Network Programming

Lecture 01

Haitham A. El-Ghareeb

Faculty of Computers and Information Sciences
Mansoura University
Egypt
helghareeb@mans.edu.eg

February 12, 2017

Contacts

about.me/helghareeb

Administrative

TCP/IP Network Stack

- Understanding TCP/IP
- Objectives of TCP/IP
- Key characteristics of TCP/IP

Data Transmission

- Data Transmission Layers
- User Area
- POSIX
- Kernel Socket
- TCP Control Block
- TCP Frame Structure
- IP Layer
- Ethernet Layer
- NIC
- However

Next Week InchALLAH

Homework

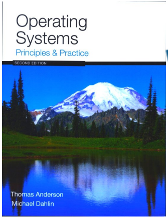
Topics

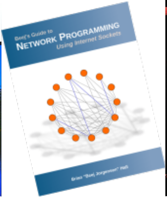
- TCP/IP Suite Review
- Unix/Linux Networking Basics
- IP4 and IP6
- Socket Programming (C++/Python/Java)
- Client - Server Architecture Model
- Network Protocols and Services
- Network Routing Protocols
- SDN

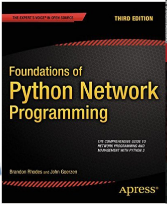
Lab

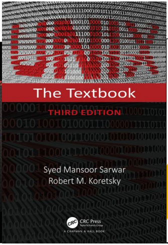
- Unix/Linux Networking Basics
- Socket Programming (C++/Python/Java)
- Packet Tracer
- Mininet

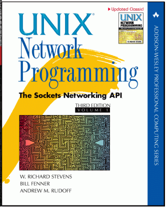
References


Operating Systems
Principles & Practice
SECOND EDITION
Thomas Anderson
Michael Dahlin


Beej's Guide to
Network Programming
Using Internet Sockets
Brian "Beej Jorgensen" Hall


THE EXPERT'S VOICE® IN OPEN SOURCE
THIRD EDITION
Foundations of Python Network Programming
THE COMPREHENSIVE GUIDE TO NETWORK PROGRAMMING AND MANAGEMENT WITH PYTHON 3
Brandon Rhodes and John Goerzen
Apress®


The Textbook
THIRD EDITION
Syed Mansoor Sarwar
Robert M. Koretsky
CRC Press
A CHAPMAN & HALL BOOK


Updated Classic!
UNIX Network Programming
The Sockets Networking API
THIRD EDITION
VOLUME 1
W. RICHARD STEVENS
BILL FENNER
ANDREW M. RUDOFF
ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

Marks

- 60 Marks - Final Exam
- 10 Marks - Oral Exam
- 10 Marks - Midterm Exam
- 10 Marks - Practical Exam
- 10 Marks - Project
- 100 Marks - Total

OFFICE HOURS

OPEN Most Days About 9 or 10
Occasionally as Early as 7,
But SOME DAYS As Late As 12 or 1
WE CLOSE About 5:30 or 6
Occasionally About 4 or 5
But Sometimes As Late as 11 or 12.
SOME DAYS OR Afternoons,
We Aren't Here At All, and Lately
I've Been Here Just About All The Time,
Except When I'm Someplace Else,
But I Should Be Here Then, Too.

Let’s Get Started!

Following content is based on `http://www.cubrid.org/blog/dev-platform/understanding-tcp-ip-network-stack/`

Understanding TCP/IP Network Stack

- Understand how data is transfered

Understanding TCP/IP Network Stack

- Understand how data is transfered
    - improve performance through tuning

Understanding TCP/IP Network Stack

- Understand how data is transfered
    - improve performance through tuning
    - improve performance through troubleshooting

Understanding TCP/IP Network Stack

- Understand how data is transfered
    - improve performance through tuning
    - improve performance through troubleshooting
    - introduction to a new technology

Objectives of TCP/IP

Objectives

- Transmit data quickly
- Keep data order
- Without any data loss

Key characteristics of TCP/IP

- Connection-Oriented
- Bidirectional Byte stream
- In-order Delivery
- Reliable through ACK
- Flow Control
- Congestion Control

Connection-Oriented

- First, a connection is made between two endpoints (local and remote)
- and then data is transferred.
- ”TCP connection identifier” is a combination of addresses of the two endpoints, having

```
<local IP address, local port number,
   remote IP address, remote port number>
```

Bidirectional Byte Stream

Bidirectional data communication is made by using byte stream.

In-order Delivery

In-order Delivery

- A receiver receives data in the order of sending data from a sender.

In-order Delivery

- A receiver receives data in the order of sending data from a sender.
- For that, the order of data is required.

In-order Delivery

- A receiver receives data in the order of sending data from a sender.
- For that, the order of data is required.
- To mark the order, 32-bit integer data type is used.

Reliability Through ACK

Reliability Through ACK

- When a sender did not receive ACK from a receiver after sending data, sender TCP re-sends the data.

Reliability Through ACK

- When a sender did not receive ACK from a receiver after sending data, sender TCP re-sends the data.
- Therefore, sender TCP buffers unacknowledged data from receiver.

Flow Control

Flow Control

- Sender sends as much data as a receiver can afford.

Flow Control

- Sender sends as much data as a receiver can afford.
- Receiver sends the maximum number of bytes that it can receive (unused buffer size, receive window) to sender.

Flow Control

- Sender sends as much data as a receiver can afford.
- Receiver sends the maximum number of bytes that it can receive (unused buffer size, receive window) to sender.
- Sender sends as much data as the size of bytes that receiver's receive window allows.

Congestion Control

Congestion Control

- Congestion Window: used separately from receive window to prevent network congestion by limiting the volume of data flowing in the network.

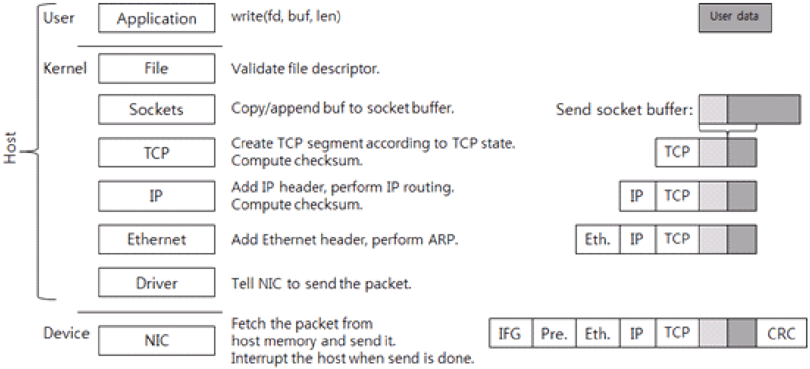
Congestion Control

- Congestion Window: used separately from receive window to prevent network congestion by limiting the volume of data flowing in the network.
- Sender sends as much data as the size of bytes that the receiver's congestion windows allows by using a variety of algorithms, such as
    - TCP Vegas
    - Westwood
    - BIC
    - CUBIC

Congestion Control

- Congestion Window: used separately from receive window to prevent network congestion by limiting the volume of data flowing in the network.
- Sender sends as much data as the size of bytes that the receiver's congestion windows allows by using a variety of algorithms, such as
  - TCP Vegas
  - Westwood
  - BIC
  - CUBIC
- Different from flow control, congestion control is implemented by the sender only.

Data Transmission


Host
User
Application
write(fd, buf, len)
User data
Kernel
File
Validate file descriptor.
Sockets
Copy/append buf to socket buffer.
Send socket buffer:
TCP
Create TCP segment according to TCP state.
Compute checksum.
TCP
IP
Add IP header, perform IP routing.
Compute checksum.
IP
TCP
Ethernet
Add Ethernet header, perform ARP.
Eth.
IP
TCP
Driver
Tell NIC to send the packet.
Device
NIC
Fetch the packet from
host memory and send it.
Interrupt the host when send is done.
IFG
Pre.
Eth.
IP
TCP
CRC

Operation Process by Each Layer of TCP/IP Network Stack for Data Transmission.

Data Transmission

There are several layers that are briefly classified into three areas:

- User area
- Kernel area
- Device area

host vs. device

host vs. device

- Tasks at user area and kernel area are performed by CPU

host vs. device

- Tasks at user area and kernel area are performed by CPU
- user area and kernel area are called "host" to distinguish them from device area

host vs. device

- Tasks at user area and kernel area are performed by CPU
- user area and kernel area are called ”host” to distinguish them from device area
- here, device is NIC that sends and receives packets

host vs. device

- Tasks at user area and kernel area are performed by CPU
- user area and kernel area are called "host" to distinguish them from device area
- here, device is NIC that sends and receives packets
- NIC is more accurate term than "LAN card"

User Area

User Area

- First, application creates data to send ”User data”

User Area

- First, application creates data to send ”User data”
- Then, calls the

```
write()
```

User Area

- First, application creates data to send ”User data”
- Then, calls the

```
write()
```

- system call to send the data.

User Area

- First, application creates data to send ”User data”
- Then, calls the

```
write()
```

- system call to send the data.
- Assume **fd** has been created. When system call is called, the area is switched to kernel area.

POSIX

POSIX

- POSIX-series operating systems (*nix) exposes socket to application using file descriptor.

POSIX

- POSIX-series operating systems (*nix) exposes socket to application using file descriptor.
- In POSIX, socket is a kind of file.

POSIX

- POSIX-series operating systems (*nix) exposes socket to application using file descriptor.
- In POSIX, socket is a kind of file.
- File layer executes a simple examination and calls the socket function by using the socket structure connected to file structure.

Kernel Socket

Kernel socket has two buffers:

1. One is the **send socket buffer** for sending
2. And the other is the **receive socket buffer** for receiving

When the write system call is called:

Kernel Socket

Kernel socket has two buffers:

1. One is the **send socket buffer** for sending
2. And the other is the **receive socket buffer** for receiving

When the write system call is called:

- data in the user area is copied to the kernel memory

Kernel Socket

Kernel socket has two buffers:

1. One is the **send socket buffer** for sending
2. And the other is the **receive socket buffer** for receiving

When the write system call is called:

- data in the user area is copied to the kernel memory
- and then added to the end of the send socket buffer

Kernel Socket

Kernel socket has two buffers:

1. One is the **send socket buffer** for sending
2. And the other is the **receive socket buffer** for receiving

When the write system call is called:

- data in the user area is copied to the kernel memory
- and then added to the end of the send socket buffer
- This is to send the data in order

Kernel Socket

Kernel socket has two buffers:

1. One is the **send socket buffer** for sending
2. And the other is the **receive socket buffer** for receiving

When the write system call is called:

- data in the user area is copied to the kernel memory
- and then added to the end of the send socket buffer
- This is to send the data in order
- Then, TCP is called

TCP Control Block

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are
    - **connection state**

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are
  - **connection state**
  - **receive window**

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are
  - **connection state**
  - **receive window**
  - **congestion window**

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are
  - **connection state**
  - **receive window**
  - **congestion window**
  - **sequence number**

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are
  - **connection state**
  - **receive window**
  - **congestion window**
  - **sequence number**
  - **resending timer**

TCP Control Block

- There is the TCP Control Block (TCB) structure connected to the socket.
- TCB includes data required for processing the TCP connection.
- Data in the TCB are
    - **connection state**
    - **receive window**
    - **congestion window**
    - **sequence number**
    - **resending timer**
    - **etc**

```
ESTABLISHED, SYN_SENT, SYN_RECV, FIN_WAIT1, FIN_WAIT2,
    TIME_WAIT, CLOSE, CLOSE_WAIT, LAST_ACK, LISTEN, CLOSING,
     UNKNOWN
```

Connection States

- **ESTABLISHED** The socket has an established connection.
- **SYN_SENT** The socket is actively attempting to establish a connection.
- **SYN_RECV** A connection request has been received from the network.
- **FIN_WAIT1** The socket is closed, and the connection is shutting down.
- **FIN_WAIT2** Connection is closed, and the socket is waiting for a shutdown from the remote end.
- **TIME_WAIT** The socket is waiting after close to handle packets still in the network.

Connection States (contd.)

- **CLOSE** The socket is not being used.
- **CLOSE_WAIT** The remote end has shut down, waiting for the socket to close.
- **LAST_ACK** The remote end has shut down, and the socket is closed. Waiting for acknowledgement.
- **LISTEN** The socket is listening for incoming connections. Such sockets are not included in the output unless you specify the –listening (-l) or –all (-a) option.
- **CLOSING** Both sockets are shut down but we still don't have all our data sent.
- **UNKNOWN** The state of the socket is unknown.

TCP State

TCP State

- If current TCP state allows for data transmission, a new TCP segment is created.

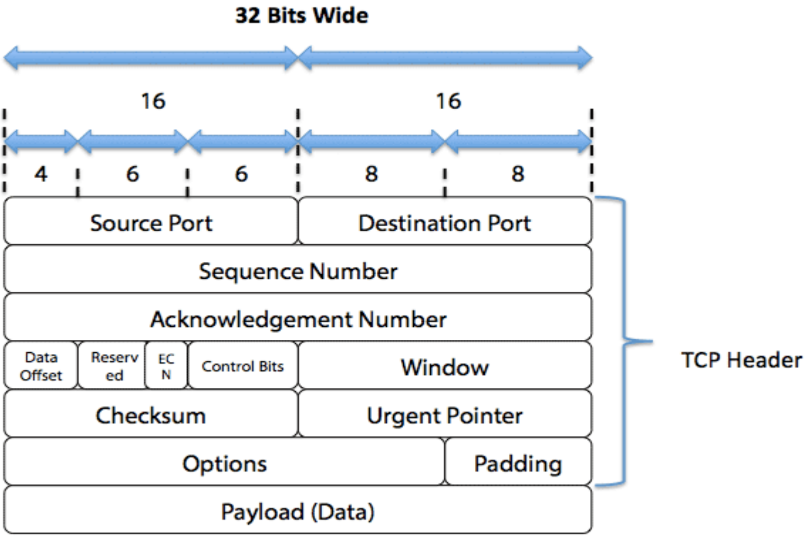
TCP State

- If current TCP state allows for data transmission, a new TCP segment is created.
- If data transmission is impossible, due to flow control or such a reason, the system call is ended here and then mode is returned to user mode (control passed to the application)

There are two TCP segments:

1. TCP header
2. Payload

TCP Frame Structure



32 Bits Wide

16 16

4 6 6 8 8

Source Port | Destination Port

Sequence Number

Acknowledgement Number

Data Offset | Reserved | ECN | Control Bits | Window

Checksum | Urgent Pointer

Options | Padding

Payload (Data)

TCP Header

Payload

Payload

- Payload includes data saved in the unacknowledged send socket buffer.

Payload

- Payload includes data saved in the unacknowledged send socket buffer.
- Maximum length of Payload is the maximum value among the receive window, congestion window, and maximum segment size (MSS).

TCP Checksum

TCP Checksum

- Then, TCP checksum is computed.

TCP Checksum

- Then, TCP checksum is computed.
- Pseudo header information that are included are:
    - IP addresses
    - segment length
    - protocol number

TCP Checksum

- Then, TCP checksum is computed.
- Pseudo header information that are included are:
    - IP addresses
    - segment length
    - protocol number
- One or more packets can be transmitted according to the TCP state.

Checksum Offload

- TCP checksum is computed by NIC, not by the kernel.
- Why?

IP Layer

- The created TCP segment goes down to the IP layer, that does:
    - Adds IP header to the TCP segment
    - Performs IP routing

IP Layer

- The created TCP segment goes down to the IP layer, that does:
    - Adds IP header to the TCP segment
    - Performs IP routing

    - **IP routing** is a procedure of searching the next hop IP in order to go to the destination IP.

Ethernet Layer

Ethernet Layer

- Searches for the MAC address of the next hop IP by using the ARP.
- It then adds the Ethernet header to the packet.
- Host packet is completed by adding the Ethernet header.

NIC

- After IP routing is performed, the transmit interface (NIC) is knows as the result of IP routing.
- The interface is used for transmitting a packet to the next hop IP and the IP.
- Therefore, the **transmit NIC driver** is called.
- At this time, if a packet capture program such as tcpdump or wireshark is running, the kernel copies the packet data onto the memory buffer that the program uses.
- In that way, the receiving packet is directly captured on the driver.
- Generally, the traffic shaper function is implemented to run on this layer.

NIC Driver

- NIC driver requests packet transmission according to the driver-NIC communication protocol defined by the NIC manufacturer.
- After receiving the **packet transmission request**, NIC copies the packets from the **main memory** to **NIC memory** and then sends it to the network line.
- At this time, by complying with the Ethernet standard, it adds the Inter-Frame Gap (IFG), preamble, and CRC to the packet.
  - IFG and preamble are used to distinguish the start of the packet (framing).
  - CRC is used to protect the data
- Packet transmission is started based on the physical speed of the Ethernet and the condition of Ethernet flow control.

NIC Interrupts

- When NIC sends a packets, NIC generates interrupts to the host CPU.
- Every interrupt has its own interrupt number, and the OS searches an adequate driver to handle the interrupt by using the number.
- The driver registers a function to handle the interrupt (an interrupt handler) when the driver is started.
- OS calls the interrupt handler and then the interrupt handler returns the transmitted packet to the OS.

However

- So far, we have discussed the procedure of data transmission through the kernel and the device when an application performs write.
- However, without a direct write request from the application, the kernel can transmit a packet directly calling TCP.
- For example ?!!

However

- So far, we have discussed the procedure of data transmission through the kernel and the device when an application performs write.
- However, without a direct write request from the application, the kernel can transmit a packet directly calling TCP.
- For example ?!!
- When an ACK is received and the receive window is expanded, the kernel creates a TCP segment including the data left in the socket buffer and sends the TCP segment to the receiver.

Next Week InchALLAH

Next Lecture

- Data Receiving
- Data Structure
- Following Code

Next Lab

- Review TCP/IP Suite (Solve selected Qs)

Homework

- Study this lecture (very well)
- Prepare for the Next Lecture
- Check some Kernel code (mainly Networking Subsystem)

YOU GO
NOW!