Chapter 1: The Internet

References:

- Rainer Klute: Das World Wide Web. Addison-Wesley, 1996, ISBN: 389319763X.
- RRZN Hannover: Internet. Ein Einführung in die Nutzung der Internet-Dienste. Es gibt inzwischen die 7. Auflage. Erhältlich bei Beratung des HRZ.
- Douglas Comer: Internetworking with TCP/IP. Prentice Hall, 1988, ISBN 0134701887.
- W. Richard Stevens: TCP Illustrated, Vol. 1. Addison-Wesley, 1994, ISBN 0201633469.
- W. Richard Stevens: UNIX Network Programming, Vol. 1, 2nd Ed. Prentice Hall, 1998. [Deutsch: Programmierung von UNIX-Netzwerken, 2. Auflage. Hanser, 2000.]
- Craig Zacker: Upgrading and Troubleshooting Networks — The Complete Reference. Osborne/McGraw-Hill, 2000, ISBN 0-07-212256-0, 918 pages.
- Jochen Musch: Die Geschichte des Netzes: ein historischer Abriß. `http://www.psychologie.uni-bonn.de/sozial/staff/musch/history.htm`
- Gregory R. Gromov: The Roads and Crossroads of Internet History. `http://www.internetvalley.com/intval1.html`
- Robert H'obbes' Zakon: Hobbes' Internet Timeline v5.1 `http://info.isoc.org/guest/zakon/Internet/History/HIT.html`
- Tim Berners-Lee: Weaving the Web. Harper, 1999, ISBN: 0062515861, 226 pages.
- Dan Connolly, Robert Cailliau: A Little History of the World Wide Web. `http://www.w3.org/History.html`
- T. Socolofsky, C. Kale: A TCP/IP Tutorial. RFC 1180.
- Thanks to Leonhard Knauff from the computing center for many explanations!

Objectives

After completing this chapter, you should be able to:

- explain why the internet is a "network of networks".
- write one page about the history of the internet.
- explain numeric IP addresses and port numbers.
- explain the notion of a "protocol stack".
- explain the main characteristics of TCP and IP,
- know where to get RFCs (and what they are).
- select an ISP (internet service provider).
- write simple networking programs with sockets.

Overview

The Internet (1)

- The internet is a system of interconnected networks that uses protocols of the TCP/IP family.
- In January 2008, about 542 million computers were connected to the internet [`http://www.isc.org/ds/`].

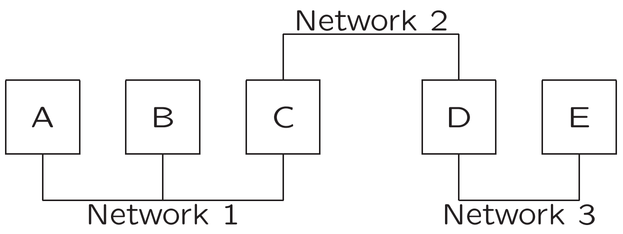
  This is the number of mappings from IP-numbers to names. Jan 2000: 72 million, Juli 1995: 5–8 million, October 1990: ~313000. `.com`: 95 mio, `.net`: 190 mio, `.edu`: 11 mio, `.org`: 1 mio, `.de`: 21 mio.
- Based on packet switching.

  Pieces of data from A to B can take different paths. Packet switching is opposed to circuit switching (dedicated path for entire connection).
- No central control, redundant connection paths.

The Internet (2)

- Although the "internet" is today sometimes used synonymous to the world wide web, the WWW is only one internet application.
- The internet supports also other services: Email, FTP, Telnet, News, Finger, Talk, DNS, NFS, ...
- Web browsers support some of these services, and an important reason for the success of the web was its integrating function.
- Web servers are accessed via the HTTP protocol.

"Network of Networks"



- If machine A wants to send data to machine E, the data packages must be relayed via machines C and D.

  Depending on which network protocol layer the transmission is done, machines C and D are called "bridge", "router", or "gateway".

Internet Addresses (1)

- Internet addresses (v4) are 32-bit numbers, written as sequence of four bytes, e.g. `141.48.14.50`.

  Symbolic names (like `haendel.informatik.uni-halle.de`) are translated into such numbers by the "domain name system" (see Chapter 2).

- Internet addresses uniquely identify a computer.

  It is not possible that two machines on the internet have at the same time the same internet address. But if computers are connected via modem, the ISP can dynamically assign internet addresses. After the connection is terminated, another computer can get the same address.

- One machine can have multiple internet addresses.

  E.g. if it has several network cards. "Multihomed".

Internet Adresses (2)

- Internet addresses are also called IP addresses because they are used in the IP protocol (see below).
- Internet addresses consists of a network part and a host part (which identifies the machine within the network). Often also network, subnet, host.

  In earlier times, several classes of IP addresses were distinguished: E.g. a class A address (with first byte 0–127) consists of one byte as network part and three bytes to identify the machine within the network. Class B addresses (first byte 128–191) are divided two bytes/two bytes, class C addresses (first byte 192–223) 3 bytes/1 byte. Because of the shortage of internet addresses, this scheme has become obsolete. Today, the border can be at any bit position. One writes e.g. `141.48.0.0/14` (14 bits network, 18 bits host within network).

Internet Adresses (3)

- IP addresses are assigned by the IANA (Internet Assigned Numbers Authority) [`http://www.iana.org`].
- It has distributed the work among three regional registries: APNIC for Asia-Pacific, ARIN for America, and RIPE NCC for Europe [`http://www.ripe.net`].
- The RIPE NCC distributes the work to Local Internet Registries (LIR), e.g. the "DFN Verein" (German Research Network) [`http://www.dfn.de`].

Internet Adresses (4)

- All large ISPs (Internet Service Providers) are also LIRs.
- Normally one gets the IP number from one's ISP.
- It is important for the routing of data packages between networks that the IP numbers are related to the internet connection.

  If they were randomly distributed, routing tables were much larger. If A can only be reached via B, they should have a common prefix.
- There are officially unregistered numbers, e.g. `192.168.x.y`.

Internet Adresses (5)

- The University of Halle was assigned the network address `141.48.0.0/16` (by the "DFN Verein").
- The last 16 bits are assigned by the computing center of the university.
- The computing center often uses 10 bits as subnet address and 6 bits as host address within the subnet, but this varies.

  There are small and large subnets. Earlier, basically very building had its own subnet. Today, VLANs (virtual local area networks) are implemented by means of switches (e.g. all computers of the library are in one subnet, no matter in what building they are located).

Internet Addresses (6)

- For the global routing of data packages only the network part is important, within the university the subnet part is used.

  The subnet mask which must be specified in some programs covers network and subnet part, e.g. in our university it is typically `255.255.255.192` (the last byte is `11000000`).
- If two IP addresses agree in the network part and in the subnet part, the computers are connected to the same network.

  I.e. they can directly communicate. Data packages are not sent via a router.

Internet Addresses (7)

- If a computer is connected to a new network, it needs a new IP address.
- Addresses with all bits 1 or 0 have special meaning:
  - ◇ All bits 1 in the host part: Broadcast.
  - ◇ All bits 0 in the host part: Identifies network.
  - ◇ `127.0.0.1`: Internal host loopback address.
- The `whois` database contains e.g. the information who is responsible for an internet address:

```
UNIX> whois -h whois.ripe.net 141.48.0.0
http://www.ripe.net/perl/whois
```


Connection of the University

- The University is connected with 1 GBit/s to the X-WiN managed by the “DFN Verein”.

  The connection is to the next core network node in Leipzig. Halle has a “cluster connection” together with the University of Applied Science Merseburg. That means that there is only a single connection point to the X-WiN. Note that the maximal bandwidth of 1 GBit/s is only for the connection to the X-WiN. If there is a lot of traffic within the X-WiN or from the X-WiN to the global Internet, the actual bandwidth might be less.

- The University pays 203.847 Euro per year to the “DFN Verein” (connection type I10, cluster).

  These are 558 Euro per day including Saturdays and Sundays. Per Minute the price is 39 Cent.

Old Data (from 2004)

- The University is connected with 155 MBit/s to the G-WiN managed by the “DFN Verein”.
- The University pays 204.517 Euro per year to the “DFN Verein” (connection type I10).

  Plus sales tax (VAT). These are 650 Euro per day including Saturdays and Sundays. Per Minute the price is 45 Cent.

- The data transfer volume is limited to 6000 GByte (incoming) per Month. It was sometimes violated.

  The Univerity then gets a warning. If the violation occurs several months in a row and the University does not order a more expensive connection, it might happen that the university is either cut off after the 6000 GB are reached or the connection is made extremely slow.

X-WiN (1)

- The DFN association has more than 300 members (universities, research centers).
- It has a meeting/conference twice a year, and the slides of the presentations are online.

  [http://www.dfn.de/de/beratung/betriebstagung/vortraege-bt/].
  For a current map of the X-WiN, see
  [http://www.dfn.de/fileadmin/3Beratung/Betriebstagungen/bt48/plenum-xwin-hma.pdf]
  See also: [http://www.aufdemhoevel.de/xwin.htm]

- Core network: Multi Gigabit, about 50 nodes.

  In January 2008, the data volume was 4641 TB (measuring data “exported” from the net during this month).

X-WiN (2)

- For many connections, DFN has rented "dark fiber" (i.e. simply the fiber) and operates its own hardware to send data through the fiber.

  The bandwidth of a fiber can be increased with newer WDMs ("wavelength division multiplexer"). E.g. 16 or 32 different wavelengths are used, each with 10 GBit/s. The (Dense-)WDMs "Optix BWS 1600g" used in many core nodes support up to 160 wavelengths with 16 Gbit/s. The fiber is rented (among others) from KPN and from GasLINE. Because gas pipelines are usually well protected from accidental damage by construction measures, this is also a good place to put fiber cables for communication. A usual cable contains 144 fibers.

- For some connections, DFN has rented only a wavelength in a fiber operated by another provider.

X-WiN (3)

- The central nodes are Hannover, Potsdam, Frankfurt, Erlangen. Of these, every pair is connected.

  This refers to the IP network (routing decisions). It does not necessarily mean that there is a single direct fiber between these nodes (because of loss during transmission, a fiber cannot be too long).

- Other core nodes are connected with at least two other core nodes.

  Usually, there is a connection from a central node via a sequence of core nodes to a different central node. E.g. there is a connection from Erlangen to Ilmenau, Jena, Leipzig, Chemnitz, Dresden and then to Potsdam. But there are sometimes additional connections, e.g. Leipzig has also a direct connection to Erlangen besides the connections to Jena and Chemnitz. Dresden is additionally connected to Erlangen.

X-WiN (4)

- The connections to other networks are, e.g.:
  - ◇ via GÉANT (DANTE) to European and north american reserach networks,
  - ◇ via DE-CIX to commercial Internet providers in Germany (Gigabit Ethernet),

    DE-CIX is located in Frankfurt [http://www.decix.de]. This is a settlement-free exchange point, i.e. the interconnected networks exchange network data packages for mutual benefit without paying each other ("peering"). X-WiN has also other connections to T-Internet and local ISPs.
  - ◇ via Global Crossing and TELIA worldwide ("global upstrem").

Traceroute Command (1)

- `traceroute` shows computers that act as routers on the way to the destination host.
- Example: "`traceroute www.tu-clausthal.de`" shows that the connection from MLU Halle to TU Clausthal is via Leipzig, Essen, Hannover, Göttingen.

  The output is shown on the next slide.
- Today, firewalls usually block the messages from routers within the university/company network.
- Traceroute prints "*" if it does not get an answer from a certain network distance (number of hops).

Traceroute Command (2)

traceroute www.tu-clausthal.de
on **haendel.informatik.uni-halle.de** (141.48.14.50):

1	141.48.14.254	(141.48.14.254)	0.3 ms
2	141.48.0.254	(141.48.0.254)	0.4 ms
3	wincisco.urz.uni-halle.de	(141.48.25.10)	0.4 ms
4	ar-leipzig2.g-win.dfn.de	(188.1.35.5)	3.4 ms
5	cr-leipzig1-ge8-0.g-win.dfn.de	(188.1.70.1)	3.1 ms
6	cr-essen1-po1-0.g-win.dfn.de	(188.1.18.105)	12 ms
7	cr-hannover1-po0-2.g-win.dfn.de	(188.1.18.50)	17 ms
8	ar-goettingen1-po0-0.g-win.dfn.de	(188.1.88.74)	19 ms
9	7200vxr.rz.tu-clausthal.de	(139.174.251.9)	21 ms
10	sr-backbone.rz.tu-clausthal.de	(139.174.254.2)	24 ms
11	*		
12	*		

Traceroute Command (3)

- `traceroute` works by sending UDP-packets that contain a short "time to live" (number of hops).

  Like the more common TCP, UDP ("user datagram protocol") is based on IP ("internet protocol"), i.e. it is a layer on top of IP. In contrast to TCP, UDP is only a slight abstraction of IP and can be used when only short messages must be sent without reliability.

- Every router decrements the "time to live" of the packet. If it reaches 0, the packet is not forwarded.
- Most routers will send an error message back. In this way, `traceroute` gets the router's IP-address.

  Error messages are sent via ICMP, the "internet control message protocol".

Traceroute Command (4)

- First, it sends a packet with "time to live" 1. In this way, it gets an error message from the first router.
- Then, it sends packets with "time to live" 2, 3, and so on until it gets an answer from the destination.

  It stops with 30 (in case the destination host does not send error messages). This maximum value can be configured with option `-m`. The "time to live" is shown in the first column of the `traceroute` output.

- For each "time to live", three packets are sent and all three roundtrip times are shown.

  The times vary a bit. On these transparencies, only one time is shown.

Traceroute Command (5)

- If no error message arrives within 5 seconds: "*".

  Not all routers send error messages, or the error message might be suppressed by a firewall.

- Since the measurements are done one after the other, it is possible that a router that is more hops away responds faster.

  The network load may have decreased between the measurements. In theory, it is also possible that routing changes between measurements.

- Some organisations permit to `traceroute` via a web interface, see [`http://www.traceroute.org`].

  In this way, `traceroute` can be used from different places in the world.

Traceroute Command (6)

`traceroute www.sis.pitt.edu`:

1	141.48.14.254	(141.48.14.254)	0.3 ms
2	141.48.0.254	(141.48.0.254)	0.3 ms
3	wincisco.urz.uni-halle.de	(141.48.25.10)	0.5 ms
4	ar-leipzig2.g-win.dfn.de	(188.1.35.5)	2.8 ms
5	cr-leipzig1-ge8-0.g-win.dfn.de	(188.1.70.1)	3.9 ms
6	cr-frankfurt1-po10-0.g-win.dfn.de	(188.1.18.189)	203.6 ms
7	dfn.de1.de.geant.net	(62.40.105.1)	9.7 ms
8	de1-1.de2.de.geant.net	(62.40.96.130)	9.7 ms
9	abilene-gw.de2.de.geant.net	(62.40.103.254)	105.4 ms
10	beast-abilene-p3-0.psc.net	(192.88.115.125)	114.9 ms
11	bar-beast-ge-0-1-0-1.psc.net	(192.88.115.17)	115.4 ms
12	pitt-cl-i2.psc.net	(192.88.115.151)	114.9 ms
13	cl2-vlan712.gw.pitt.edu	(136.142.2.162)	115.4 ms
14	*		

Traceroute Command (7)

`traceroute paradox.sis.pitt.edu` from Gießen in 2000:

1	cisgis.uni-giessen.de	(134.176.24.40)	0.3 ms
2	cisgis-vlan-1.uni-giessen.de	(134.176.253.1)	1.4 ms
3	ar-marburg2.g-win.dfn.de	(188.1.42.133)	2.1 ms
4	ar-marburg1.g-win.dfn.de	(188.1.81.1)	1.9 ms
5	cr-frankfurt1.g-win.dfn.de	(188.1.80.45)	3.4 ms
6	ir-frankfurt2.g-win.dfn.de	(188.1.80.38)	3.4 ms
7	dfn.de1.de.geant.net	(62.40.103.33)	4.9 ms
8	de1-1.de2.de.geant.net	(62.40.96.130)	3.7 ms
9	62.40.103.254	(62.40.103.254)	84.6 ms
10	clev-nycm.abilene.ucaid.edu	(198.32.8.29)	96.8 ms
11	abilene.psc.net	(192.88.115.122)	100.1 ms
12	pitt.psc.net	(198.32.224.8)	102.7 ms
13	136.142.2.101	(136.142.2.101)	103.2 ms
14	paradox.sis.pitt.edu	(136.142.116.28)	104.8 ms

Further Commands (1)

- `ping` checks whether a host is reachable.

  It does so by sending an `ECHO_REQUEST` message via ICMP. With the option `-s`, it sends one message per second and prints timing statistics when one presses `Crtl`+`C`. One can use e.g. `ping -v -t 1 www.acm.org` to send a message with "time to live" 1 and print the error message from the first router (in case one has no `traceroute`). With the option `-U`, the UDP protocol is used (the port can be set with `-p`).

- `ping www.acm.org`

  `www.acm.org is alive`

- `ping 134.176.28.94`

  `no answer from 134.176.28.94`

Further Commands (2)

- `netstat -r`

  shows the routing table of the local computer.

- E.g. `netstat -rn` on `141.48.14.50` shows:

Destination	Gateway	Flags	Ref	Use	Interface
141.48.14.0	141.48.14.50	U	1	7380	eri0
224.0.0.0	141.48.14.50	U	1	0	eri0
default	141.48.14.254	UG	1	16025	
127.0.0.1	127.0.0.1	UH	5	527	lo0

The option `-n` means that numeric IP-addresses are used instead of host names. Of course, the output is more interesting on a machine that is connected to multiple networks. `141.48.14.254` is the gateway that connects the local network to the university net and further. `224.0.0.0` is a multicast address (`BASE-ADDRESS.MCAST.NET`). Under "Flags", `U` means "up" and `G` means "gateway", `H` the local host (?).

Further Commands (3)

- `netstat` shows all network connections that currently exist on the machine.
- `netstat -i` and `ifconfig -a` show information about the network interfaces.

  Or use e.g. `ifconfig eri0` for the network interface `eri0`.
  "mtu" is the "maximum transfer unit" (1500 for ethernet).
- `arp wega` shows the ethernet address of `wega`.
- For more information, see e.g. `man netstat`.
- Under Windows, try: `ping`, `netstat`, `ipconfig`, `route`, `arp`, `tracert` (traceroute).

Overview

History of the Internet (1)

- 1958: ARPA founded (reaction to Sputnik 1957).

  ARPA or DARPA: Advanced Research Projects Agency of the US Department of Defense. 1962 J.C.R. Licklider became head of the computer research program at DARPA. He envisioned a globally interconnected set of computers.
- 1961–68: Packet-oriented network protocols developed.

  L. Kleinrock (MIT) published the first paper on packet-switching theory (1961). Also researchers at RAND (Paul Baran, 1964) and NPL developed packet switching networks.
- 1968: BBN gets contract for ARPANET hardware.

  IMPs: Interface Message Processors.

History of the Internet (2)

- 1969: ARPANET starts, 4 universities connected.

  University of California at Los Angeles (UCLA), University of California at Santa Barbara (UCSB), Stanford Research Institute (SRI), University of Utah. Each had a different hardware! The connection was via the IMPs with 50 KBit/s lines from AT&T.
- 1970: ALOHAnet (first packet radio network).
- 1972: ARPANET with about 40 computers is presented on the First International Conference on Computer Communications.

  ARPANET offered remote login, file transfer, and electronic mail. Electronic mail was introduced in 1971/1972 (after `telnet` and `ftp`). According to a 1973 study, Email was 75% of all ARPANET traffic.

History of the Internet (3)

- 1972: InterNetwork Working Group founded.

  Goal: Common protocol for international connections between autonomous networks. Vinton Cerf appointed first chair.

- 1973: First international ARPANET connections.

  With the University College of London via NORSAR (Norway).

- 1973/74: TCP protocol specified (Kahn/Cerf).

  The separation into TCP and IP was done later (1978).

- 1976/77: UUCP: Unix-to-Unix-Copy Protocol.

  Developed by AT&T Bell Labs (Mike Lesk and others), distributed with UNIX V7. "Poor Man's ARPANET".

History of the Internet (4)

- 1979: USENET/News established, based on UUCP.

  Between Duke University and the University of North Carolina.

- 1981: CSNET founded.

  Only Universities with ARPA projects had access to ARPANET. CSNET ("Computer Science Network", later "Computer and Science Network") was founded with money from NSF to provide networking services to scientists with no ARPANET connection.

- 1981: BITNET founded.

  The first connection was between the City University of New York and Yale University using IBM's Network Job Entry protocol. EARN (European Academic and Research Network) was founded 1983 on a very similar basis (gateway to BITNET 1983). IBM funding.

History of the Internet (5)

- 1982: EUnet (European UNIX Network) founded.
- 1983 (January 1): ARPANET switches to TCP/IP.
- 1983: Gateway between CSNET and ARPANET.
- 1983: ARPANET split into ARPANET and MILNET (68 of the 113 nodes went to MILNET).
- 1983: The source code of TCP/IP implementation in 4.2 BSD is made publicly available.

  Developed by the University of California at Berkeley funded by DARPA. Many vendors use this source code (helps to spread TCP/IP, improves portability).

History of the Internet (6)

- 1983/84: Name server, Domain Name System developed.
- 1986: NSFNET founded (Backbone 56 Kbit/s).
- 1988: EUnet switches to TCP/IP.
- 1988: NSFNET Backbone T1 (1.544 Mbit/s).
- 1989: More than 100 000 computers are connected to the internet.
- 1990: ARPANET is taken out of service, NSFNET takes over its role.

History of the WWW (1)

- 1945: Vannevar Bush proposes a machine "Memex" for links between documents on microfiche.
- 1963–68: Doug Engelbart develops the "oNLine System" (NLS), he invents the mouse for it.
- 1965: Ted Nelson invents the notion "Hypertext".
- 1979: Charles Goldfarb invents SGML.
- 1980: Tim Berners-Lee develops a hypertext program "Enquire" while he is at CERN for half a year.

  CERN: European Center for particle physics.

History of the WWW (2)

- 1987: CERN is connected to the internet.
- 1989: Tim Berners-Lee writes a proposal for a global hypertext project at CERN.

  Robert Caillian independently developed a proposal for a hypertext project. 1990 they develop together a new version.
- 1990: From October to December Tim Berners-Lee develops the first Web-Server "`httpd`" and the first browser "`WorldWideWeb`" on a NeXT machine.
- 1991 (summer): The program "`WorldWideWeb`" is made publicly available on the internet.

History of the WWW (3)

- 1991–93: Tim Berners-Lee coordinates the further development of URIs, HTTP, and HTML.
- 1993 (January): About 50 WWW/HTTP servers.

  In October already over 200, in June 1994 over 1500, in November 1995 about 73 500.
- 1993 (Feb.–Aug.): Mosaic is released.

  Mosaic provided "a consistent and easy-to-use hypertext-based interface to a wide variety of information sources, including Gopher, WAIS, World Wide Web, NNTP/News, Techinfo, Textinfo, FTP, local file systems, telnet, tn3270, and others." Developed by Marc Andreessen and Eric Bina at the NCSA (National Center for Supercomputing Applications). Mosaic is released for X, PCs, and Macintosh.

History of the WWW (4)

- 1993: NCSA also offers a web server, `NCSA HTTPD`.

  In December 1993, a long story about WWW and Mosaic appeared in The New York Times. By the years's end, more than a thousand copies of Mosaic and HTTPD were downloaded per day from NCSA.
- 1994 (May): First International WWW Conference.

  The conference is held at CERN. In October already the second conference takes place (in Chicago).
- 1994: Marc Andreessen and others leave NCSA to found "Mosaic Communications Corp" (Netscape).

History of the WWW (5)

- 1994: W3 Consortium founded, Tim Berners-Lee becomes its director [http://www.w3.org].

  The W3C is located at the MIT. CERN decided for a new particle accelerator and had no money to develop the WWW further.

- 1995: Sun Microsystems presents HotJava, a browser that contains interactive objects.
- 1995: Microsoft buys the web browser developed by a company called Spyglass (using the Mosaic code) and turns it into Internet Explorer.

Overview

Requests for Comments (1)

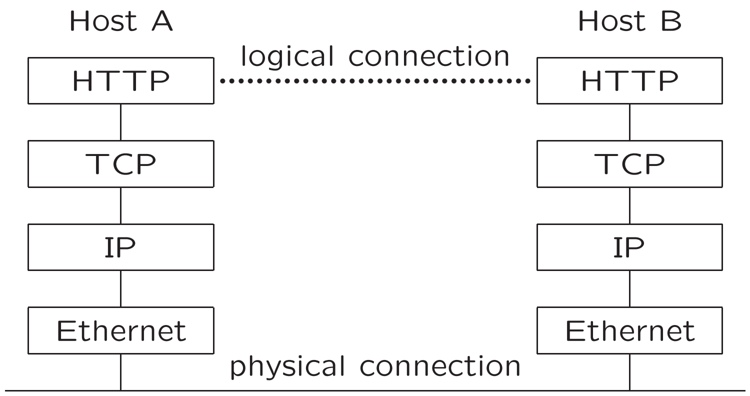
- "Requests for Comments" (RFCs) are documents about the internet, which contain e.g. specifications of protocols, tutorials, historical information.
- RFCs are available in many places on the internet, e.g. [http://www.rfc-editor.org], [http://www.cis.ohio-state.edu/cs/Services/rfc/index.html]
- The official RFCs are always plain ASCII, but sometimes one can get versions that were converted to HTML.

Requests for Comments (2)

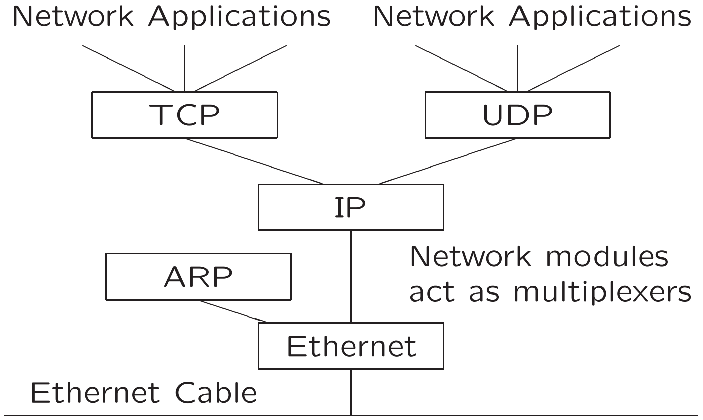
- Exercise: Get one of the following RFCs:
  - ◇ RFC 1122: Requirements for Internet Hosts I
  - ◇ RFC 1180: A TCP/IP Tutorial.
  - ◇ RFC 1630: Universal Resource Identifiers.
  - ◇ RFC 1866: Hypertext Markup Language — 2.0

    Only of historical interest. Newer standards: [http://www.w3.org].
  - ◇ RFC 1945: HTTP/1.0.
  - ◇ RFC 2324: Hyper Text Coffee Pot Control Protocol.
  - ◇ RFC 2616: HTTP/1.1.

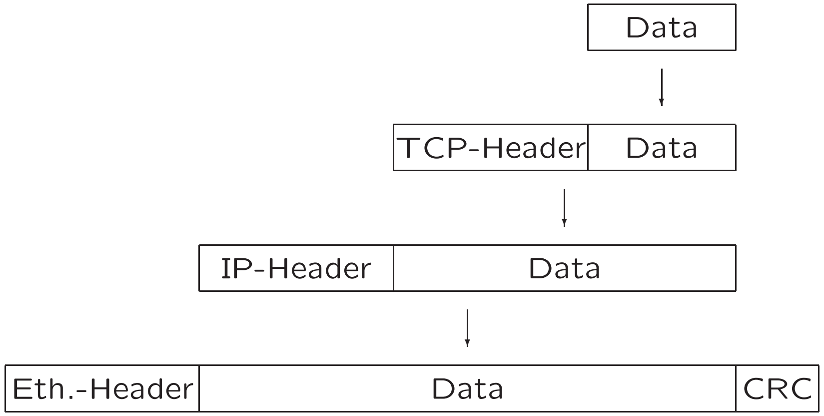
Protocol Stack



Protocols on one Machine



Framing of Data Packets



Comparison with OSI-Model

OSI	TCP/IP	
Application	Application	Telnet, FTP, HTTP, etc.
Presentation		
Session	Transport	TCP, UDP
Transport		
Network	Network	IP, ICMP
Data Link	Link	Device Driver Hardware
Physical		

Ethernet (1)

- Common network protocol for LANs (Local Area Networks).

  However, there are many other such protocols. The IP protocol is not bound to Ethernet, it can also build on other protocols.

- Bus topology: One machine sends on the network cable, all other machines receive.
- Since each data packet contains the destination address, the other machines can ignore it.

  However, it is possible to listen to all packets sent on the ethernet cable. There are programs that do this and search e.g. for passwords.

Ethernet (2)

- It is possible that two machines start at the same time to send data on the network cable (collision).

  A small number of collisions is normal.

- However, while they send they also listen on the network cable and detect the interference by the other machine.
- Then they wait a short time and try it again.

  The time is randomized so that the probability is small that they both start again at the same time. If that should happen, the possible time interval is doubled each time ("binary exponential backoff").

Ethernet (3)

- It is important that the collision is detected before a machine finishes to send its packet.
- This is ensured by restricting the cable length and requiring a minimum packet length.

  Electrical signals travel on a copper cable with 2/3 of the speed of light (200 km/ms). The maximum length of the classical "thick" ethernet cable is 2500 m (5 segments of 500 m each, connected by repeaters that amplify the signal). The worst case is that the machine at the other end starts sending just before the signal reaches it, thus each packet must be at least 5 km/200(km/ms), i.e. 0.025 ms long. At 10 Mbit/s, i.e. 10000 bit/ms, this means the minimum length of a packet must be 250 bit. The shortest valid ethernet packet is 72 bytes long, which is more than double the theoretical minimum.

Ethernet (4)

- Computers are identified on the ethernet via 48 bit addresses, e.g. `8:0:20:12:5a:f6`.

  Ethernet addresses are usually written as 6 bytes in hexadecimal notation, separated by colons. More generally, globally unique addresses in some network protocols on the OSI Level 2 layer including Ethernet are called MAC (Media Access Code) or MAC (MA Control) address.

- The address is defined in the network card and normally not configurable.

  The first three bytes are assigned by the IEEE to the card manufacturer, the last three bytes are assigned by the card manufacturer. If a computer gets a new ethernet card, it automatically gets a new ethernet address, but it can keep its IP address. For many cards, the ethernet address actually is configurable (interesting for hackers).

Ethernet (5)

- Ethernet originally was defined with 10 Mbit/s, and this is still in use if the cabling is a bit older.

  Coaxial cable is not supported by the faster versions.

- Modern ethernet cards also support 100 Mbit/s ("Fast Ethernet"), which is today standard.

  It got on the market around 1996. A current PC can transfer via FTP maximally 30–40 Mbit/s, so the network is not the limiting factor. A film in TV quality needs 3–4 Mbit/s. [c't 3/2002, P. Jöcker]

- There is also Gigabit Ethernet and even 10 GB/s.

  Gigabit ethernet got on the market around 1999. It is e.g. used to connect ethernet switches on every floor with the backbone switch.

Ethernet (6)

- Ethernet is used with different cables, e.g.
  - ◇ Thick Ethernet (10Base5)

    This was the original cable (thick coaxial cable), but it is hardly ever used anymore. 10 means 10 Mbit/s and 5 means 500 m segment length (segments can be connected with repeaters). From the network card, an AUI cable leads to a transceiver that is attached with a vampire clamp to the cable. Both cable ends must be terminated with a 50 Ω-resistor to avoid signal reflections.

  - ◇ Thin Ethernet (10Base2)

    BNC-T-connectors are used to connect the network cards into to one long bus of coaxial cable (RG58, about 5mm diameter). The maximum segment length is 185 m (rounded up to give the 2 in the name). Both ends of the bus must be terminated with a 50 Ω resistor. At most 30 nodes can be connected to a segment.

Ethernet (7)

- Ethernet cables, continued:
  - ◇ Twisted-Pair (10BaseT, 100BaseTX)

    This is currently the most common technology used in LANs. The cables are basically telephone cables, containing four pairs of wires, of which 10BaseT and 100BaseTX use only two: One pair for sending and one for receiving. Each computer has its own cable of up to 100 m length which leads to a central hub (two computers can be connected without a hub using a crossover cable). This avoids the problem of e.g. 10Base2 where if the connection is broken at one computer, the entire network does not work. If the cable is good quality ("Category 5 UTP"), the same cable can be used for 100 MBit/s. The connectors are RJ-45.

  - ◇ Fiber Optic (e.g. 10BaseFL, 100BaseFX).

    For longer distances, e.g. 10BaseFL: 2000 m, 100BaseFX: 412 m.

Connection Possibilities

- Bus cable: old, error-prone

  If connection broken at one computer: Entire network does not work.

- Hub: problems with collisions in larger networks

  A Hub forwards a received network packet to all its ports.

- Switch: concurrent connections, today standard

  A switch learns which computer is connected to which port (it reads the sender addresses in the packets). Once it knows where the destination machine is connected, it forwards the packet only to that port. This drastically reduces the problem of collisions. There can be several concurrent connections between unrelated machines via the switch. A switch might also temporarily store a network packet before it forwards the packet (and apply certain correctness checks). In this way, it might interface between 10BaseT and 100BaseTX.

Ethernet: History, Standards

- Ethernet was designed 1970. First, there was only Thick Ethernet. 1980 a first standard was published (DEC, Intel, Xerox).
- 1982: “The” Ethernet Standard was published. It includes Thin Ethernet.
- The IEEE developed a standard similar to Ethernet (Ethernet is a Trademark of Xerox) IEEE 802.3, which was published 1985.
- The version of 1990 introduced Twisted Pair.

Ethernet-Frame Format (1)

- Preamble: 8 bytes

  Alternating 0 and 1 bits to synchronize clocks. The last byte ends with two 1 bits to mark the beginning of the real information.
- Destination Ethernet-Adress (Recipient): 48 Bit
- Source Ethernet Address (Sender): 48 Bit
- Packet Type/Length (see below): 16 Bit
- Data: At least 46 Byte, at most 1500 Byte.
- Frame Check Sequence (CRC): 32 Bit.

  The receiving node ignores the frame if the CRC is incorrect or the bits are not a multiple of 8.

Ethernet-Frame Format (2)

- Unfortunately, the original Ethernet II standard and the IEEE 802.3 standard are slightly different:
  - ◇ In the original standard, the two bytes after the source address are the packet type.

    It determines the protocol on the next higher level, i.e. to which protocol stack module the packet is delivered: E.g. 2048 means IP, 2054 means ARP.
  - ◇ In the IEEE standard, it is the length field.

    This is the number of actual data bytes, which can be less than 46 (the rest is padding).

Ethernet-Frame Format (3)

- Since the possible lengths are $<$ 1500, and the types are larger, the two formats can be distinguished.
- Since IP datagrams contain their length, the length field is not needed to remove the padding.
- IEEE 802.2 defines a “Logical Link Control” sub-layer, on top of the IEEE 803.2 version of Ethernet.
- It uses the beginning of the data for a type field.

  Whereas in the standard ethernet case, IP datagrams are simply sent in the data field (see RFC 894), the IEEE 802.2/803.2 encapsulation is a bit more complicated (RFC 1042).

IP: Internet Protocol (1)

- Data packets (Byte-Strings) are sent from the source to the destination.
- Source and destination are identified via a 32 Bit IP-address (see above).
- Not reliable: It is not guaranteed, that the packet is delivered to the destination node.

  There is not even a guaranteed information about the packet loss. "Best effort": Data packets are not thrown away arbitarily, but only in case of faults, buffer overflows, etc. The protocol TCP above IP ensures the reliability by sending back receipts and retransmitting packets if necessary.

IP: Internet Protocol (2)

- Connectionless: every packet is treated independently from the others.

  It is possible that packages of a larger message take different routes through the network.
- The IP-layer is responsible for the fragmentation of larger packets if the layer below can only transmit small packets.

  E.g. Ethernet can transmit at most 1500 Bytes in one packet.
- Routing decisions are done locally.

  There is no place in the network that plans the entire route: Each node decides only on the next one.

IP-Datagram Format (1)

- Protocol Version: 4 Bit.

  At the moment 4 (or 6 for IPng/IPv6).
- Header length in 32-Bit words: 4 Bit.

  Usually 5, i.e. the head is normally 20 bytes long.
- Type of service (TOS) (Priorities etc.): 8 Bit.

  Nearly always 0.
- Total packet length in bytes: 16 Bit.

  All numbers in the TCP/IP headers are sent "big endian", i.e. the most significant byte first ("network byte order"). It is called "big endian" because one starts at the big end. E.g. 40 is sent as first byte 0, second 40. Intel machines internally use the opposite order.

IP-Datagram Format (2)

- Unique identification of the datagram: 16 Bit.

  E.g. counter. It is used to compose fragments of a datagram.
- Unused: 1 Bit.
- Flag "Don't Fragment": 1 Bit.
- Flag "More Fragments": 1 Bit.
- Fragment offset: 13 Bit.

  This is the position of the data sent in this packet relative to the beginning of the entire message (in units of 8 bytes).

IP-Datagram Format (3)

- Time to live (TTL): 8 Bit.

  E.g. 32. This is the time the packet may exist in the net. Every rounter must decrement this value by at least one (or by the number of seconds if the packet stays longer than a second on the router).

- Transport-level protocol: 8 Bit

  Protocol on the next higher level, to which this datagram must be passed: 6 für TCP, 17 für UDP, 1 für ICMP.

- Header checksum: 16 Bit.

  The one's complement of the 16-bit sum of all 16-bit words of the header (assuming the header checksum field is 0). If the reciever adds up all header fields including the checksum, the result must be all 1 bits. The data must be checked by higher level protocols.

IP-Datagram Format (4)

- Source IP address (sender): 32 Bit.
- Destination IP address (recipient): 32 Bit.
- Options (if header length > 5).

  The options consist of (Header length $-$ 5) words of 32 bits (this is why the header length is specified). E.g. one can request that every router adds its IP address in this field ("record route"), but unfortunately, the protocol permits only 9 addresses (because of the 4-bit header length). One can also specify IP addresses that must be traversed ("loose source routing"), or all IP addresses that can be traversed ("strict source routing"). There are more options.

- Data

  Length of Data in bytes = Total length $-$ Header length $*$ 4.

ARP: Address Resolution (1)

- When the IP network software wants to send a packet, it first compares the destination address with its own address.
- If the two agree in the network part (and the subnet part), it knows that the destination machine is connected to the same physical network, and no routing via a gateway is necessary.
- Suppose the physical network is an ethernet. How does the IP module determine the ethernet address of the recipient?

ARP: Address Resolution (2)

- This is done by sending an ARP packet as broadcast (to all machines on the local network/subnet).
- The ARP packet means: "If you have the IP address X, please tell me (IP address Y, ethernet address Z) your ethernet address."
- The answers are buffered (in the ARP table), so that the machine does not need to ask each time.

  Of course, the entries are not remembered forever (IP addresses and ethernet cards can change). The normal expiration time is 20 minutes. Some computers also have a file `/etc/ethers`.

SLIP: Serial Line IP (1)

- SLIP is a protocol for connecting two computers over a serial interface (RS-232).

  This can be a direct connection over a serial line cable, or it can be a connection via two modems and a telephone connection. E.g. a PC is connected with a computer that acts as gateway to the internet.

- SLIP resides on the same level as ethernet, i.e. it is used to transport IP packets.
- The format is very simple: Every packet is sent followed by the character `END` (`C0`).

SLIP: Serial Line IP (2)

- Most implementations also send the character `C0` before the packet to terminate garbage (noise).

  The IP-level understands that this was not a valid data packet and ignores it.

- Since the byte `C0` is special it cannot be directly send in the data (it would terminate the packet). But IP packets can of course contain this byte.
- The problem is solved by encoding `C0` in the data by the sequence of the two bytes `DB`, `DC`.
- Now `DB` is also special. It is encoded as `DB`, `DD`.

SLIP: Serial Line IP (3)

- Problems:
  - ◇ Both computers must know the IP address of the other computer.
  - ◇ On each computer only one higher level protocol (IP) can use the SLIP-connection, since there is no type field.
  - ◇ There is no checksum (but the higher level protocols check the correct transmission).
- There is also "Compressed SLIP" that basically suppresses repeated parts of TCP packets.

Point-to-Point Protocol (1)

- Today, normally PPP is used for connections between two computers via a serial line (or modems).
- It solves the problems of SLIP, but is significantly more complicated.
- PPP does not only determine how data packets are exchanged between the computers, it also defined a "Link Control Protokol" (LCP) to establish, configure, and test the data-link connection.

Point-to-Point Protocol (2)

- In addition there are different "Network Control Protocols" (NCP), in order to determine parameters for the transmission of higher-level protocols (e.g. IP).

  This is similar to the compression in "Compressed SLIP".

- There are also protocols for authentication (PAP, CHAP).

Point-to-Point Protocol (3)

- PPP frames have the following format:
  - ◇ Flag, Address, Control: `7E`, `FF`, `03` (fixed for PPP)

    The first three bytes are fixed to make PPP an instance of the ISO HDLC standard ("high-level data link control").

  - ◇ Protocol: 2 Bytes

    E.g. 33 for IP, 49185 for LCP.

  - ◇ Data: Up to 1500 bytes.
  - ◇ Checksum (CRC): 2 Byte.
  - ◇ Flag: `7E` (fixed)

Point-to-Point Protocol (4)

- Since in this way the data cannot contain `7E`, again an "escape mechanism" is needed:
- `7D` means: Bit 6 of the following byte is inverted.

  E.g. instead of `7E`, the two byte sequence `7D`, `5E` is transmitted. In the same way, `7D` is encoded as `7D`, `5D`.

- Normally, all ASCII control characters are made harmless in this way (they might confuse modems).

  One can use the link control protocol to specify which of the 32 ASCII control characters must be espaced.

TCP (1)

- TCP is the "Transmission Control Protocol".
- In user view, the data is transferred as a continuous stream (sequence of bytes), packet boundaries are not visible.
- TCP creates a full duplex connection similar to a telephone line.

  I.e. there are two independent data streams from A to B and from B to A.

TCP (2)

- TCP is reliable: The recipient sends back an acknowledgement for the received data.

  If the sender does not get the acknowledgement within a certain time limit, it sends the data again. If the sender was not successful after some tries, the user is at least informed about the problem.

- TCP permits several concurrent connections from or to a machine. The connections are distinguished via 16 bit port numbers.

  Each connection is identified by both IP addresses and both port numbers. Therefore, it is possible that different connections exist to the same port on the same machine.

TCP (3)

- The server process tells the operating system on its computer that it is ready to accept connections on a certain port ("passive open").
- Some common ports are e.g.:

Port	Service	Port	Service
7	`echo`	23	`telnet`
13	`daytime`	25	`SMTP/mail`
20	`ftp-data`	37	`time`
21	`ftp`	79	`finger`
22	`ssh`	80	`WWW`

  See `/etc/services`, `/etc/inet/services`.

TCP (4)

- Port numbers below 1024 are reserved for privileged processes (owned by `root` on UNIX).

  E.g. a normal user cannot start a web server that listens on port 80, but he/she can start a web server that listens on port 8080.

- The client process tells the operating system with which port on which machine it wants to connect.

  The operating systems assigns the client an unused port on the local machine.

TCP (5)

- Sliding Window Principle: Each machine tells the other how many bytes of buffer space it still has. Until this size, packets can be sent without waiting for an acknowledgement.

  The buffer space is needed in order to assemble the packets in the correct sequence and deliver the data bytes to the sender. Since packets can take different routes, they may reach the destination e.g. in the opposite order.

- "Piggybacking": Together with the receipt, data in the opposite direction can be sent.

  Packets in both directions have the same format.

TCP (6)

- Adaptive Timeout: The timelimit for the retransmission of unacknowldged packets is not constant, but is adapted by measuring round trip times.

  TCP is used in fast local area networks and for relatively slow worldwide connections. If e.g. a gateway becomes overloaded, a protocol with a fixed time limit could make the situation worse by retransmitting packets that are actually not lost.

TCP-Segment Format (1)

- Source port number: 16 Bit
- Destination port number: 16 Bit
- Sequence number: 32 Bit

  This is the byte number of the first data byte in this segment. However, it is not counted from 0, but from an arbitrarily chosen number, agreed upon when the connection built up. In this way, packets cannot be so easily confused e.g. after a machine crashed.

- Acknowledgement number: 32 Bit

  The next expected sequence number, i.e. one higher than the sequence number of the last received byte (refers to sequence numbers in the opposite direction).

TCP-Segment Format (2)

- Header length (in 32-bit words): 4 Bit.

  Necessary because of options.

- Reserved: 6 Bit.
- Flags (URG, ACK, PSH, RST, SYN, FIN): 6 Bit.

  E.g. `SYN` is used when a connection is established (in the first segment in both directions) in order to synchronize the sequence numbers. When `FIN` is set, no more data will flow in that direction.

- Window size: 16 Bit

  Number of bytes (from the destination) that would still fit into the sender's buffer after the bytes acknowldged in this segment.

TCP-Segment Format (3)

- Checksum (covers header and data): 16 Bit.
- Urgent-Pointer: 16 Bit.

  This is used to transmit emergency data, e.g. when an interrupt key was pressed. The urgent pointer is an offset that must be added to the sequence number of this segment to the the sequence number of the last byte of urgent data.

- Options (if any).
- Data (if any).

  If data flow only in one direction, the TCP segments in the other direction are only sent as acknowledgements.

Outlook: IPv6 (1)

- Internet Protocoll, Version 6 was published in 1994 by the Internet Engineering Task Force (IETF) as a proposed standard.

  1998 it became a draft standard.

- One important change is that IPv6 addresses are 128 bit long (whereas the current IPv4 addresses are 32 bit long).

  This is important because there is a shortage in IP addresses today. However, it also means that software that uses network connections might have to be changed.

Outlook: IPv6 (2)

- Currently, there are a few experimental IPv6 networks, but more and more hardware and software products support it.

  Windows XP also contains IPv6 support.

- IPv6 packets can be tunneled from one IPv6 network into another one via the IPv4 internet.

  Tunneling is a general technique when one has to send one protocol via another protocol. At the beginning of the tunnel, the IPv6 packets are put into an IPv4 envelope, at the end of the tunnel, they are unpacked and become again IPv6 packets.

Overview

1. The Internet: A Network of Networks
2. History of Internet and WWW
3. Network Protocols
4. Internet Service Providers
5. Basics of Socket-Programming

Internet Service Providers (1)

Selection Criteria:

- Kind of connection (Modem, ISDN, DSL, ...)
- Price

  Price per minute? Different rates at different times of the day? Telephone costs included? Is there an additional cost per connection? Basic fee per month? Contractual minimum per month? Flatrate? How easily can one change to another ISP?

- Speed

  If the ISP has too many customers for the bandwidth of its lines (and its exchanges with other networks), it becomes the bottleneck (normally the bottleneck is the modem). Of course, this depends on the time of the day.

Internet Service Providers (2)

- Reliability, Availability

  Being logged off from time to time is not nice.

- Technical help in case of problems? Price?
- Restrictions to the internet connection.

  E.g. in freenet, the first HTTP access always returns the freenet web page, no matter what URL was requested. It also automatically uses an HTTP proxy (cache) even if this is not configured in the browser.

- Fixed IP address?

  This is only important if one is always connected to the internet and wants to run servers. There are also dynamic DNS providers, e.g. [http://www.dyndns.org].

Internet Service Providers (3)

Additional Services:

- Online services like AOL or T-Online have contents of their own (only for members).
- Email-Account included?

  Emails must be saved while one is not online. Email services differ e.g. in the number of megabytes that can be stored, whether emails are thrown away after some time, whether a POP3 access is possible, whether emails can be collected from other accounts or forwarded, whether SPAM/viruses are detected. If the ISP does not have this service, there are e.g. [http://freemail.web.de], [http://www.gmx.de], [http://mail.yahoo.de], [http://www.hotmail.com]. Some Email services offer to send an SMS when one gets email ("unified messaging").

- Web space for a homepage included?

Modems (1)

- "Modulator–Demodulator": Uses a normal analog telephone connection.
- V.90 standard: Max. receive rate: 56 Kbit/s, max. send rate: 33.6 Kbit/s.

  V.92 standard (Summer 2000): Now 48 Kbit/s send rate.

- Modems usually do a data compression, thus higher data rates can be reached.

  V42.bis can compress under optimal circumstances to 25%.

Modems (2)

- In Germany, there are different telephone cable connectors for usual telephones and for modems.

  "F" coded TAE-connector for telephone ("Fernsprechen"),
  "N" coded for modem, Fax, etc. ("Nachrichten-Endgerät").
  Today, there is usually an "F" coded connector in the middle, and two "N" coded at both sides. An "F"-coded plug is at the bottom slightly wider, an "N"-coded in the second part from the top.

- Connected to the computer via a serial line (9 pin or 25 pin connector) or USB.
- Usually have the capability to send/receive faxes.

ISDN, DSL

- ISDN gives 64 Kbit/s, but two channels can be coupled to get 128 Kbit/s.

  If one uses two channels, one has to pay for both (i.e. two times the normal rate).

- DSL: Up to 768 kbit/s ($= 12 * 64$ Kbit/s).
- E.g. download time for Rational Rose (234 MB):

Modem	(56 Kbit/s):	10 h, 18 min.
ISDN	(128 Kbit/s):	4 h, 33 min.
DSL	(768 kbit/s):	46 min.

Configuring a Modem (1)

- Connection data of some German Call-by-Call ISPs:
  - ◇ MSN Easysurfer Power [0193670]

    User: msn@easysurfer-power.de, Password: msn. If one has preselected a different phone company, it might by safer to use the area code 01033 (Telekom) in order not to pay for the phone call. Currently costs 1.17 cent/min. Time is measured in seconds. Uses lines of MCI Worldcom. Nameserver: 192.76.144.66.

  - ◇ Arcor Spar: Internet by Call [0192076]

    Area code 01070 (Arcor), User: arcor-spar, Password: internet. Currently costs 0.67 cent/min (Mo-Fr 9-18) or 0.79 cent/min (nights and weekends) plus 8.1 cent per connection (counts whole minutes). Nameserver: 145.253.2.11, 145.253.2.75.

Configuring a Modem (2)

- Students of the University of Halle can use e.g. 0345/6860 and 0345/55-21940 with their username and network password from the computing center.

  See [http://pc-serv.urz.uni-halle.de/einwahl/index.htm].
  The modem must support V.42 or MNP4 error correction.

- Price comparisons: [http://www.teltarif.de], [http://www.heise.de/itarif/].
- It has happened that dialer programs got as "trojan horses" on computers and changed the internet connection to very expensive 0190-numbers.

Configuring a Modem (3)

- Under Properties/Networking one chooses "PPP" as "Type of Dial-Up Server" and "TCP/IP" as allowed network protocol.
- Under Properties/Dailing one can configure automatic disconnection for idle connections.

  See also modem configuration under "Connection".

- A usual modem configuration is 8 data bits, no parity bit, and 1 stop bit.

  The interface speed is usually 115200 bit/s. It should be higher than the actual transfer speed, because the modem compresses the data.

Speed and Reachability

"Internet Weather Report":

- [http://www.internettrafficreport.com/]
- [http://weather.ucr.edu/]

  Measurements from the University of California at Riverside.
  Other connections: [http://www.noc.ucla.edu/weather.html]

Speed of your internet connection:

- [http://de.netmeter.eu/]
- [http://www.speedmeter.de/speedtest]

Overview

Sockets (1)

- Network programming is today normally done with the "socket" interface.

  Sockets are connection endpoints of a communication line between two processes on different computers.

- The interface was developed as part of 4.3 BSD UNIX. Today it is contained in all UNIX versions, and there is also a Winsock library for Windows.

  See e.g. the Winsock FAQ: [`http://tangentsoft.net/wskfaq/`]. Windows 95 and NT 3.x ship with Winsock, Version 1.1, newer Windows versions (98, ME, NT 4.0, 2000) ship with Winsock 2.

- The following example code is in "C".

Sockets (2)

- Sockets are similar to files: Under UNIX, the system calls `read` and `write` can also be used with a socket descriptor (handle) instead of a file descriptor.
- There are also special functions `recv` and `send`.

  Under Windows, only these work.

- Opening a socket is a bit more complicated than opening a file. It is done in several steps which differ between server and client (see below).

  One reason for the complexity of the socket interface is that it supports not only TCP/IP, but all kinds of network protocols and interprocess communication mechanisms.

Sockets (3)

- Sockets for TCP are created with the call

  `s = socket(AF_INET, SOCK_STREAM, 0);`

  The socket-descriptor `s` is declared as `int`. The function `socket` returns a small positive number or `-1` in case of errors (under Windows use the constant `SOCKET_ERROR`).

- The three arguments (`family`, `type`, `protocol`) of `socket` are necessary because sockets can be used also with other networking protocols.

  E.g. `AF_INET6` would be used for IPv6, and `SOCK_DGRAM` would select UDP instead of TCP. Instead of `AF_INET` ("address family") `PF_INET` ("protocol family") would be more correct, but the constants have equal values.

Sockets (4)

- C programs using sockets must include certain header files and be linked with networking libraries.

  Under UNIX (at least Solaris), the declaration files `<sys/types.h>` and `<sys/socket.h>` must be included. The linker must be called with the options `-lsocket -lnsl` for the networking libraries. For more information on a socket interface function, call "`man -s 3socket` ⟨Function⟩". Under Windows, `windows.h` and `winsock.h`/`winsock2.h` must be included, and the program must be linked with `wsock32.lib`/`ws2_32.lib`.

- The Winsock library must be initialized.

  `err = WSAStartup(ver_requested, &ver_info);`
  `ver_requested` has type `WORD`. E.g. assign `MAKEWORD(2,2)` for version 2.2.
  `ver_info` has type `WSADATA`. Selected version: `ver_info.wVersion`.
  `WSAStartup` returns `0` if successful and `-1` in case of errors.

Sockets (5)

- The Winsock library must be shut down at the end.

  `err = WSACleanup();`
  The function returns `0` if successful and `-1` in case of errors.

- If a call to a socket function fails, one can get more information on the kind of the error.

  Under UNIX, the global variable `extern int errno;` is set as for other system calls. Error numbers are declared in `errno.h`. One can get the error message with `strerror(errno)`. One can also call `perror(msg)` to print `msg` followed by the system error message.
  Under Windows, one can call `WSAGetLastError()` to return the number of the last error. Constants for error numbers are defined in `winsock.h`. E.g. `WSANOTINITIALISED` means that one has forgotten to call `WSAStartup`.

Sockets (6)

- Steps in a Client:
  - ◇ Call `socket` to create a socket.
  - ◇ Fill `sockaddr` structure with the server's address.
  - ◇ Call `connect` ("active open").
- Steps in a Server:
  - ◇ Call `socket` to create a socket.
  - ◇ Call `bind` to set the local address (port).
  - ◇ Call `listen` ("passive open"): Creates queue.
  - ◇ Call `accept` to fetch a connection from the queue.

Network Addresses (1)

- Each network protocol family has its own structure that contains the address information, e.g.

```
struct sockaddr_in addr; /* TCP/IP Address */
```

Under UNIX (Solaris), this structure is declared in `<netinet/in.h>`, which is automatically included from `<netdb.h>`. Under Windows, it is declared in `winsock.h/winsock2.h`.
Every address structure is at least 16 bytes long, which are, however, not completely used by TCP/IP (IPv6 needs more than 16 Byte).
It is good programming practice (but not necessary) to initialize the structure first completely to 0:

```
memset(&addr, 0, sizeof(addr));
```

The function `memset` is declared in `<string.h>`.

Network Addresses (2)

- First, the address family is stored in the structure:

```
addr.sin_family = AF_INET;
```

- Then the IP address of the server:

```
addr.sin_addr.s_addr = inet_addr("127.0.0.1");
```

The function `inet_addr` converts the numeric IP-address into a number. To use the name of the server, see Chapter 2.

- And finally the port number, e.g. 13 for `daytime`:

```
addr.sin_port = htons(13);
```


Network Addresses (3)

- Address and port number must be stored in the network byte order.

The function `htons` ("host to network short") does this conversion for 16-bit numbers (type `short` in C) `htonl` for 32-bit numbers (`long`).

- On SUN SPARC machines, it would work to assign simply 13 to the port number part of the address structure, but not on Intel machines.

On Intel machines, one would get port 3328 ($= 13 * 256$).

Connecting to a Server (1)

- Now the client opens the connection to the server:

```
if(connect(s, (struct sockaddr*) &addr,
                        sizeof(addr)) < 0) {
        perror("Error connecting to server");
        exit(1);
}
```

- The function returns `0` if successful and `-1` (on Windows: `SOCKET_ERROR`) if an error occured.
- It has three arguments:
    - ◇ The first is the socket descriptor one got from the function `socket` (see above).

Connecting to a Server (2)

- Arguments of `connect`, continued:
  - ◇ Second argument: Pointer to a structure that contains the server's network address.

    In C, one can determine the address of a variable by prefixing it with "`&`". Since each protocol has a structure of its own, and `connect` works with different protocols, one converts the type of the second argument to "pointer to a `sockaddr` structure". This only tells the compiler to shut up about the apparent type error.
  - ◇ Third argument: Length of the structure (bytes).

    This gives an additional protection, although the address family stored at the beginning of the structure should suffice. However, if the definition of the structure should ever be changed, programs that were compiled with the old structure can be detected. In C, the operator `sizeof` returns the size of a variable in bytes.

Sending/Receiving Data (1)

- One can use the standard UNIX calls `write` and `read` also for sockets instead of files.

  This works only under UNIX. Under Windows, use `send` and `recv`.
- However, there are also special calls `send` and `recv`.
- These function can be called only with a socket descriptor that is in a connected state (e.g. after `connect` or `accept`).

  For connectionless protocols like UDP there are also calls `sendto`, `sendmsg` and `recvfrom`, `recvmsg`, see the manual.

Sending/Receiving Data (2)

- `send` and `recv` have four arguments:
  - ◇ The socket descriptor.
  - ◇ A character array that contains the data to be sent, or where the data received will be stored.
  - ◇ Number of bytes to be sent or maximum number of bytes that can be received (array size).
  - ◇ Flags (normally 0).
- `read` and `write` are the same, except that the last argument is missing.

Sending/Receiving Data (3)

- All four functions return the number of bytes sent or received (or `-1`/`SOCKET_ERROR` in case of errors).
- Suppose one wants to receive/read data:
  - ◇ If the operating system (OS) has already received data for this connection, it will put them into the buffer (array) specified in the call and return the number of bytes.
  - ◇ Even if the buffer is larger, and more bytes will arrive over the net, the function returns what the OS currently has (total length is unknown).

Sending/Receiving Data (4)

- Receiving data, continued:
  - ◇ Only if the operating system currently has no data for the connection, it will put the calling process to sleep until data are received or the connection is terminated.
  - ◇ When the connection is terminated, `0` is returned.
- It is not possible to wait always until the connection is terminated, since some programs first send some data, then receive, then send again, and so on.

Sending/Receiving Data (5)

- Also when one wants to send data (with `write` or `send`), one has to check the result value (number of bytes actually sent).
- Via TCP, the receiving machine declares how many free bytes of buffer space it still has.
- The sending machine will then send only that many bytes, and the call to `send`/`write` will return.
- Unless one got an error indication, one can simply call the function again (with the buffer address adjusted to the remaining portion).

Sending/Receiving Data (6)

- E.g. reading the data sent by the `daytime` server:

```
#define ANS_MAX 100
char answer[ANS_MAX]; char *p; int free; int n;
...
p = answer; free = ANS_MAX;
while(free > 0 && (n = recv(s,p,free,0)) > 0) {
        free = free - n;
        p = p + n;
}
if(n < 0) perror("Error reading from socket");
p = 0;
puts(answer);
```


Programming a Server (1)

- A server normally calls `bind` after it has created a socket (with `socket`) in order to specify the own address (port number).

  A client could also `bind` if it wants a specific port (seldom needed). Otherwise the call to `connect` selects any free port on the local machine.

- Before `bind` can be called, and address structure must be filled with IP address and port number as explained above.

Programming a Server (2)

- One usually assigns

```
addr.sin_addr.s_addr = htonl(INADDR_ANY);
```

This means that the local machine will listen for incoming connections on all of its IP addresses (in case it has more than one network connection). Of course, one can also select a specific IP address of the local machine if that is important.

- The call to `bind` looks as follows:

```
bind(s, (struct sockaddr*)&addr, sizeof(addr))
```

The arguments are identical to those of `connect`, but here they determine the local address, whereas in `connect` they are the address of the other machine. A TCP connection is identified by both addresses. As expected, `bind` returns `0` if successful and `-1` in case of an error.

Programming a Server (3)

- The function `listen` does the "passive open" for the socket: It now waits for incoming connections.

```
listen(s, QUEUESIZE)
```

In contrast, `connect` does the active open.

- The second argument tells the operating system how many connection requests can be maximally queued.

The argument must be a positive `int`. If more clients want to connect to the server before it calls `accept` to retrieve a connection, the operating system will refuse the connection requests. As always, `listen` returns `0` if successful and `-1` if not.

Programming a Server (4)

- The function `accept` is called with a listening socket and returns a connected socket as soon as there is a connection with a client:

```
c = accept(s, (struct sockaddr*)&c_addr, &len)
```

- After the call to `accept` there are two sockets:
  - ◇ The listening socket `s` can be used for further calls of `accept`.
  - ◇ On the connected socket `c`, on can use `recv`/`send` to serve the client.

Programming a Server (5)

- If there are no clients in the queue that was created with `listen`, `accept` blocks until a client connects.

The server process is put to sleep while there is no client to serve. There is an option for non-blocking calls.

- When `accept` returns, the variable `c_addr` contains the network address of the client.

It is declared as `struct sockaddr_in c_addr`.
`socklen_t len` is an input-output parameter: Before the call, one must assign `sizeof(c_addr)`. This is necessary, because sockets work not only with TCP/IP. Already IPv6 would need more space. After the call, `len` contains the actual length of the structure.

- As the other functions, `accept` returns `0` if ok.

Programming a Server (6)

- It is important to close the socket `c` after the client was served: `close(c)`.

  There is only a limited number of socket descriptors per process.

- A simple server first calls `socket`, `bind`, and `listen`, and then calls `accept` in a loop:

```
while((c = accept(s,(struct sockaddr*)&c_addr,
                  &len) >= 0) {
    Serve client: recv(c,...) send(c,...)
    close(c);
}
```


Parallel Servers (1)

- Many servers create for every connection with a client a new process (copy of the main server process).
- The original server process can then immediately call again `accept` to fetch the next connection with a client.
- However, generating a new process for every client is relatively inefficient.

  With threads instead of processes this works better, since the overhead for generating a new thread is smaller than for generating a new process.

Parallel Servers (2)

- Therefore, often already several copies of the server process are created when the server is started.
- These processes call `accept` in a round robin fashion.
- Depending on the operating system, also several copies of the server process can call `accept` on the same socket and get connected one after the other.