EE 472 – Embedded Systems

Dr. Shwetak Patel
Assistant Professor
Computer Science & Engineering
Electrical Engineering

Agenda

- Lab 3
- Finish semaphores
- Working outside the processor
  - The Local and Remote Models
  - TCP/IP
Semaphores

- A type of flagging scheme
- Used to protect critical sections
- Atomic operation
- Have two operations:
  - Wait(s) or Pend(s) or P(s)
  - Signal(s) or Post(s) or V(s)

Deadlock

- Two or more processes wait indefinitely
  - When two processes share the same two resources
Connecting to Outside Devices

- Interface with sensors
  - Ex: GPS, CCD, etc

- Connect to other embedded platforms
  - Ex: The CAN bus found in a car, networking

- Actuators
  - Ex: Motor controls, displays, audio system

Basic Concepts

- Physical layer
  - Ex: Copper wire, wireless RX / TX hardware

- Protocol
  - Specification for communication
Models

- Local Model
- Remote Model

Model Components

- Address
- Control
- Data
Local Models

- Serial
- Parallel

Local Model Architectures

- Memory mapped
- Program controlled I/O Ports
- Peripheral processor
Memory Mapped Approach

- Looks like a memory address in the processor
- Read and write to addresses
- Advantages/Disadvantages?

Program Controlled IO

- Similar to memory mapping approach
- Uses a separate transport medium
Control and Synchronization

- Shared variables
- Messages
- Polling
- Interrupts

Asynchronous

- No clock or timing
- Burst data rates
Synchronous

- Predetermined data rate
- Use a clock
- Can encode timing in the data

The Remote Model

- Local Model – up to 5 meters
- New challenges, but similar concepts to local model
- Architecture has multiple layers (stack)
OSI Architecture

- Open Systems Interconnection Model (OSI)

- 7 layers
  - Physical
  - Data Link
  - Network
  - Transport
  - Session
  - Presentation
  - Application

TCP/IP

- TCP/IP is a common well known example

- 4 layers
  - Physical
  - Network
  - Transport
  - Application
Models of Use

- Client-Server
- Peer to peer
- Multicast

Networking/Internet

- Basics
  - Data
  - Address
- Infrastructure
  - Physical links
  - Routers
  - Hubs
## Protocol Stack

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>TCP/IP Hierarchy</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>HTTP</td>
<td>SMTP, POP3, FTP, …</td>
</tr>
<tr>
<td>Presentation</td>
<td></td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>Session</td>
<td>Application Layer</td>
<td></td>
</tr>
<tr>
<td>Transport</td>
<td>Transport Layer</td>
<td></td>
</tr>
<tr>
<td>Network</td>
<td>Network Layer</td>
<td>IP, ICMP</td>
</tr>
<tr>
<td>Link</td>
<td>Link Layer</td>
<td>ARP, RARP, Ethernet, PPP, …</td>
</tr>
<tr>
<td>Physical</td>
<td>Physical Layer</td>
<td></td>
</tr>
</tbody>
</table>

## Packet Creation

- Data is sent down the protocol stack
- Each layer adds to the data by prepending headers
Link Layer

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>TCP/IP Hierarchy</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>7th</td>
<td>Application Layer</td>
<td>HTTP</td>
</tr>
<tr>
<td>6th</td>
<td>Presentation Layer</td>
<td>SMTP</td>
</tr>
<tr>
<td>5th</td>
<td>Session Layer</td>
<td>POP3</td>
</tr>
<tr>
<td>4th</td>
<td>Transport Layer</td>
<td>FTP</td>
</tr>
<tr>
<td>3rd</td>
<td>Network Layer</td>
<td></td>
</tr>
<tr>
<td>2nd</td>
<td>Link Layer</td>
<td></td>
</tr>
<tr>
<td>1st</td>
<td>Physical Layer</td>
<td></td>
</tr>
</tbody>
</table>

- Link Layer - Ethernet
  - Computer to Computer communication on the same network
  - Each device has unique media access control (MAC) address (48-bit) ex: 00-C0-4F-48-47-93

Ethernet Packet:

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preamble</td>
<td>8 bytes</td>
</tr>
<tr>
<td>Dest. address</td>
<td>6 bytes</td>
</tr>
<tr>
<td>Source address</td>
<td>6 bytes</td>
</tr>
<tr>
<td>Type</td>
<td>2 bytes</td>
</tr>
<tr>
<td>Data</td>
<td>64 - 1500 bytes</td>
</tr>
<tr>
<td>CRC</td>
<td>4 bytes</td>
</tr>
</tbody>
</table>
ARP : Address Resolution Protocol

- ARP provides mapping
  32bit IP address <-> 48bit MAC address
  128.97.89.153 <-> 00-C0-4F-48-47-93

- ARP cache
  maintains the recent mappings from IP addresses to MAC addresses

Protocol

1. ARP request broadcast on Ethernet
2. Destination host ARP layer responds

Network Layer
**IP: Internet Protocol**

- Unreliable, connectionless datagram delivery service
- Responsible for routing of data through intermediate networks and computers (uses IP address)

**IP header:**

<table>
<thead>
<tr>
<th>Version</th>
<th>IHL</th>
<th>Type of Service</th>
<th>Total Length</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Identification</th>
<th>Flags</th>
<th>Fragment Offset</th>
<th>Time to Live</th>
<th>Protocol</th>
<th>Header Checksum</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td>Destination IP address</td>
<td></td>
<td>IP data payload (many bytes)</td>
</tr>
</tbody>
</table>

**IP Routing**

- All the intermediate routers have a routing Table
  - Destination IP address
  - IP address of a next-hop router
Transport Layer

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>TCP/IP Hierarchy</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>7th</td>
<td>Application Layer</td>
<td>HTTP, SMTP, POP3, FTP, ...</td>
</tr>
<tr>
<td>6th</td>
<td>Presentation Layer</td>
<td>Application Layer</td>
</tr>
<tr>
<td>5th</td>
<td>Session Layer</td>
<td>Transport Layer</td>
</tr>
<tr>
<td>4th</td>
<td>Transport Layer</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>3rd</td>
<td>Network Layer</td>
<td>IP, ICMP</td>
</tr>
<tr>
<td>2nd</td>
<td>Link Layer</td>
<td>ARP, RARP, Ethernet, PPP, ...</td>
</tr>
<tr>
<td>1st</td>
<td>Physical Layer</td>
<td>TCP Payload</td>
</tr>
</tbody>
</table>

TCP : Transmission Control Protocol

- Reliable transmission, ordered deliver, congestion control Protocol
- 1. Set up connection, 2. Transfer data, 3. Close connection

<table>
<thead>
<tr>
<th>TCP Header Format</th>
<th>Source Port</th>
<th>Destination Port</th>
<th>Sequence Number</th>
<th>Acknowledgement Number</th>
<th>Data Offset</th>
<th>Window</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Checksum</td>
<td>Urgent Pointer</td>
<td></td>
<td></td>
<td>Options (0 to 10 Words of 32 Bits)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td>TCP Payload</td>
<td></td>
</tr>
</tbody>
</table>
TCP : Data transfer

Server

Send Packet 1
Start Timer

ACK arrives at this time
Cancel Timer

Timer

Packet arrives
ACK sent back

Client

Send Packet 2
Start Timer

Receive ACK 2
Cancel Timer

Timer

Receive Packet 2
Send ACK 2

Packet Lost

Timer

Packet should arrive
ACK should be sent

Send Packet 1
Start Timer

ACK would normally arrive at this time

Time Expires

Retransmit Packet 1
Start Timer

Receive ACK 1
Cancel Timer

Receive Packet 1
Send ACK 1
TCP vs. UDP

- UDP -> “best effort” – snail mail

- TCP -> reliable – snail mail with delivery confirmation

Application Layer
HTTP : Hyper Text Transfer Protocol

- Webpages are a bunch of text files
- Each transaction creates a new connection

Steps in Transaction
1. Establish connection
2. Request
   
   GET <URL> <CR>

3. Response
   
   Response Code <Data> <CR>

4. Close connection

Ex: Access www.washington.edu

Client | Server
---|---
HTTP | HTTP
Transp. | Transp.
TCP | TCP
Net. | Net.
IP | IP
Link | Link
ethernet | ethernet
Ex: Access www.washington.edu

Client

App.
HTTP
GET "http://www.washington.edu"<CR>

HTTP
Transp.
TCP
Initiate connection (hdshk)
Package data (add TCP header)
send http request packet

Net.
IP
Link
ethernet
send data to next hop

Net.
IP
Link
ethernet
Router(s)

Server

App.
HTTP

HTTP
Transp.
TCP

Net.
IP
Link
ethernet
Router(s)

Ex: Access www.washington.edu

Client

App.
HTTP

HTTP
Transp.
TCP

Net.
IP
Link
ethernet
Router(s)

Server

App.
HTTP

HTTP
Transp.
TCP

Net.
IP
Link
ethernet
Router(s)

Http response
200 "" <CR> <html file in MIME format>

Assemble response
Send http response packets

Relay data
Ex: Access www.washington.edu

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>HTTP</td>
<td>HTTP</td>
</tr>
<tr>
<td>Transp.</td>
<td>Transp.</td>
</tr>
<tr>
<td>TCP</td>
<td>TCP</td>
</tr>
<tr>
<td>Net.</td>
<td>Net.</td>
</tr>
<tr>
<td>IP</td>
<td>IP</td>
</tr>
<tr>
<td>Link</td>
<td>Link</td>
</tr>
<tr>
<td>ethernet</td>
<td>ethernet</td>
</tr>
</tbody>
</table>

Send data to next hop

ARP to provide IP/MAC translation

Relay data

Close connection (hdshk)

Protocol Stack

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>TCP/IP Hierarchy</th>
<th>Protocols</th>
</tr>
</thead>
<tbody>
<tr>
<td>7th</td>
<td>Application Layer</td>
<td>HTTP</td>
</tr>
<tr>
<td>6th</td>
<td>Presentation Layer</td>
<td>SMTP, POP3, FTP, ...</td>
</tr>
<tr>
<td>5th</td>
<td>Session Layer</td>
<td>TCP, UDP</td>
</tr>
<tr>
<td>4th</td>
<td>Transport Layer</td>
<td>IP, ICMP</td>
</tr>
<tr>
<td>3rd</td>
<td>Network Layer</td>
<td>ARP, RARP</td>
</tr>
<tr>
<td>2nd</td>
<td>Link Layer</td>
<td>Ethernet, PPP, ...</td>
</tr>
<tr>
<td>1st</td>
<td>Physical Layer</td>
<td></td>
</tr>
</tbody>
</table>
Sockets

- Programming construct for sending data over a network session
- Allows you to treat the connections as I/O (i.e. send() or receive())

```
Server
socket()  .......... Open communication and print
           ↓
bind()     .......... Register well-known address with the system
           ↓
listen()   .......... Establish clients connection requests queue size
           ↓
accept()   .......... Accepts tcp server connection request on the queue
           ↓
bloks until connection from client

Client
socket()  .......... Open communication and print
           ↓
connect()  .......... Set up connection to server
           ↓
read()     .......... Send/receive data
           ↓
write()    .......... Send/receive data

close()    .......... Shut down
```
Make API

- Socket connections
- Web server API

Questions?