Sockets

Thomas Schwarz, SJ

Networking

- Networking is organized in layers:
  - Application: Access to network resources
  - Transport: Provides message and error delivery
  - Internet: Moves packets from source to destination through networks owned by different entities
  - Network Interface: How to move data between two machines / switches / routers

Networking

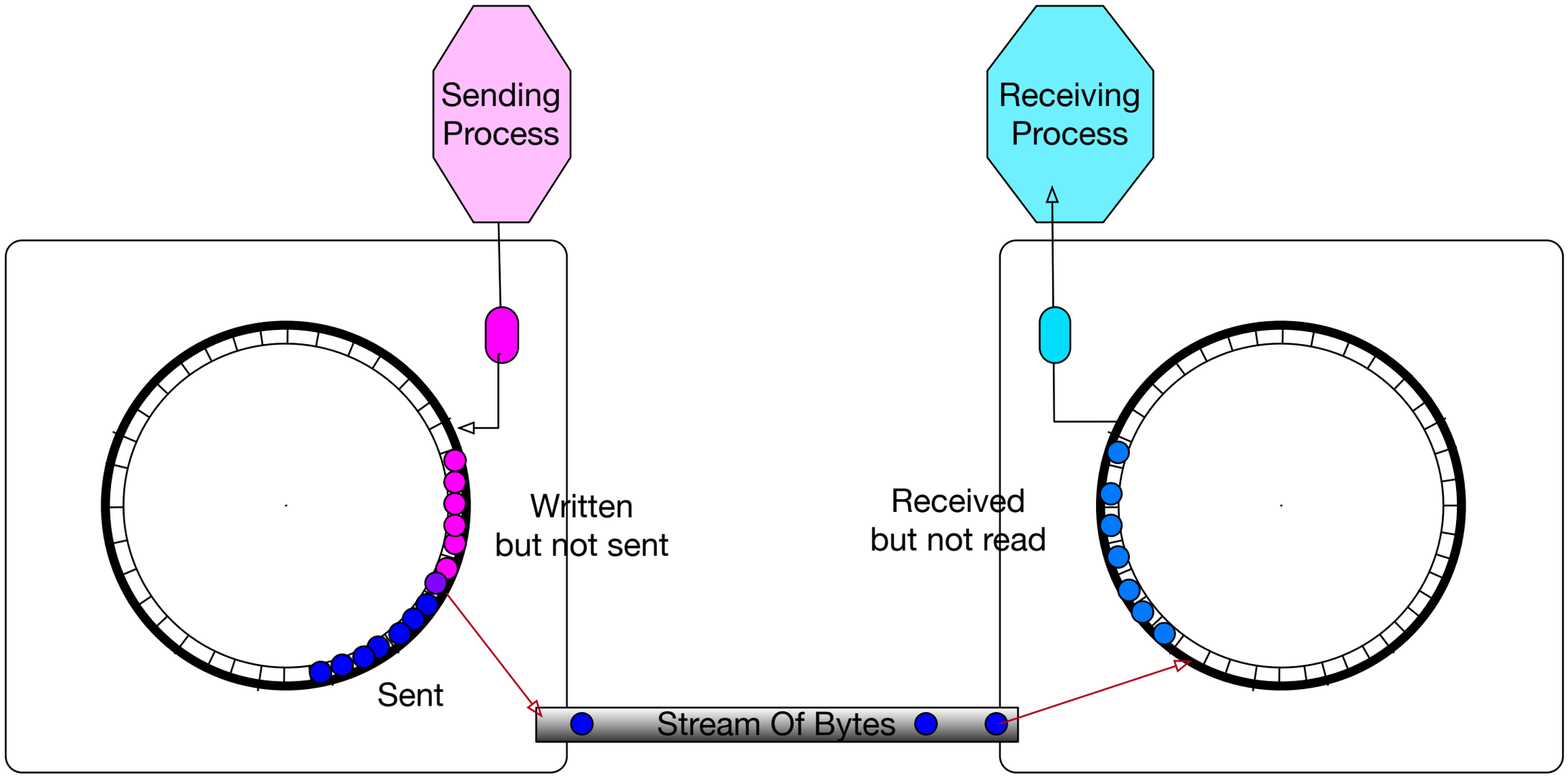
- Transport layer provides:
  - UDP: connectionless, unreliable, fast, datagrams
  - TCP: established connection with error resilience

Transmission Control Protocol: TCP

- Process-to-process communication
- Stream-oriented protocol
- Full duplex communication
- Connection oriented
- Reliable Service

Transmission Control Protocol: TCP

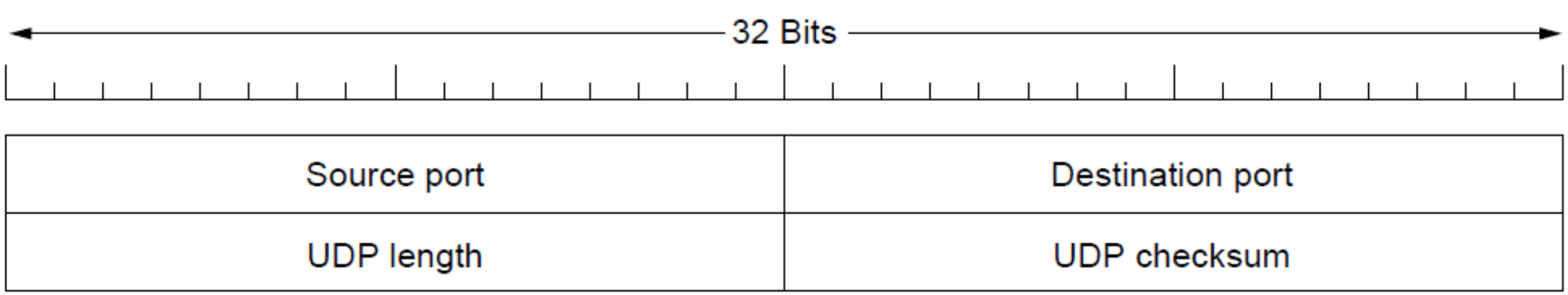
- TCP: Receives bytes to send from process and sends them reliably to the receiving process



UDP

- User Datagram Protocol (UDP)
  - Only adds socket addressing to the networking layer
  - Useful if you do not want the overhead of connection establishment and maintenance

UDP Header



Length: Includes the 8B UDP header

Checksum: Calculated from a pseudo-header (part of IP header)
UDP header (without check-sum)
Data

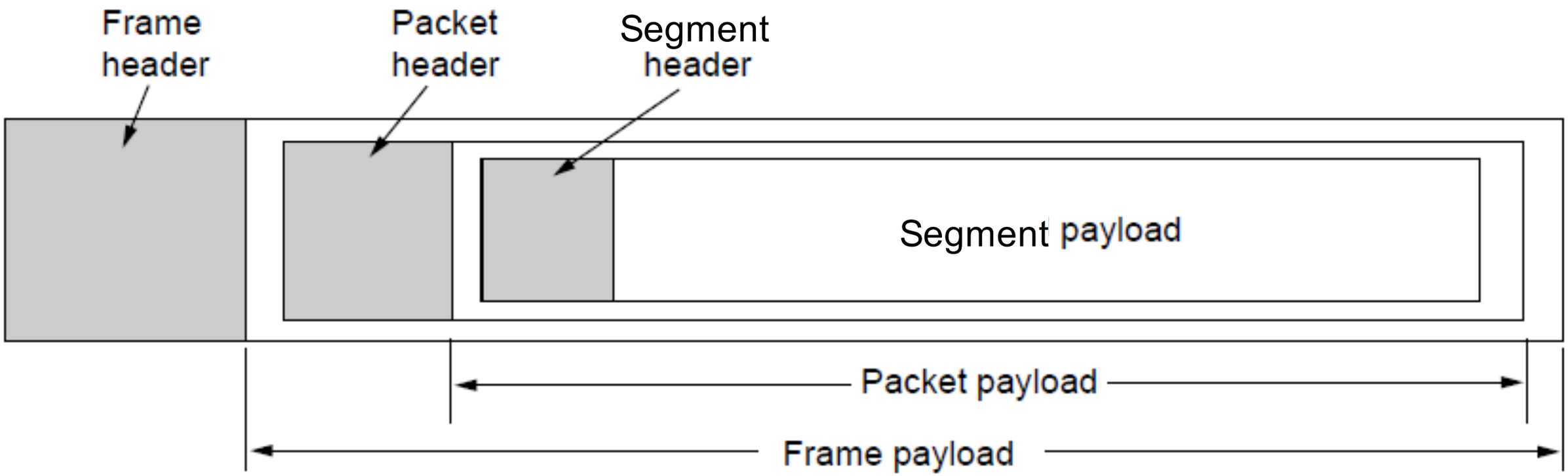
Transport Service Primitives

- Typical primitives provided to application programs

Primitive	Packet sent	Meaning
Listen	-	Block until some process tries to connect
Connect	Connection Request	Actively attempt to establish a connection
Send	Data	Send information
Receive	-	Block until a data packet arrives
Disconnect	Disconnection Request	Request a release of the connection

Transport services provided to application layer

- Transport layer embeds segments in packets that are embedded in frames



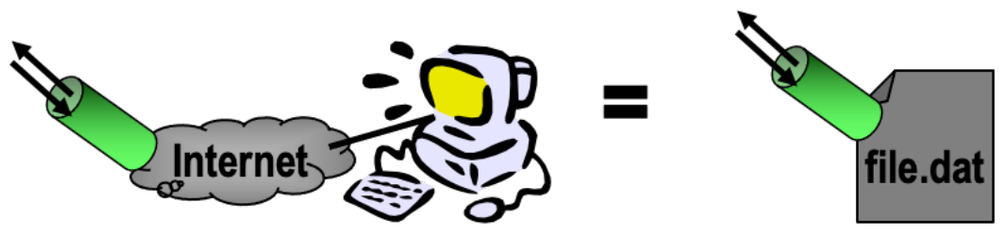
Berkeley Sockets

- Developed for Unix 4.2BSD (1983)
  - Still used for internet programming especially on Unix systems
  - Windows has winsock

Primitive	Meaning
SOCKET	Create a new communication end point
BIND	Attach a local address to a socket
LISTEN	Announce willingness to accept connections; give queue size
ACCEPT	Block the caller until a connection attempt arrives
CONNECT	Actively attempt to establish a connection
SEND	Send some data over the connection
RECEIVE	Receive some data from the connection
CLOSE	Release the connection

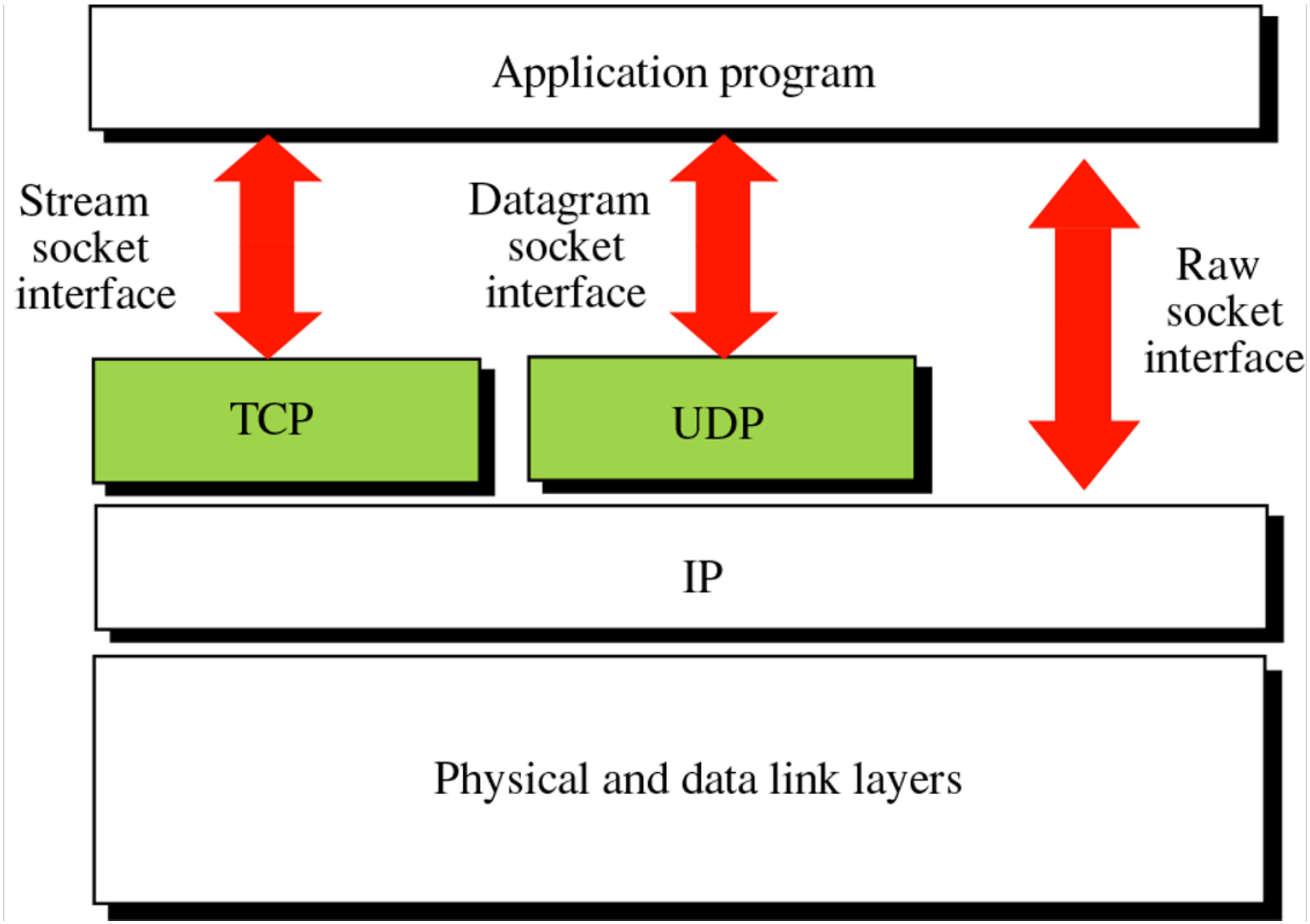
Berkeley Socket

- Basic Idea:
  - Network connection is like a file
    - Read from / Write to like to a file



- Socket procedures in Unix are systems calls
  - Implemented in the “top half” of the kernel
- Windows implemented as a library (DLL)

Berkeley Sockets



Transport Addresses

- Transport layer allows communications between processes
- Implemented via ports
  - Each host is identified by IP address
  - Each process is identified by a port number

Port Numbers

- Internet Corporation for Assigned Names and Numbers (ICANN)
  - Well-known ports: Assigned by ICANN
  - Registered ports: Neither assigned nor controlled, but can be registered to prevent duplication
  - Dynamic ports: used as temporary or private port numbers

Port Numbers

- Example:
  - telnet (needs to be installed on MacOS and Windows OS)
  - telnet 129.6.15.28 13
    - Connects to the daytime service at NIST Gaitersburg on port 13
- In MacOS / UNIX, you can find port assignments in
  - /etc/services

Port Numbers

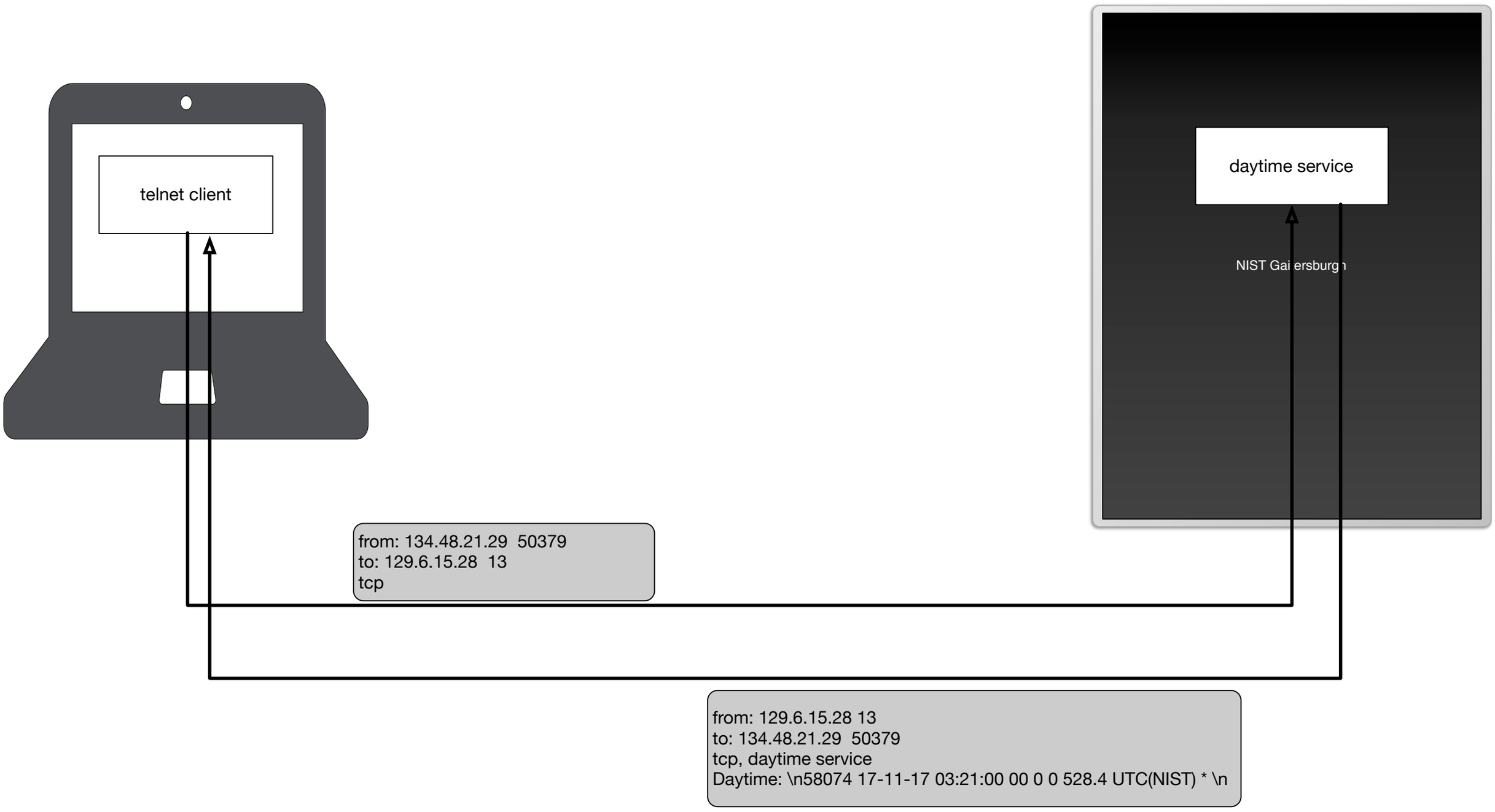
thomasschwarz — -bash — 80×24

```
Connected to time-a-g.nist.gov.
Escape character is '^]'.

58074 17-11-17 02:10:08 00 0 0 532.0 UTC(NIST) *
Connection closed by foreign host.
MSCSs-MacBook-Pro-2:~ thomasschwarz$ clear
```

```
MSCSs-MacBook-Pro-2:~ thomasschwarz$ telnet 129.6.15.28 13
Trying 129.6.15.28...
Connected to time-a-g.nist.gov.
Escape character is '^]'.
Connection closed by foreign host.
MSCSs-MacBook-Pro-2:~ thomasschwarz$ telnet 129.6.15.28 13
Trying 129.6.15.28...
Connected to time-a-g.nist.gov.
Escape character is '^]'.
Connection closed by foreign host.
MSCSs-MacBook-Pro-2:~ thomasschwarz$
```

Port Numbers



Port Numbers

from: 134.48.21.29 50379
to: 129.6.15.28 13
tcp

- Destination address selects the server
- Destination port address selects the service (here day-time-server)
- Source address & port are needed to find the destination for the response

from: 129.6.15.28 13
to: 134.48.21.29 50379
tcp, daytime service
Daytime: \n58074 17-11-17 03:21:00 00 0 0 528.4 UTC(NIST) * \n

Port Numbers

thomasschwarz — -zsh — 80×38

```
Last login: Fri Nov  5 02:03:37 on ttys000
thomasschwarz@Peter-Canisius ~ % cat /etc/services
#
# Network services, Internet style
#
# Note that it is presently the policy of IANA to assign a
# port number for both TCP and UDP; hence, most entries her
# even if the protocol doesn't support UDP operations.
#
# The latest IANA port assignments can be gotten from
#
#       http://www.iana.org/assignments/port-numbers
#
# The Well Known Ports are those from 0 through 1023.
# The Registered Ports are those from 1024 through 49151
# The Dynamic and/or Private Ports are those from 49152 thr
#
# $FreeBSD: src/etc/services,v 1.89 2002/12/17 23:59:10 eri
#       From: @(#)services      5.8 (Berkeley) 5/9/91
#
# WELL KNOWN PORT NUMBERS
#
rtmp              1/ddp    #Routing Table Maintenance Proto
tcpmux            1/udp     # TCP Port Service Multiplexer
tcpmux            1/tcp     # TCP Port Service Multiplexer
#                          Mark Lottor <MKL@nisc.sri.com>
nbp               2/ddp    #Name Binding Protocol
compressnet       2/udp     # Management Utility
compressnet       2/tcp     # Management Utility
compressnet       3/udp     # Compression Process
compressnet       3/tcp     # Compression Process
#                          Bernie Volz <VOLZ@PROCESS.COM>
echo              4/ddp    #AppleTalk Echo Protocol
#                 4/tcp    Unassigned
#                 4/udp    Unassigned
rje               5/udp     # Remote Job Entry
rje               5/tcp     # Remote Job Entry
#                          Jon Postel <postel@isi.edu>
```

thomasschwarz — -zsh — 80×38

```
ibm_wrless_lan  1461/udp    # IBM Wireless LAN
ibm_wrless_lan  1461/tcp    # IBM Wireless LAN
#                          <flanne@vnet.IBM.COM>
world-lm        1462/udp    # World License Manager
world-lm        1462/tcp    # World License Manager
#                          Michael S Amirault <ambi@world.std.com>
nucleus         1463/udp    # Nucleus
nucleus         1463/tcp    # Nucleus
#                          Venky Nagar <venky@fafner.Stanford.EDU>
msl_lmd         1464/udp     # MSL License Manager
msl_lmd         1464/tcp     # MSL License Manager
#                           Matt Timmermans
pipes           1465/udp     # Pipes Platform  mfarlin@peerlogic.com
pipes           1465/tcp     # Pipes Platform
#                          Mark Farlin <mfarlin@peerlogic.com>
oceansoft-lm    1466/udp     # Ocean Software License Manager
oceansoft-lm    1466/tcp     # Ocean Software License Manager
#                          Randy Leonard <randy@oceansoft.com>
csdmbase        1467/udp     # CSDMBASE
csdmbase        1467/tcp     # CSDMBASE
csdm            1468/udp     # CSDM
csdm            1468/tcp     # CSDM
#                          Robert Stabl <stabl@informatik.uni-muenchen.de>
aal-lm          1469/udp     # Active Analysis Limited License Manager
aal-lm          1469/tcp     # Active Analysis Limited License Manager
#                          David Snocken  +44 (71)437-7009
uaiact          1470/udp     # Universal Analytics
uaiact          1470/tcp     # Universal Analytics
#                          Mark R. Ludwig <Mark-Ludwig@uai.com>
csdmbase        1471/udp     # csdmbase
csdmbase        1471/tcp     # csdmbase
csdm            1472/udp     # csdm
csdm            1472/tcp     # csdm
#                          Robert Stabl <stabl@informatik.uni-muenchen.de>
openmath        1473/udp     # OpenMath
openmath        1473/tcp     # OpenMath
#                          Garth Mayville <mayville@maplesoft.on.ca>
telefinder      1474/udp     # Telefinder
```

Finding Open Ports

- To find open ports:
  - Can use a port scanner over the network that systematically tries out all ports
  - Can use systems tools
    - MacOS:

```
thomasschwarz@Peter-Canisius ~ % lsof -i -P | grep -i "listen"
rapportd    527 thomasschwarz    5u  IPv4 0xc604072814ca13a1      0t0  TCP *:62127 (LISTEN)
rapportd    527 thomasschwarz    9u  IPv6 0xc604072814573699      0t0  TCP *:62127 (LISTEN)
ControlCe  1666 thomasschwarz   12u  IPv4 0xc604072801237e41      0t0  TCP *:7000 (LISTEN)
ControlCe  1666 thomasschwarz   13u  IPv6 0xc6040727f7e8ad79      0t0  TCP *:7000 (LISTEN)
ControlCe  1666 thomasschwarz   16u  IPv4 0xc6040727f968c381      0t0  TCP *:5000 (LISTEN)
ControlCe  1666 thomasschwarz   17u  IPv6 0xc6040728037e4b39      0t0  TCP *:5000 (LISTEN)
mongod     1736 thomasschwarz   10u  IPv4 0xc604072814a513a1      0t0  TCP localhost:27017 (LISTEN)
Google    62219 thomasschwarz  136u  IPv4 0xc604072814c9fe41      0t0  TCP localhost:49787 (LISTEN)
```

Finding Open Ports

- On Windows:
  - netstat
  - nbtstat

Socket Address

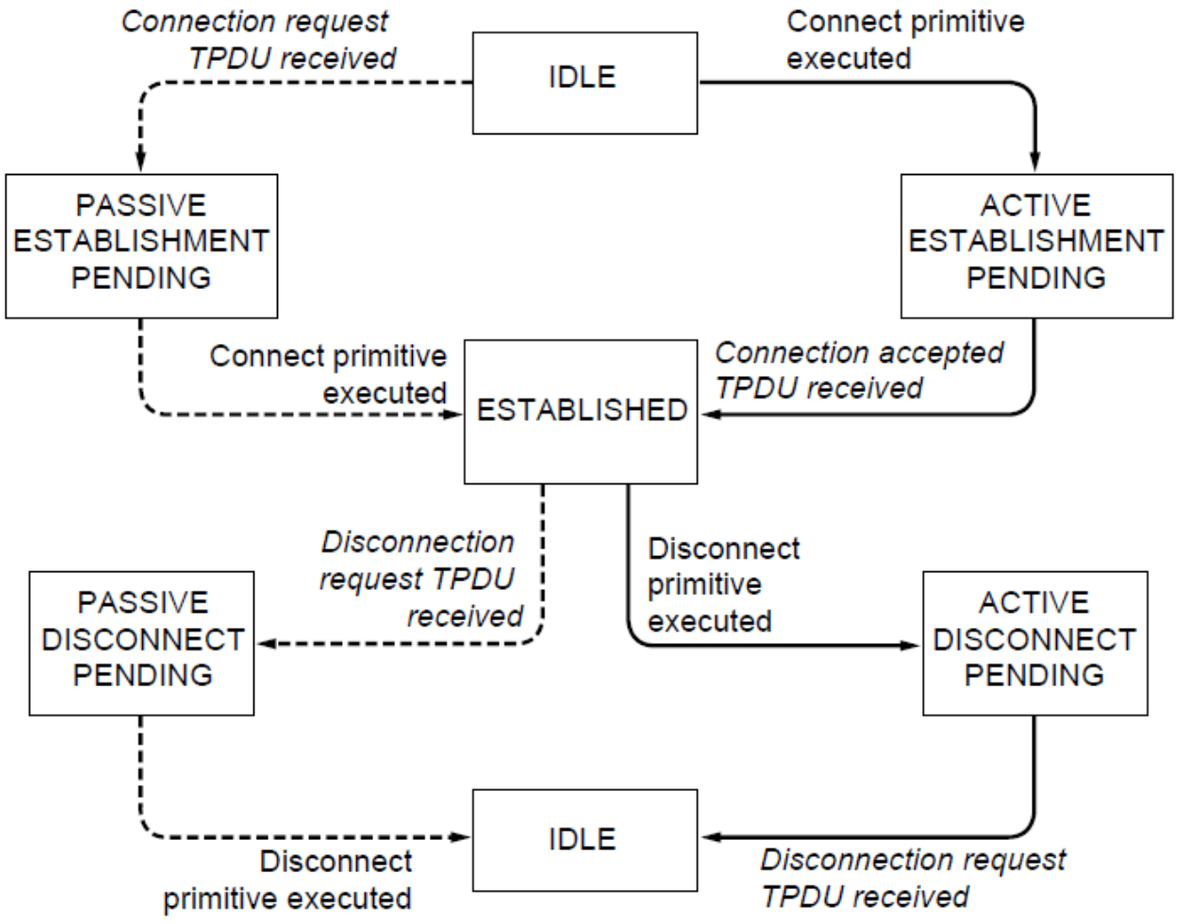
- The combination of IP address and port number is the *socket address*

Transport Service Primitives

- Primitives that applications might call to transport data for a simple connection-oriented service:
  - Client calls connect, send, receive, disconnect
  - Server calls listen, receive, send, disconnect

Primitive	Segment sent	Meaning
LISTEN	(none)	Block until some process tries to connect
CONNECT	CONNECTION REQ.	Actively attempt to establish a connection
SEND	DATA	Send information
RECEIVE	(none)	Block until a DATA packet arrives
DISCONNECT	DISCONNECTION REQ.	This side wants to release the connection

Transport Service Primitives



Solid lines (right) show client state sequence

Dashed lines (left) show server state sequence

Transitions in italics are due to segment arrivals.

Addressing

- How does an application find port numbers?
  - Portmapper (which listens at a well known port)
    - User sends service name and gets port address
    - Services must register with the portmapper
  - Initial connection protocol
    - Each machine with services has a process server that acts as proxy for less heavily used servers
      - inetd on Unix systems
    - Listens to a range of ports waiting for connection requests
    - Process server spawns requested server (if necessary)