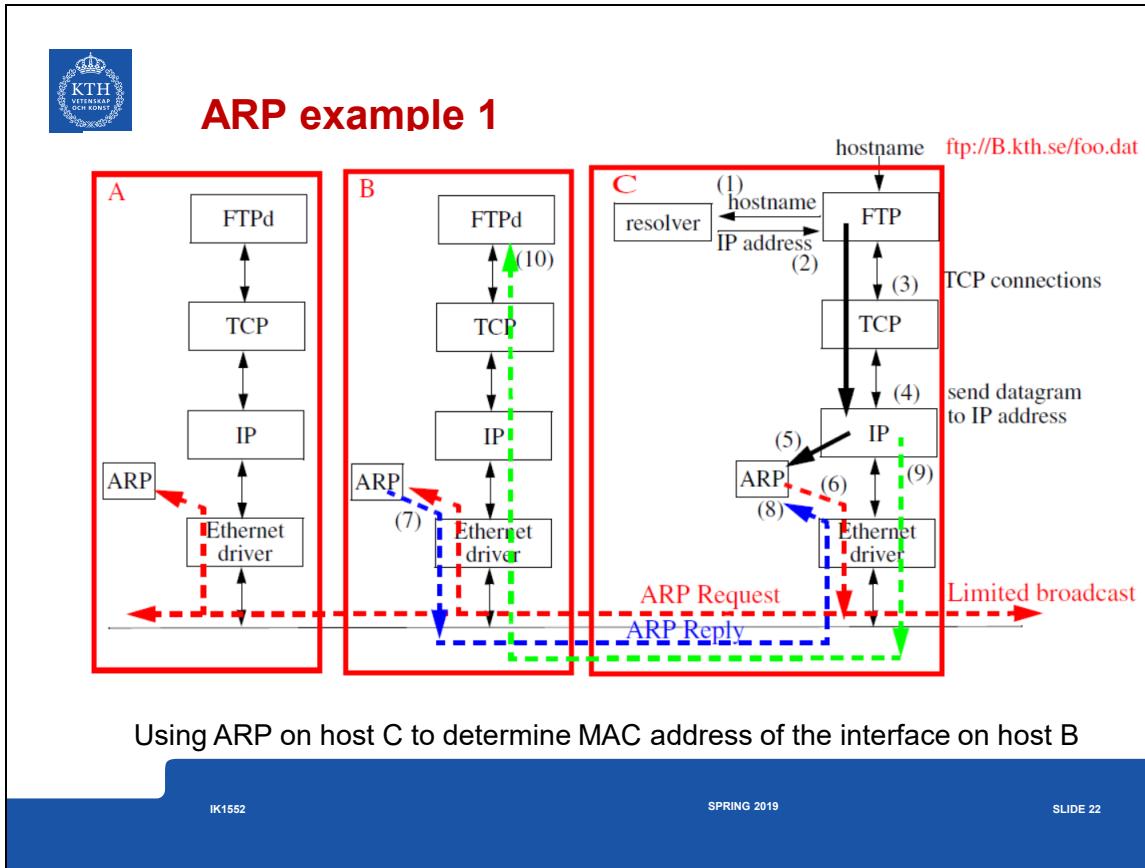
IK1552

SPRING 2019

SLIDE 21

D. Plummer, 'Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware', *Internet Request for Comments*, vol. RFC 826 (INTERNET STANDARD), Nov. 1982 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc826.txt>

Slide 22



Slide 23



## Address Resolution Cache

Since you have just looked up the address, save (cache) it for reuse:

- to limit ARP traffic
- works because of correlations in use of addresses

You can examine the arp cache:

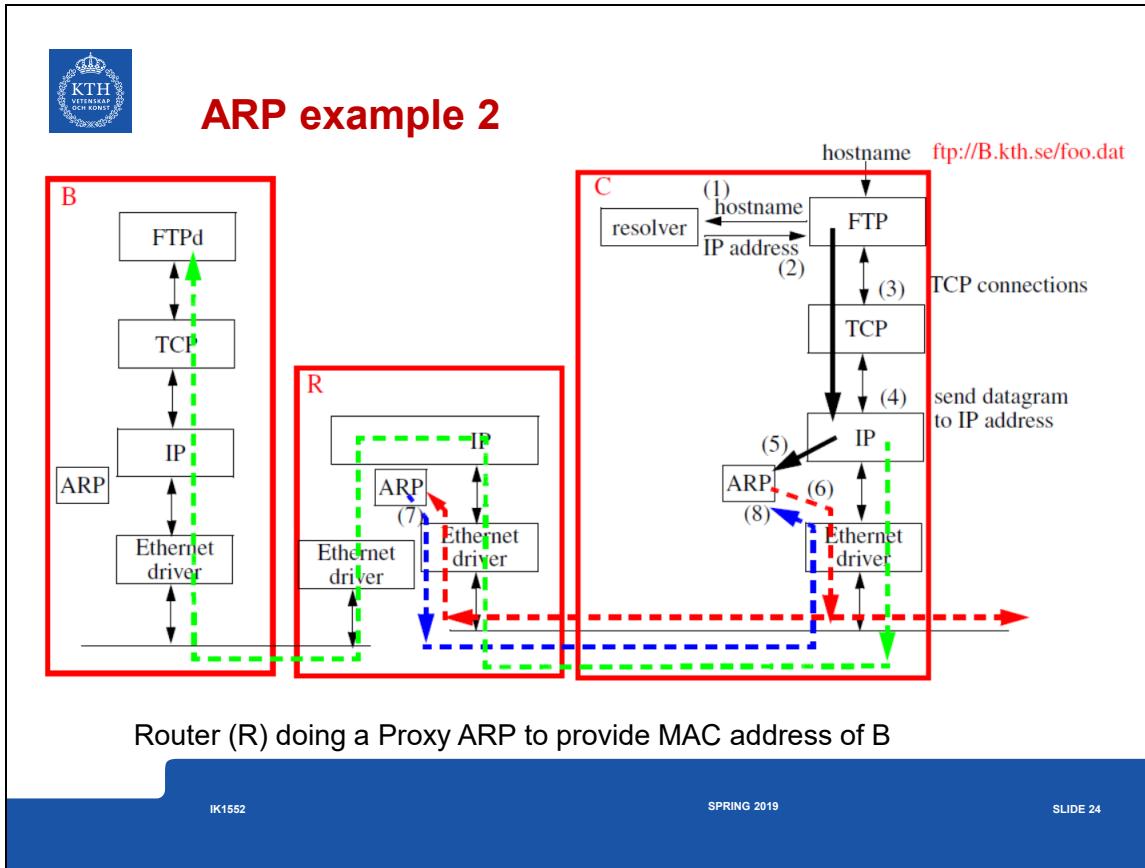
```
arp -a  
machine-name (x.x.x.x) at xx:xx:xx:xx:xx:xx  
...  
arp -an  
(x.x.x.x) at xx:xx:xx:xx:xx:xx  
...
```

Note that the later form (with the “n” option) does *not* lookup the hostname, this is *very useful* when you don’t yet have a name resolution service working!

### ARP Refinements

Since the sender’s Internet-to-Physical address binding is in every ARP broadcast; (all) receivers update their caches before processing an ARP packet

Slide 24



Slide 25



## Proxy ARP (RFC 826)

Lets a **router** on the network answer for a host which is NOT necessarily on the local network segment.

But how does this router know?

- It can make an ARP request itself or
- Perhaps it already knows - because it has an entry in its ARP cache

D. Plummer, 'Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware', *Internet Request for Comments*, vol. RFC 826 (INTERNET STANDARD), Nov. 1982 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc826.txt>

Slide 26



## Gratuitous ARP

Host sends a request for its own address

- generally done at boot time to inform other machines of its address (possibly a new address) - gives these other hosts a chance to update their cache entries immediately
- lets hosts check to see if there is another machine claiming the same address ⇒ “**duplicate IP address sent from Ethernet address a:b:c:d:e:f**”

As noted before, hosts have paid the price by servicing the broadcast, so they can cache this information - this is one of the ways the proxy ARP server could know the mapping.

Note that **faking** that you are another machine can be used to provide **failover** for servers (see for example heartbeat, fake, etc. at <http://linux-ha.org/wiki/Download> for a `send_arp` program). [It can also be used very various **attacks!**]

Slide 27



## Additional ARP commands

**Publish entries** (i.e., mechanically make an entry and answer replies)

Publishing entries is one way that (embedded) devices can learn their IP address.

```
# arp -s birkexample 08:00:2B:00:EE:0B pub  
# arp -an  
(192.168.1.1) at 0:4:5a:de:e8:f9 ether  
...  
(172.16.32.20) at 8:0:2b:0:ee:b ether permanent published
```

where birkexample has the IP address: 172.16.32.20

**Explicitly delete entries**

```
# arp -d birkexample  
birkexample (172.16.32.20) deleted  
# arp -an  
(192.168.1.1) at 0:4:5a:de:e8:f9 ether
```

IK1552

SPRING 2019

SLIDE 27

## Slide 28



## ARP - as seen with ethereal

Time Source Destination Protocol Info  
1.995245 172.16.33.3 Broadcast ARP Who has 172.16.33.2? Tell 172.16.33.3  
Frame 2 (60 bytes on wire, 60 bytes captured) Arrival Time: Mar 23, 2005 11:32:45.184792000 Time delta from previous packet:  
1.995245000 seconds Time since reference or first frame: 1.995245000 seconds

IEEE 802.3

Ethernet Destination: ff:ff:ff:ff:ff:ff (Broadcast)

Source: 00:40:8c:30:d4:32 (172.16.33.3)

Length: 36

Trailer: 00000000000000000000000000000000

Type: ARP (0x0806)

Address Resolution Protocol (request)

Hardware type: IEEE 802 (0x0006)

Protocol type: IP (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: request (0x0001)

Sender MAC address: 00:40:8c:30:d4:32 (172.16.33.3)

Sender IP address: 172.16.33.3 (172.16.33.3)

Target MAC address: ff:ff:ff:ff:ff:ff (Broadcast)

Target IP address: 172.16.33.2 (172.16.33.2)

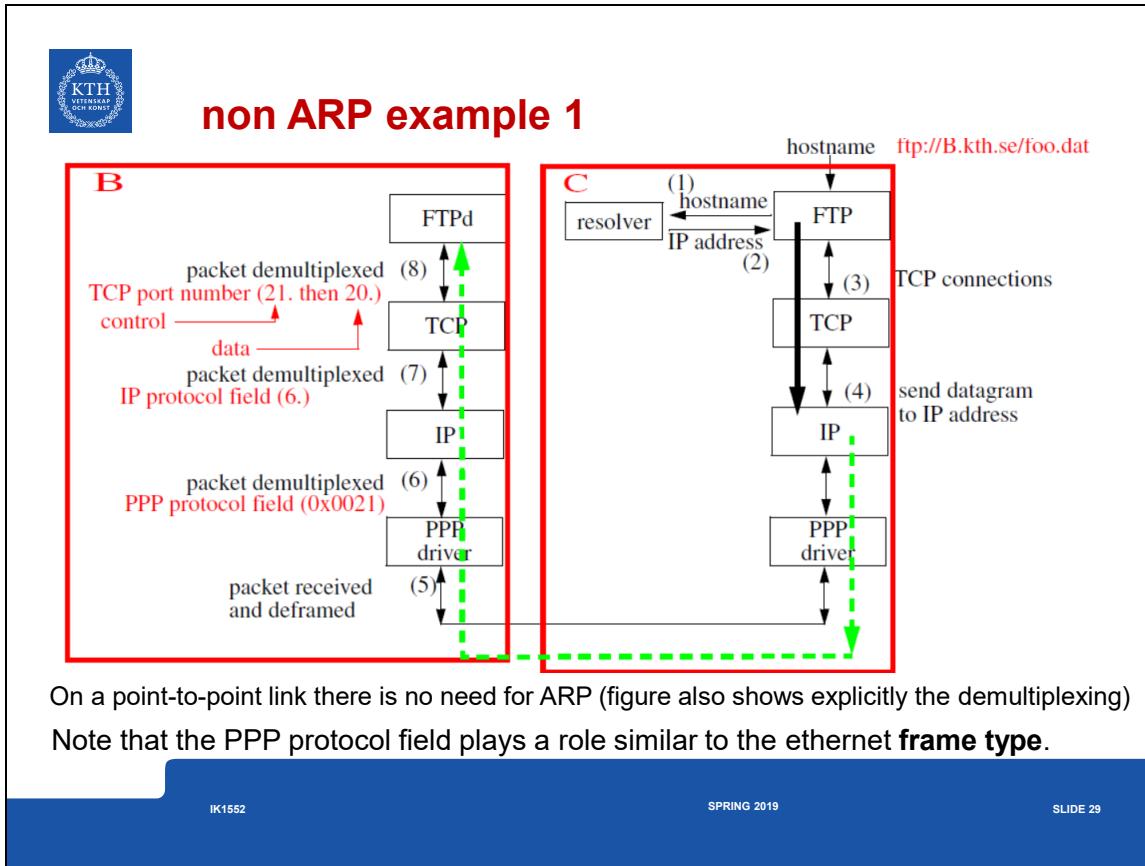
0000 ff ff ff ff ff 00 40 8c 30 d4 32 00 24 aa aa .....@.0.2.\$..

0010 03 00 00 00 08 06 00 06 08 00 06 04 00 01 00 40 .....@

0020 8c 30 d4 32 ac 10 21 03 ff ff ff ff ff ac 10 .0.2.!..... << unlike what page 163 says it is **not** all zeros!

0030 21 02 00 00 00 00 00 00 00 00 00 00 !.....

Slide 29



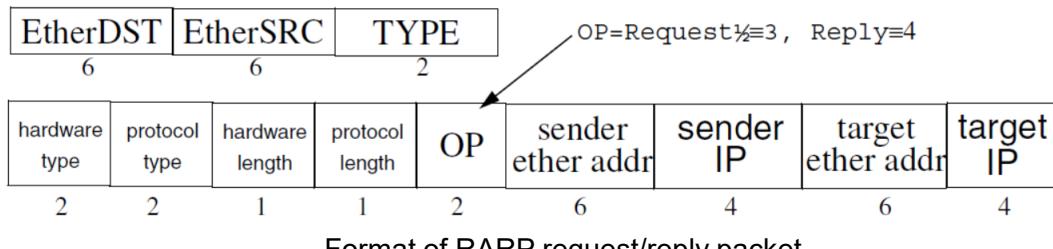
Slide 30



## RARP: Reverse Address Resolution Protocol (RFC 903)

How do you get your own IP address, when all you know is your link address?

- Necessary if you don't have a disk or other stable store
- RARP request - broadcast to every host on the network (i.e., EtherDST=0xFFFFFFFFFFFF), TYPE=0x8035
- RARP server: "I know that address!" and sends an RARP reply
- Source host - receives the RARP reply, and now knows its own IP addr



Note: You can now see what the "publish" aspect of the arp command is for.

[RFC 903] R. Finlayson, T. Mann, J. C. Mogul, and M. Theimer, 'A Reverse Address Resolution Protocol', *Internet Request for Comments*, vol. RFC 903 (INTERNET STANDARD), Jun. 1984 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc903.txt>

Slide 31



## RARP - as seen with ethereal

Source Destination Protocol Info 0.000000 172.16.33.3 Broadcast RARP Who is 00:40:8c:30:d4:32? Tell 00:40:8c:30:d4:32

Frame 1 (60 bytes on wire, 60 bytes captured)

Arrival Time: Mar 23, 2005 11:32:43.189547000

Time delta from previous packet: 0.000000000 seconds

Time since reference or first frame: 0.000000000 seconds

Ethernet II, Src: 00:40:8c:30:d4:32, Dst: ff:ff:ff:ff:ff:ff

Destination: ff:ff:ff:ff:ff:ff (Broadcast)

Source: 00:40:8c:30:d4:32 (172.16.33.3) Type: RARP (0x8035)

Trailer: 00000000000000000000000000000000...

Address Resolution Protocol (reverse request)

Hardware type: Ethernet (0x0001)

Protocol type: IP (0x0800)

Hardware size: 6

Protocol size: 4

Opcode: reverse request (0x0003)

Sender MAC address: 00:40:8c:30:d4:32 (172.16.33.3)

Sender IP address: 0.0.0.0 (0.0.0.0)

Target MAC address: 00:40:8c:30:d4:32 (172.16.33.3)

Target IP address: 0.0.0.0 (0.0.0.0)

0000 ff ff ff ff ff 00 40 8c 30 d4 32 80 35 00 01 .....@.0.2.5..

0010 08 00 06 04 00 03 00 40 8c 30 d4 32 00 00 00 00 .....@.0.2.... <<< as the source does not know its own IP address

0020 00 40 8c 30 d4 32 00 00 00 00 00 00 00 00 00 00 ..@.0.2..... <<< as the source does not know the target's IP address

0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

IK1552

SPRING 2019

SLIDE 31

Slide 32



## RARP server

Someone has to know the mappings - quite often this is in a file "/etc/ethers"

Since this information is generally in a file, RARP servers are generally implemented as **user processes** (because a kernel process should **not** do file I/O!)

- Unlike ARP responses which are generally part of the TCP/IP implementation (often part of the kernel).
- How does the process get the packets - since they aren't IP and won't come across a socket?
  - BSD Packet filters
  - SVR4 Data Link Provider Interface (DLPI)
  - SUN's Network Interface Tap (NIT)
  - Interestingly in the appendix to RFC 903 an alternative to having data link level access was to have two IOCTLs, one that would "sleep until there is a translation to be done, then pass the request out to the user process"; the other means: "enter this translation into the kernel table"
- RARP requests are sent as hardware level broadcasts - therefore are **not** forwarded across routers:
  - multiple servers per segment - so in case one is down; the first response is used
  - having the router answer

R. Finlayson, T. Mann, J. C. Mogul, and M. Theimer, 'A Reverse Address Resolution Protocol', *Internet Request for Comments*, vol. RFC 903 (INTERNET STANDARD), Jun. 1984 [Online]. Available: <http://www.rfc-editor.org/rfc/rfc903.txt>

Slide 33



## Alternatives to RARP

In a later lecture we will examine:

- BOOTP and DHCP (for both IPv4 and IPv6) and
- autoconfiguration for IPv6.

Slide 34



## Novel IPX/SPX Addresses

Another approach to network addresses - which are tied to the MAC address  
IPX/SPX == Internetwork Packet Exchange/Sequenced Packet Exchange  
IPX address: 32 bits of network ID and 48 bits of host ID (the ethernet address)

**Problems:**

- There is no central authority for allocating the network IDs
- So if you interconnect multiple IPX networks you may have to renumber every network
- If you change Ethernet cards, you get a new address!
- Assumes that all machines are attached to a high capacity LAN.

**Advantages:**

- You only have to assign network numbers, then the hosts figure out their own address. Simpler administration.

Novell NetWare provides: Service Advertising Protocol (SAP), Routing Information Protocol (RIP), and NetWare Core Protocol (NCP).

Slide 35



## Useful tools

For looking at and generating packets!

IK1552

SPRING 2019

SLIDE 35

## Slide 36



## tcpdump

**Under HP-UX 11.0**

```
# ./tcpdump -i /dev/dlpi0
tcpdump: listening on /dev/dlpi0
22:25:43.217866 birk2.5900 > nucmed35.50251: . ack 3089200293 win 8080 (DF)
22:25:43.290636 birk2.5900 > nucmed35.50251: P 0:4(4) ack 1 win 8080 (DF)
22:25:43.360064 nucmed35.50251 > birk2.5900: . ack 4 win 32768
22:25:43.363786 birk2.5900 > nucmed35.50251: P 4:167(163) ack 1 win 8080 (DF)
22:25:43.364159 nucmed35.50251 > birk2.5900: P 1:11(10) ack 167 win 32768
22:25:43.543867 birk2.5900 > nucmed35.50251: . ack 11 win 8070 (DF)
22:25:43.577483 birk2.5900 > nucmed35.50251: P 167:171(4) ack 11 win 8070 (DF)
22:25:43.640052 nucmed35.50251 > birk2.5900: . ack 171 win 32768
22:25:43.643793 birk2.5900 > nucmed35.50251: P 171:334(163) ack 11 win 8070 (DF)
22:25:43.644132 nucmed35.50251 > birk2.5900: P 11:21(10) ack 334 win 32768
22:25:43.750062 birk2.5900 > nucmed35.50251: . ack 21 win 8060 (DF)
22:25:43.873349 birk2.5900 > nucmed35.50251: P 334:338(4) ack 21 win 8060 (DF)
22:25:43.940073 nucmed35.50251 > birk2.5900: . ack 338 win 32768
13 packets received by filter
0 packets dropped by kernel
```

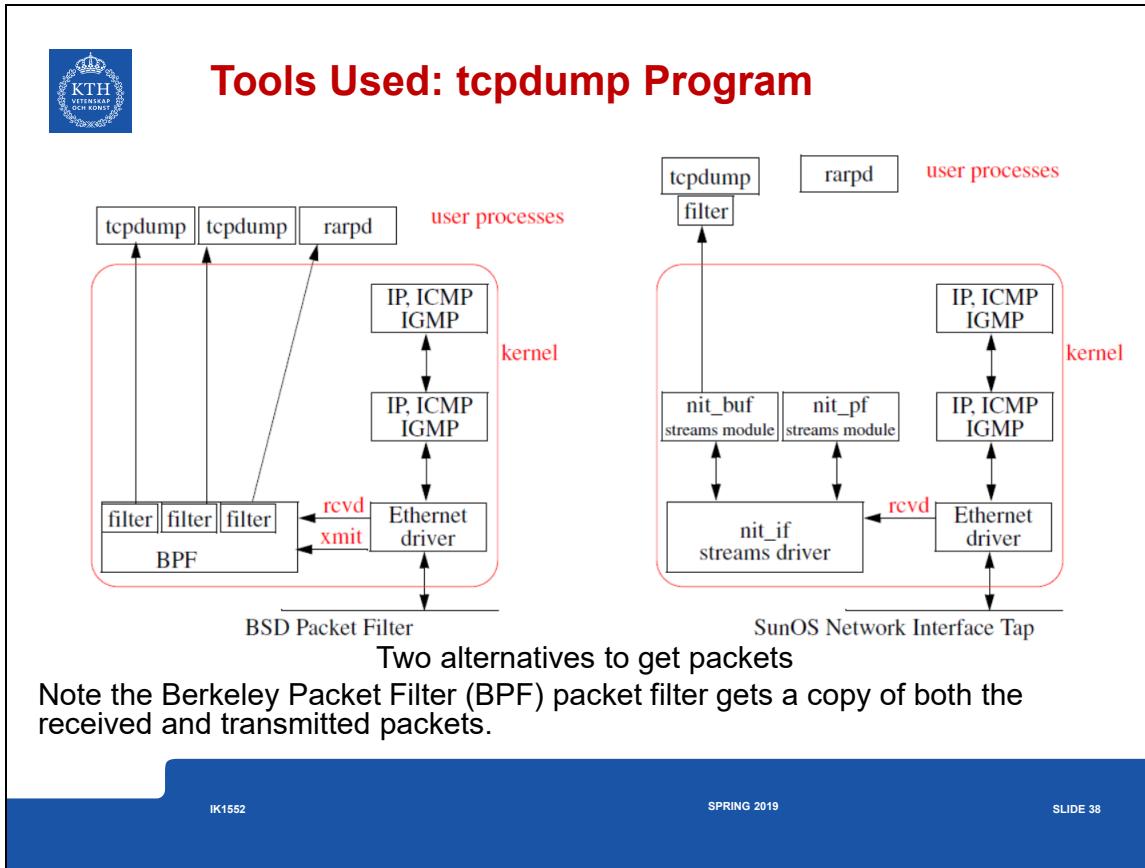
## Slide 37



## tcpdump - Linux

```
nucmed30:/home/maguire # /usr/sbin/tcpdump -i eth1
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on eth1, link-type EN10MB (Ethernet), capture size 96 bytes
14:21:52.736671 IP nucmed30.local.domain.must-p2p > jackb.ssh: P 1818006646:1818006726(80) ack 307068981 win 591>
14:21:52.737291 IP jackb.ssh > nucmed30.local.domain.must-p2p: P 1:113(112) ack 80 win 32768 <nop,nop,timestamp>
14:21:52.737917 IP nucmed30.local.domain.must-p2p > jackb.ssh: P 80:160(80) ack 113 win 5910 <nop,nop,timestamp>
14:21:52.802719 IP jackb.ssh > nucmed30.local.domain.must-p2p: . ack 160 win 32768 <nop,nop,timestamp> 25983516 2>
...
14:21:57.782196 arp who-has jackscan tell nucmed30.local.domain
14:21:57.784218 arp reply jackscan is-at 00:40:8c:30:d4:3214:21:57.784253 IP nucmed30.local.domain > jackscan: icmp 64:
echo request seq 1
14:21:57.784971 IP jackscan > nucmed30.local.domain: icmp 64: echo reply seq 1
14:21:58.782187 IP nucmed30.local.domain > jackscan: icmp 64: echo request seq 2
14:21:58.782912 IP jackscan > nucmed30.local.domain: icmp 64: echo reply seq 2
14:21:59.783036 IP nucmed30.local.domain > jackscan: icmp 64: echo request seq 3
14:21:59.783759 IP jackscan > nucmed30.local.domain: icmp 64: echo reply seq 3
...
14:21:59.802600 IP jackb.ssh > nucmed30.local.domain.must-p2p: . ack 2864 win 32768 <nop,nop,timestamp> 25984216 >
14:22:00.739485 IP nucmed30.local.domain.must-p2p > jackb.ssh: P 2864:2944(80) ack 897 win 5910 <nop,nop,timesta>
84 packets captured
84 packets received by filter
0 packets dropped by kernel
```

Slide 38



Slide 39



## Wireshark (formerly Ethereal)

First we start Wireshark capturing packets, then we ping another machine on the LAN:

A screenshot of a Microsoft Windows XP Command Prompt window. The window title is "Command Prompt". The text inside the window shows the following command and its output:

```
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\maguire>ping 192.168.1.1

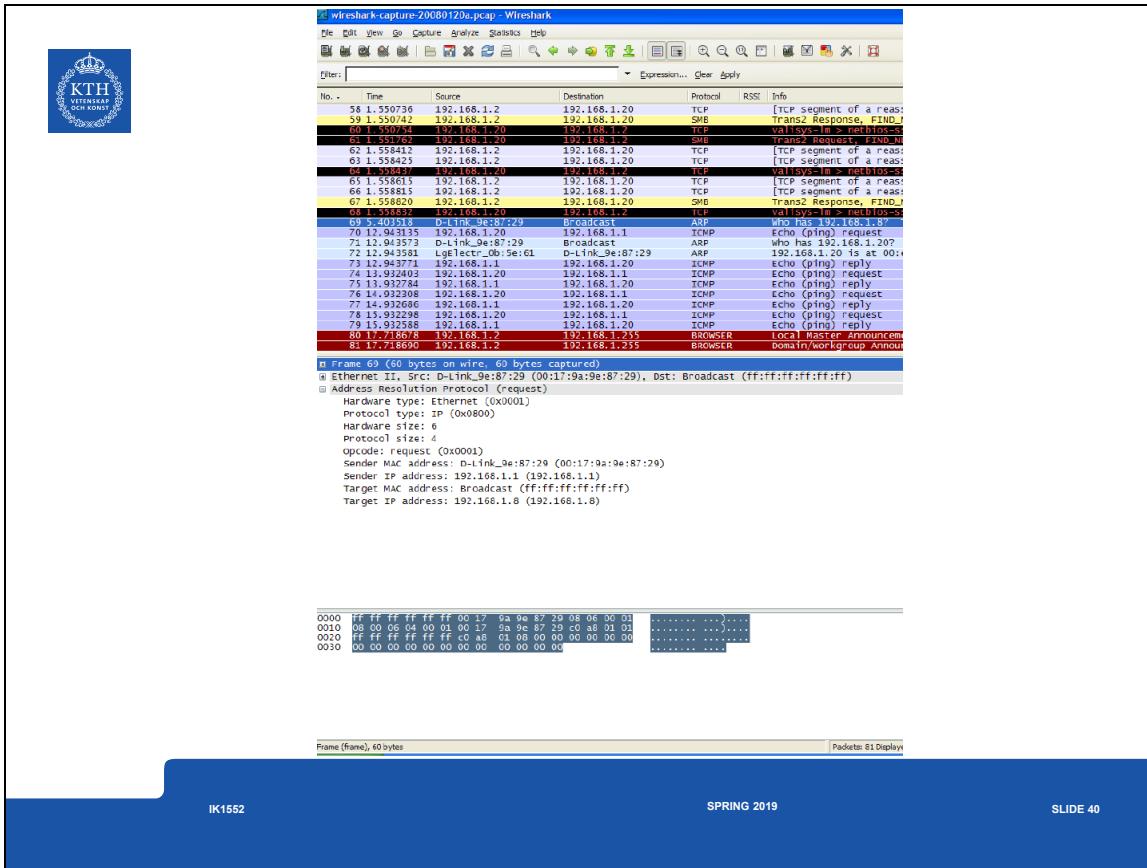
Pinging 192.168.1.1 with 32 bytes of data:
Reply from 192.168.1.1: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\maguire>
C:\Documents and Settings\maguire>
```

Ping another machine on the LAN

## Slide 40



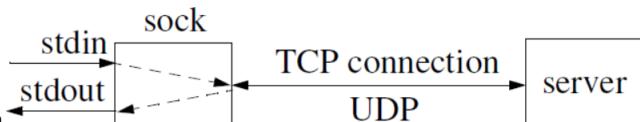
Slide 41



## Tools Used: sock Program

A simple test program to generate TCP, UDP data  
To test and debug TCP, UDP implementations

- Interactive client: defa
- Interactive server: -s
- Source client: -i
- Sink server: -i -s
- Default TCP, -u for UDP



### Source Code Available: (Tcpdump and sock)

For Win95/98/2000/NT: <http://www.winpcap.org/windump/>

For BSD like:

<http://ftp.vim.org/documents/published/books/stevens.tcpipiv1.tar.gz>

Slide 42



## Linux Socket filter

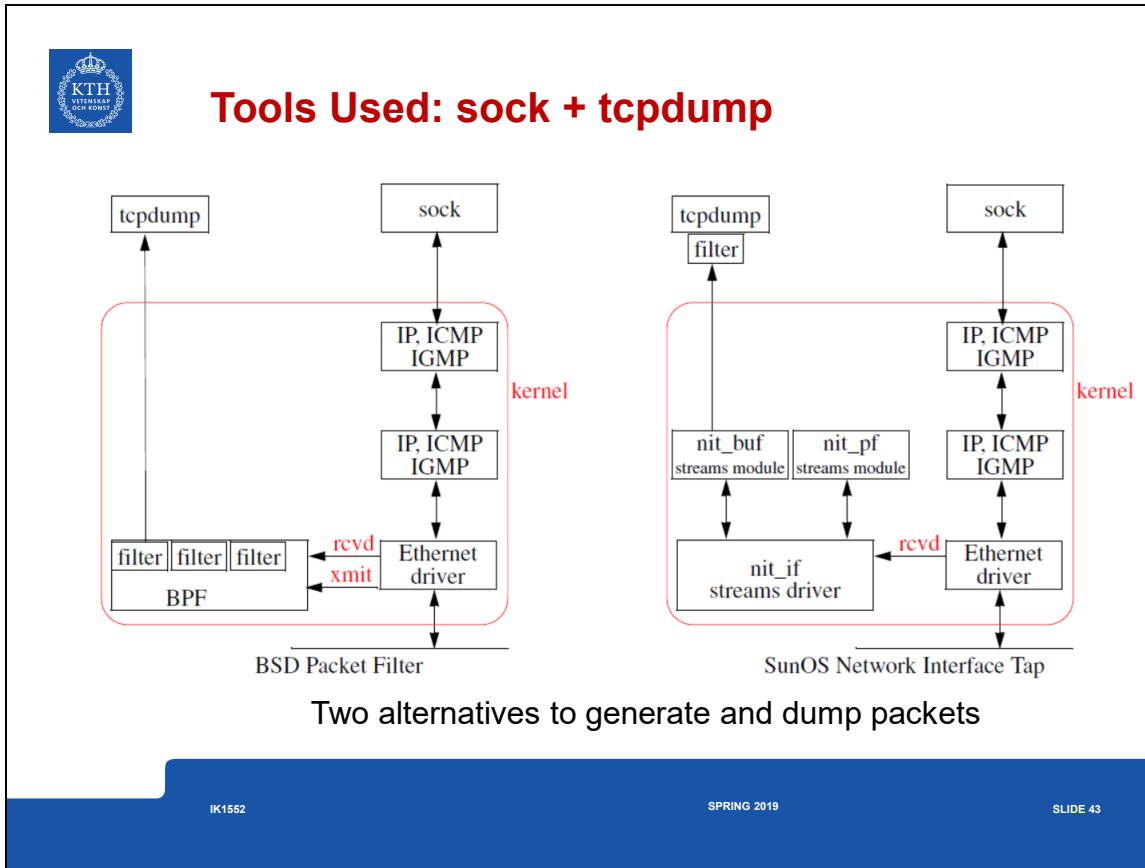
If you want to sniff the network your self (with a program) - try the Linux Socket Filter:

- Gianluca Insolvibile, 'The Linux Socket Filter: Sniffing Bytes over the Network', *Linux Journal*, 31 May 2001. Available: <http://www.linuxjournal.com/article/4659>
- Gianluca Insolvibile, 'Inside the Linux Packet Filter, Part II', *Linux Journal*, Feb. 2002 [Online]. Available: <http://www.linuxjournal.com/article/5617>.

Gianluca Insolvibile, 'The Linux Socket Filter: Sniffing Bytes over the Network', *Linux Journal*, May 2001 [Online]. Available: <http://www.linuxjournal.com/article/4659>. [Accessed: 03-Mar-2016]

Gianluca Insolvibile, 'Inside the Linux Packet Filter, Part II', *Linux Journal*, Feb. 2002 [Online]. Available: <http://www.linuxjournal.com/article/5617>. [Accessed: 03-Mar-2016]

Slide 43



## Slide 44



## Generating packets

```
/*
 * simple UDP client example
 * G. Q. Maguire Jr. <maguire@kth.se>
 *
 * 2016.03.03
 *
 * Under linux with gcc compile with:
 *   gcc Simple_UDP_client-example2.c -o Simple_UDP_client-example2
 */
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <netinet/in.h>

#define bigBufferSize 8192
#define destination_host "127.0.0.1"

/* 7 is the UDP port number for Echo */
/* 9 is the UDP port number for Discard */
#define Destination_Port 9
#define Destination_Port 52000

main(argc, argv)
int argc;
char **argv;
{
    int client_socket_fd;           /* Socket to client, server */
    struct sockaddr_in server_addr; /* server's address */

    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);
    };
}
```

IK1552

SPRING 2019

SLIDE 44

Slide 45



## Generating packets - continued

```
/* initialize the server address structure */
memset( (char*)&server_addr, 0, sizeof(server_addr));
server_addr.sin_family=AF_INET;

server_addr.sin_port=htons(Destination_Port);

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);
}

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

if ((sendto(client_socket_fd, bigBuffer, strlen(bigBuffer), sendto_flags,
           (struct sockaddr*)&server_addr, sizeof(server_addr)) ) == -1) {
    perror("Unable to send to socket");
    close(client_socket_fd);
    exit(1);
}

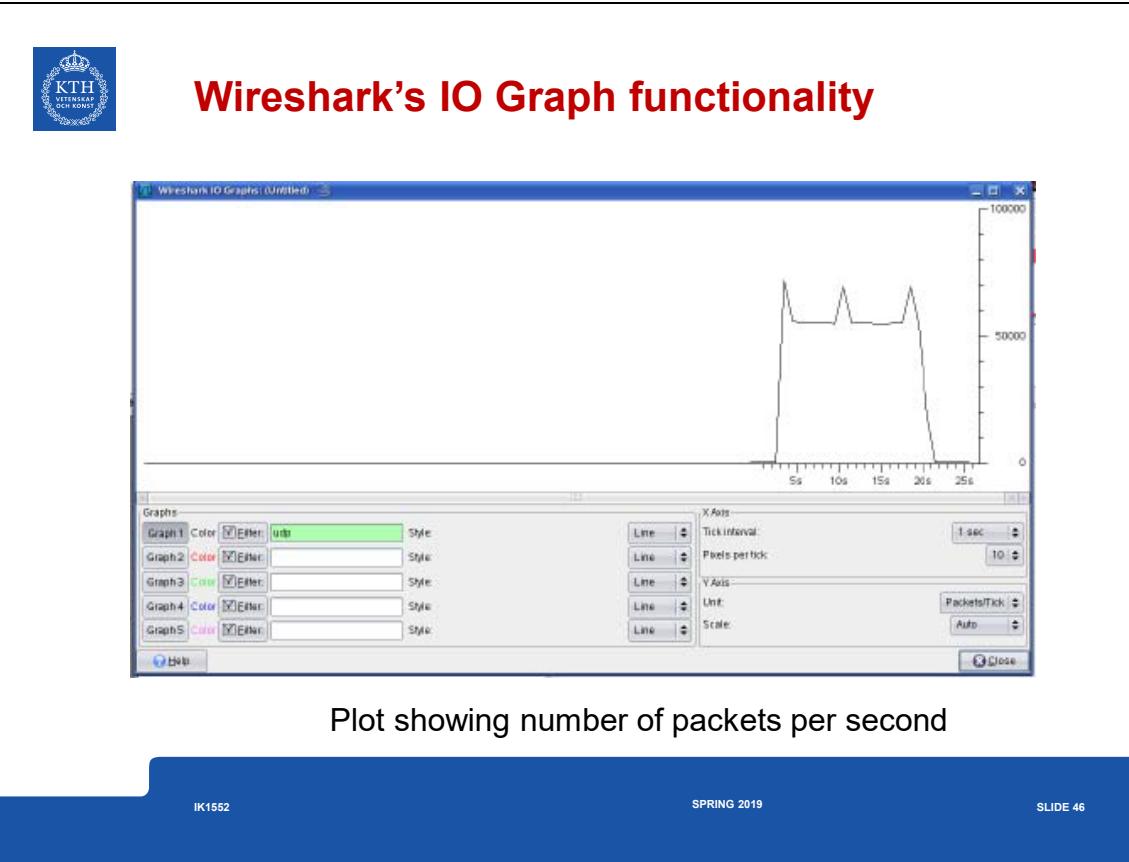
close(client_socket_fd);/* close the socket */
exit(0);
}
```

IK1552

SPRING 2019

SLIDE 45

Slide 46



Slide 47



## Some statistics on this packet trace

Microsoft Excel - z5-udp-only-with-chart

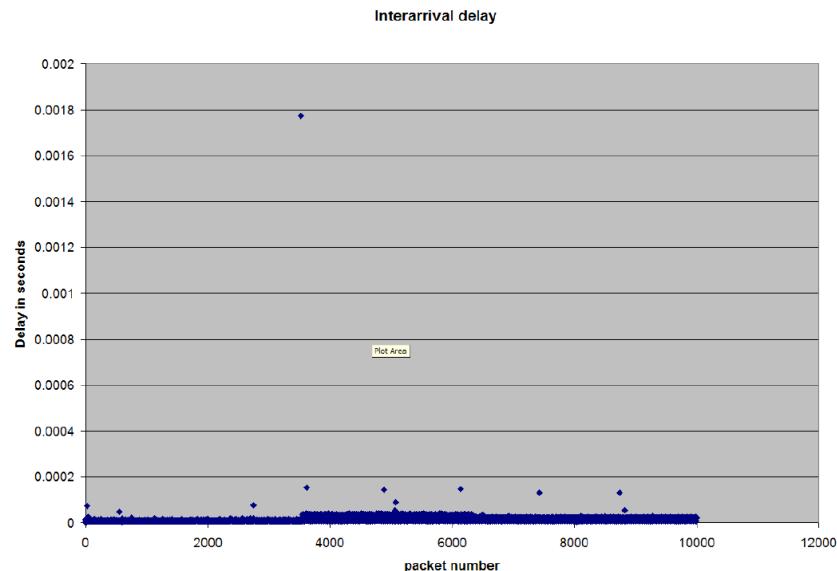
No.	Time	Source	Destination	Protocol	Info	G	H	I	J	K
2	9 5.036328	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	0	Minimum	0 00000E+00		
3	10 5.035344	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	1.6E-05	Maximum	1.772000E-03		
4	11 5.035352	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	8E-06	Average	9.714500E-06		
5	12 5.035359	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	7E-06	std	1.847505E-05		
6	13 5.035365	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	6E-06				
7	14 5.035372	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	7E-06				
8	15 5.036301	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	9E-06				
9	16 5.035389	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	8E-06				
10	17 5.035397	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	8E-06				
11	18 5.036406	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	9E-06				
12	19 5.036416	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	1E-05				
13	20 5.035427	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	1.1E-05				
14	21 5.036436	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	9E-06				
15	22 5.035443	192.168.1.130	237.11	UDP	Source port: 1028 Destination port: 9	7E-06				
23	5.036456	192.168.1.130	237.11	UDP						

IK1552 SPRING 2019 SLIDE 47

Slide 48

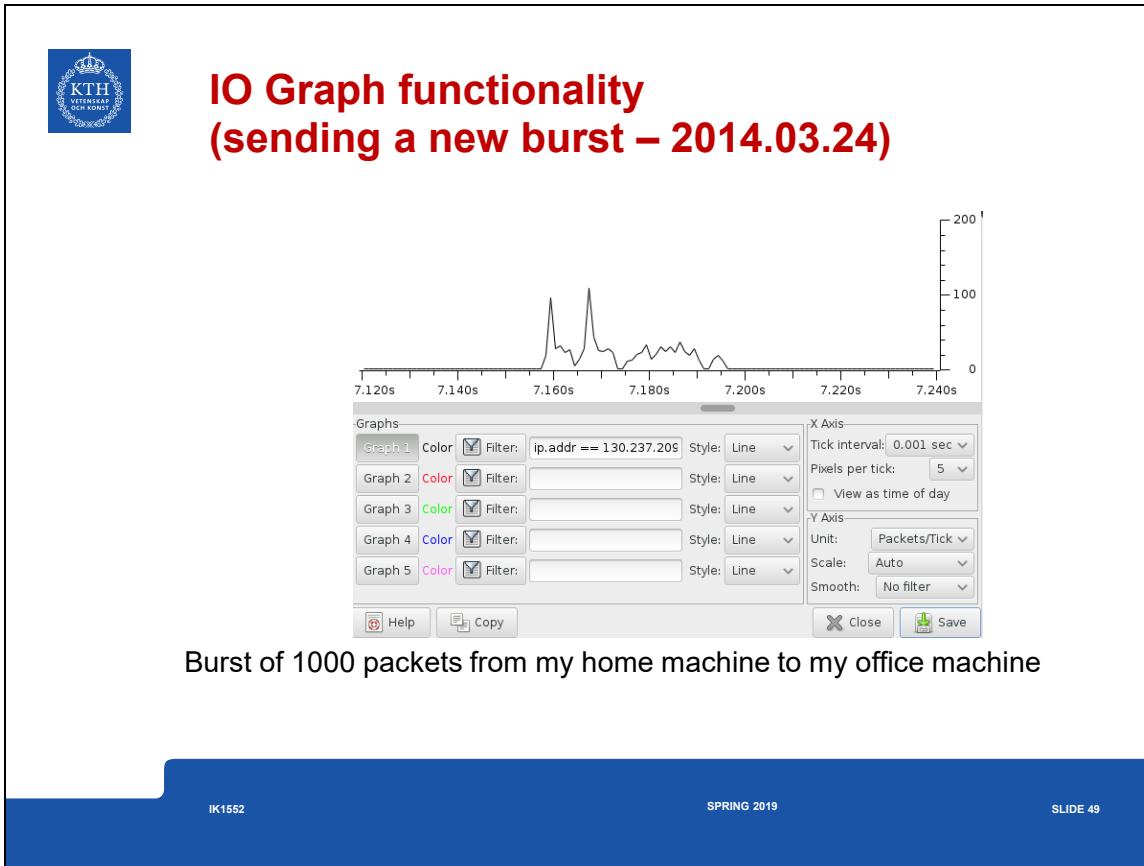


## Interarrival delay and variance



Note those packets with high or very high delays

Slide 49



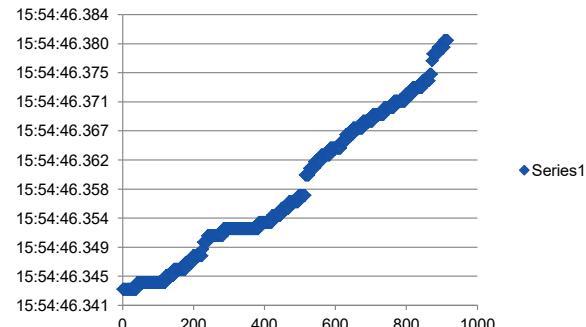
Slide 50



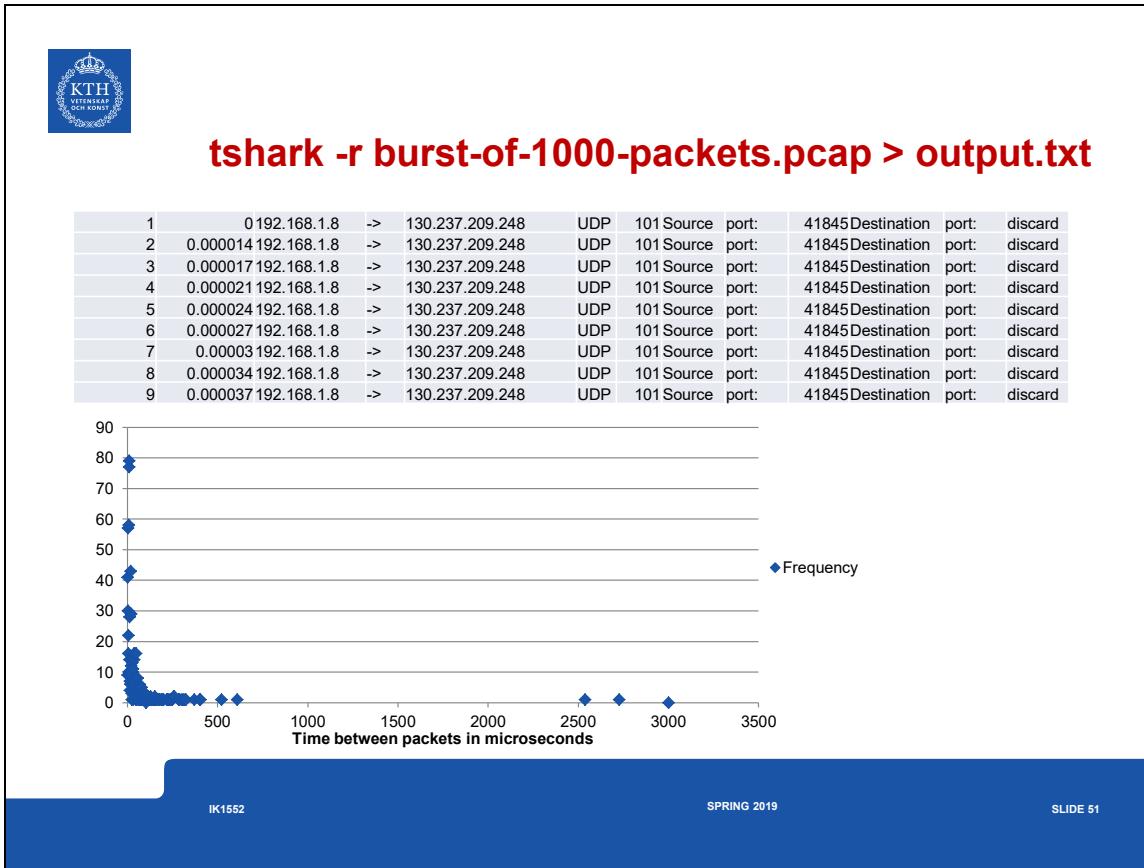
## Export as pcap file; input to tcpdump

```
/usr/sbin/tcpdump -r burst-of-1000-packets.pcap
```

```
15:54:46.343179 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.384  
15:54:46.343193 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.380  
15:54:46.343196 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.375  
15:54:46.343200 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.371  
15:54:46.343203 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.367  
15:54:46.343206 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.362  
15:54:46.343209 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.358  
15:54:46.343213 IP birkv.41845 > ccsser2.discard: UDP    length 59    15:54:46.354
```



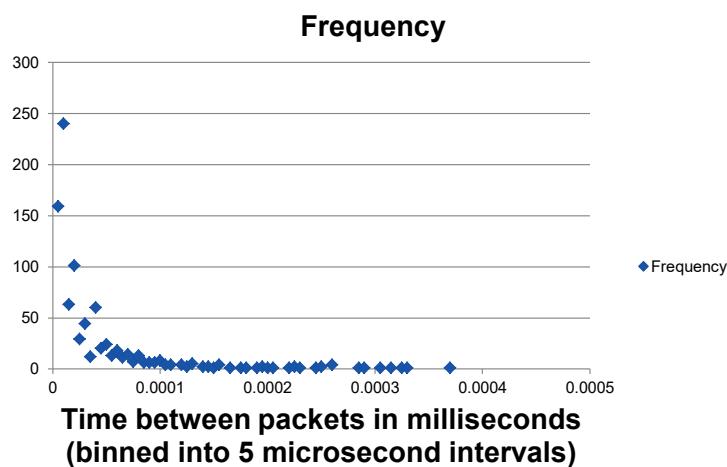
## Slide 51



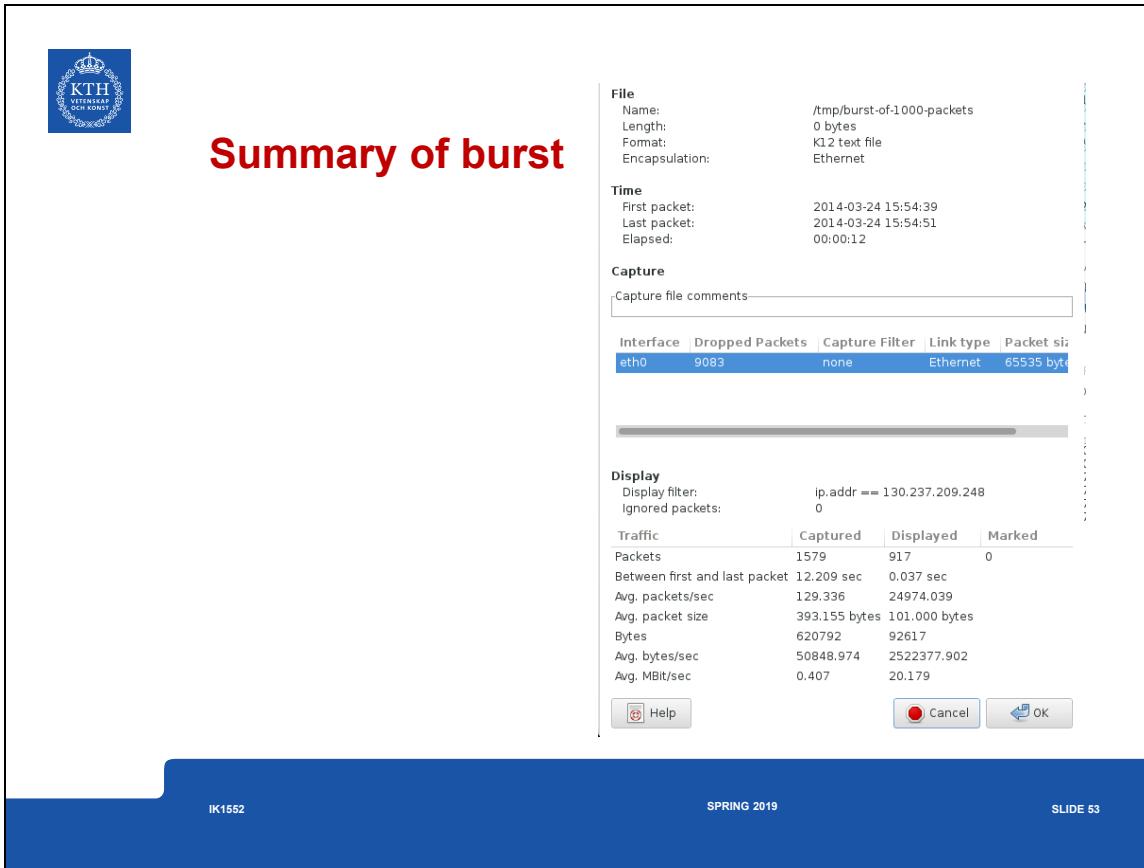
Slide 52



## A closer look at the histogram



Slide 53



Slide 54



## Traffic generators

- Distributed Internet Traffic Generator (D-ITG) [Avallone 2004, Avallone 2006] - <http://www.grid.unina.it/software/ITG/>
- Gensyn generate multiple TCP streams in parallel <http://www.item.ntnu.no/people/personalpages/fac/poulh/gensyn>
- iperf3: A TCP, UDP, and SCTP network bandwidth measurement tool <https://github.com/esnet/iperf>
- MGEN: network performance tests and measurements using UDP/IP traffic <http://www.nrl.navy.mil/itd/ncs/products/mgen> (See also <http://www.nrl.navy.mil/itd/ncs/products/>)
- RUDE & CRUDE - Real-time UDP Data Emitter (RUDE) and Collector for RUDE (CRUDE) <http://rude.sourceforge.net/>
- ostinato packet generator and analyzer <http://ostinato.org/>
- UDPGen <https://github.com/steerapi/udpgen>
- Spirent Federal Systems' SmartBits - hardware tester <http://www.spirentfederal.com/IP/Products/SmartBits/Datasheets/>

For additional traffic generators (as of Sept. 2010) see:

<http://www.icir.org/models/trafficgenerators.html>

IK1552

SPRING 2019

SLIDE 54

[Avallone 2006] Stefano Avallone, Donato Emma, Antonio Pescapè, and Giorgio Ventre, 'High Performance Internet Traffic Generators', *The Journal of Supercomputing*, vol. 35, no. 1, pp. 5–26, Jan. 2006. DOI: 10.1007/s11227-006-0798-1

[Avallone 2004] Stefano Avallone, Antonio Pescapé, and Giorgio Ventre, "Analysis and experimentation of Internet Traffic Generator", International Conference on Next Generation Teletraffic and Wired/Wireless Advanced Networking (NEW2AN'04), February 02-06, 2004 <http://ditg.comics.unina.it/D-ITGpublications/New2an-ITG.pdf>

Slide 55



## Summary

- Routing Principles
- Routing Mechanism: Use the *most specific* route
  - IP provides the mechanism to route packets
- Routing Policy: What routes should be put in the routing table?
  - Use a routing daemon to provide the *routing policy*
- Routing table
- ARP and RARP
- IPX/SPX Addresses - we will see something similar when we talk about IPv6
- tcpdump, ethereal, socat

For further information about routing see:

Bassam Halabi, *Internet Routing Architectures*, Cisco Press, 1997,  
ISBN 1-56205-652-2. -- especially useful for IGRP.

We will examine routing policies and algorithms in a later lecture.

Bassam Halabi, *Internet routing architectures*. Indianapolis, IN: Cisco Press : New Riders Pub., 1997, ISBN: 1-56205-652-2.

Slide 56



## References

1. Geoff Huston, "Analyzing the Internet BGP Routing Table", Cisco Systems web page,  
[http://www.cisco.com/en/US/about/ac123/ac147/ac174/ac176/about\\_cisco\\_ipj\\_archive\\_article09186a00800c83cc.html](http://www.cisco.com/en/US/about/ac123/ac147/ac174/ac176/about_cisco_ipj_archive_article09186a00800c83cc.html)
2. Tian Bu, Lixin Gao, and Don Towsley, "On Characterizing BGP Routing Table Growth", Proceedings of Globe Internet 2002, 2002  
[http://www-unix.ecs.umass.edu/~lgao/globalinternet2002\\_tian.pdf](http://www-unix.ecs.umass.edu/~lgao/globalinternet2002_tian.pdf)
3. Harsha Narayan, Ramesh Govindan, and George Varghese, 'The Impact of Address Allocation and Routing on the Structure and Implementation of Routing Tables', in *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, New York, NY, USA, 2003, pp. 125–136 [Online]. DOI: 10.1145/863955.863971 <http://www.cs.ucsd.edu/~varghese/PAPERS/aram.pdf>

Slide 57



# ¿Questions?

IK1552

SPRING 2019

SLIDE 57