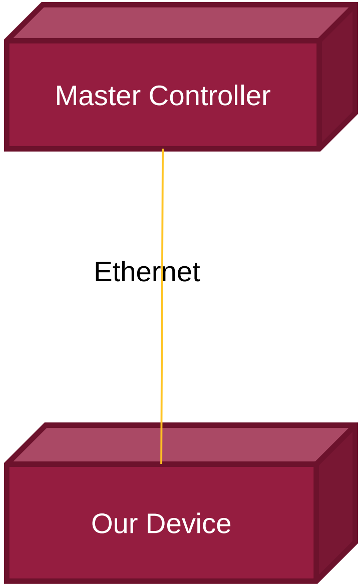
SER486: Embedded C Programming

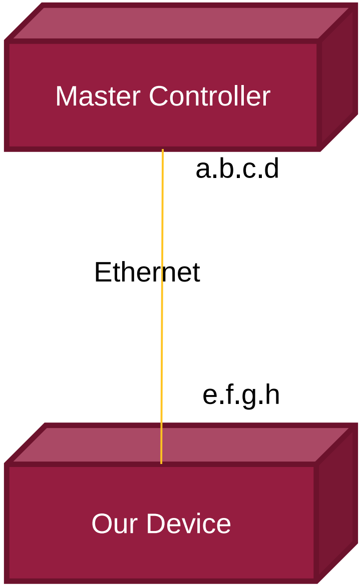
Networking:
DHCP and NTP Architecture

Arizona State University

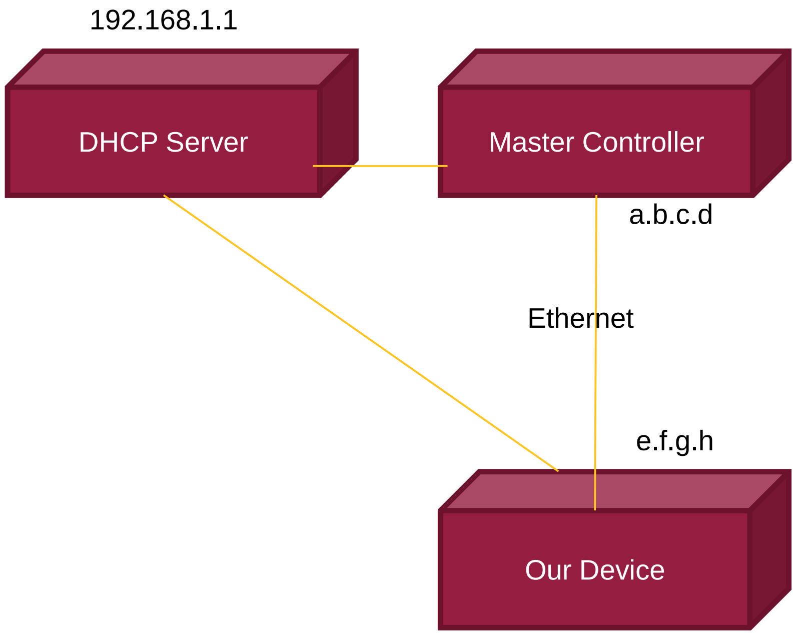
Our goal: Communicate with the Master Controller



This requires IP addresses

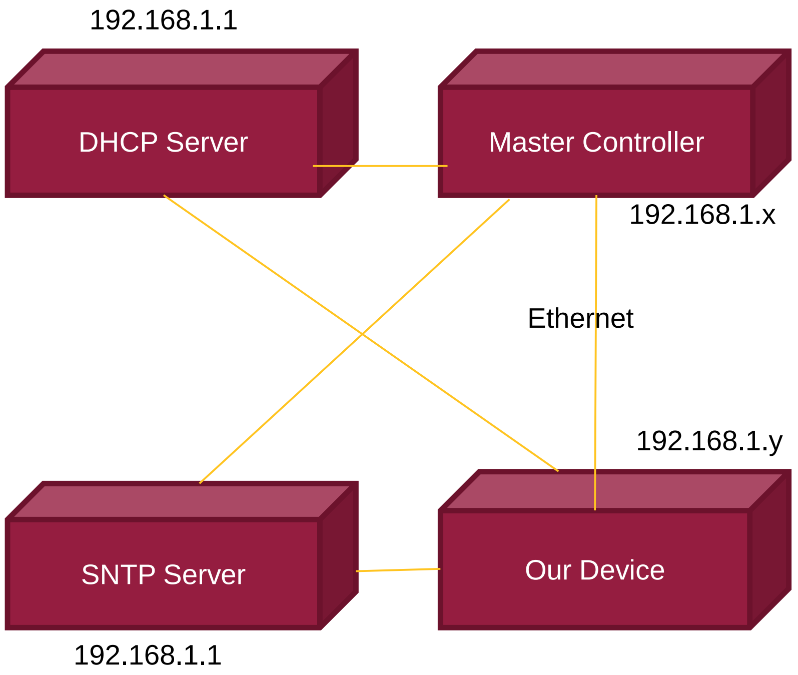


Dynamic IP Addresses Require DHCP



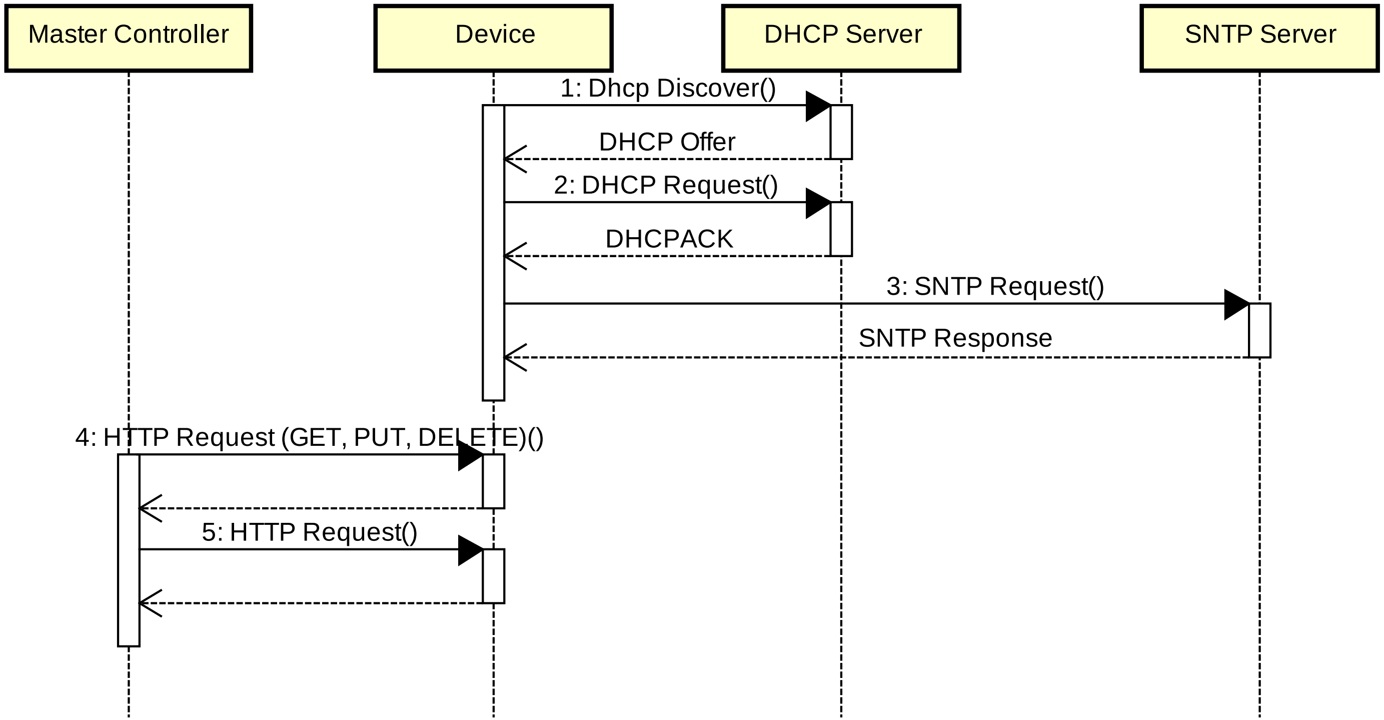
- Most common mechanism that hosts are assigned their IP addresses.
- Defined by RFC 2131.
  - packet formats
  - communications sequences
- Performed over UDP
  - DHCP protocol supports retries
- Requires one DHCP server to provide addresses

Time synchronization via SNTP

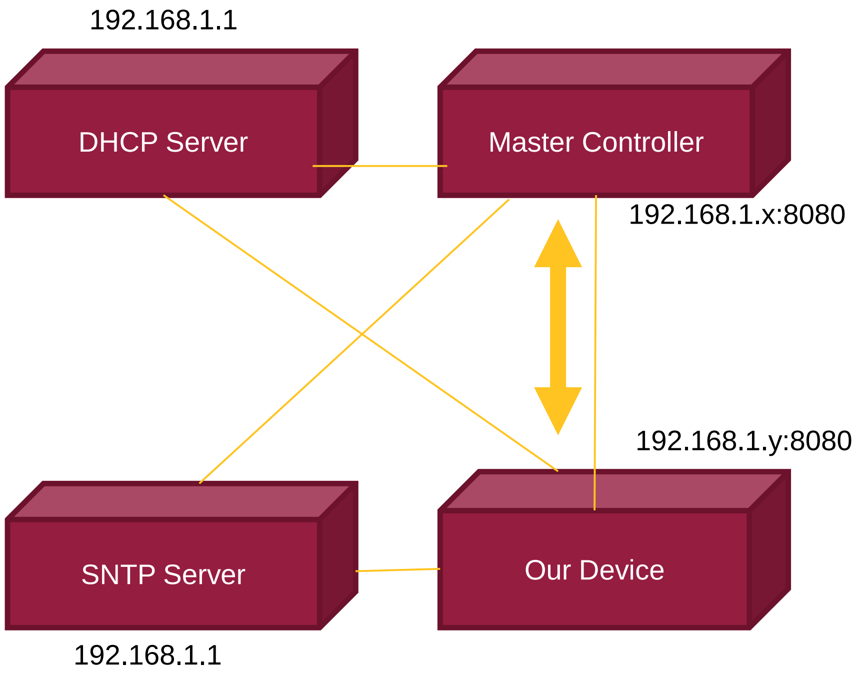


- Common mechanism for time synchronization for embedded devices
- Defined by RFC 4330
  - packet formats
  - communications sequences
- Performed over UDP
- Requires one SNTP server to provide time

Startup sequence



Web API interface



- Will use GET/PUT/DELETE methods over HTTP
- GET
  - Master controller requests information from the device
- PUT
  - Master controller requests device to replace certain information
- DELETE
  - Master controller requests device to delete information

Where do the devices reside?

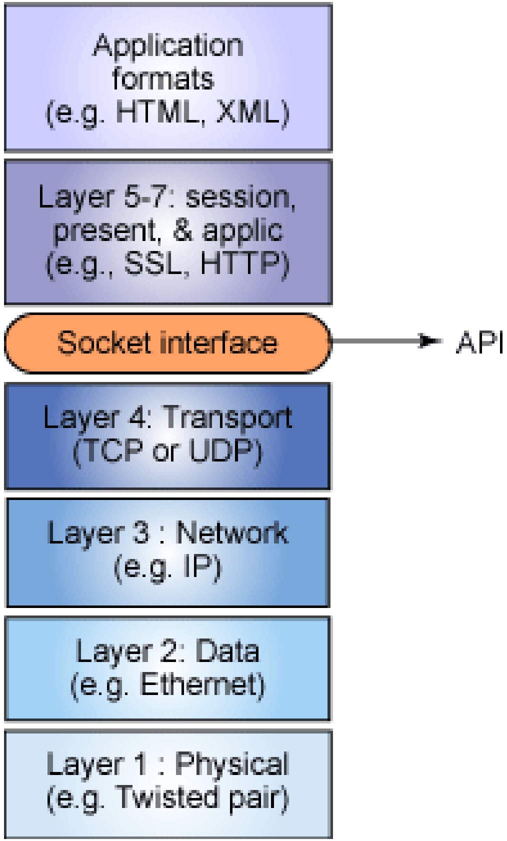
Device	Address	Notes
Local Host	127.0.0.1	
DHCP Server	127.0.0.1	Simulated server. Available to simulated IIoT device but not localhost.
NTP Server	127.0.0.1	Simulated server. Available to simulated IIoT device but not localhost.
IIoT Device	127.0.0.100	Address assigned by virtual DHCP server

Device	Address	Notes
Development Host	192.168.1.x	Assigned by DHCP
DHCP Server	192.168.1.1	(subject to change)
NTP Server	192.168.1.1	(subject to change)
IIoT Device	192.168.1.y	Address assigned by DHCP server

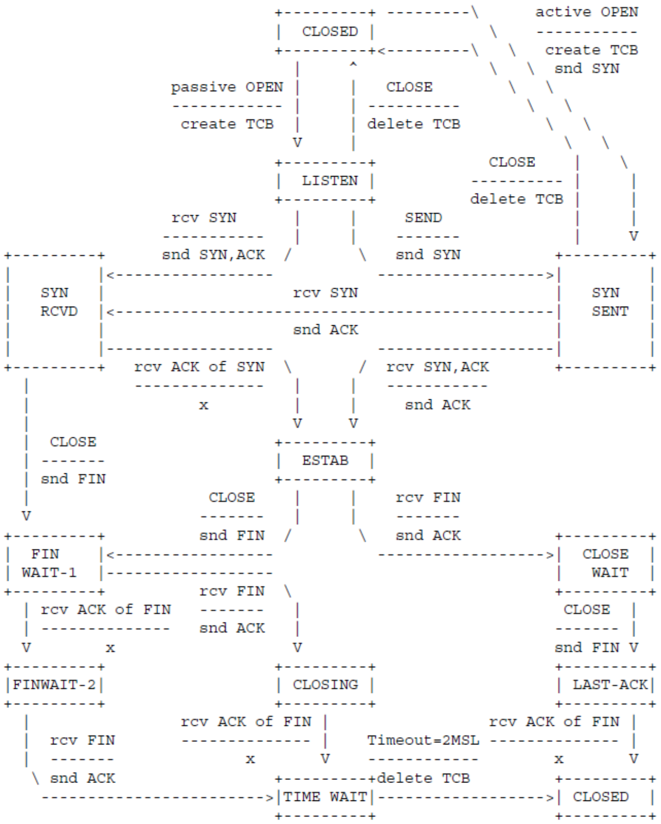
OUR Transport Protocol: TCP

FROM IETF RFC 793:

- **TCP is a connection-oriented, end-to-end reliable protocol designed to fit into a layered hierarchy of protocols which support multi-network applications.**
- **The TCP provides for reliable inter-process communication between pairs of processes in host computers attached to distinct but interconnected computer communication networks.**
- **Very few assumptions are made as to the reliability of the communication protocols below the TCP layer.**
- **TCP assumes it can obtain a simple, potentially unreliable datagram service from the lower level protocols.**
- **In principle, the TCP should be able to operate above a wide spectrum of communication systems ranging from hard-wired connections to packet-switched or circuit-switched networks.**


Application formats (e.g. HTML, XML)
Layer 5-7: session, present, & applic (e.g., SSL, HTTP)
Socket interface
API
Layer 4: Transport (TCP or UDP)
Layer 3 : Network (e.g. IP)
Layer 2: Data (e.g. Ethernet)
Layer 1 : Physical (e.g. Twisted pair)

TCP Connection Diagram



```
                              +---------+ ---------\      active OPEN
                              |  CLOSED |            \    -----------
                              +---------+<---------\   \   create TCB
                                |     ^              \   \  snd SYN
                   passive OPEN |     |   CLOSE        \   \
                   ------------ |     | ----------       \   \
                    create TCB  |     | delete TCB         \   \
                                V     |                      \   \
                              +---------+            CLOSE    |    \
                              |  LISTEN |          ---------- |     |
                              +---------+          delete TCB |     |
                   rcv SYN      |     |     SEND              |     |
                  -----------   |     |    -------            |     V
 +---------+      snd SYN,ACK  /       \   snd SYN          +---------+
 |         |<-----------------           ------------------>|         |
 |   SYN   |                    rcv SYN                     |   SYN   |
 |   RCVD  |<-----------------------------------------------|   SENT  |
 |         |                    snd ACK                     |         |
 |         |------------------           -------------------|         |
 +---------+   rcv ACK of SYN  \       /  rcv SYN,ACK       +---------+
   |           --------------   |     |   -----------
   |                  x         |     |     snd ACK
   |                            V     V
   |  CLOSE                   +---------+
   | -------                  |  ESTAB  |
   | snd FIN                  +---------+
   |                   CLOSE    |     |    rcv FIN
   V                  -------   |     |    -------
 +---------+          snd FIN  /       \   snd ACK          +---------+
 |  FIN    |<-----------------           ------------------>|  CLOSE  |
 | WAIT-1  |------------------                              |   WAIT  |
 +---------+          rcv FIN  \                            +---------+
   | rcv ACK of FIN   -------   |                            CLOSE  |
   | --------------   snd ACK   |                           ------- |
   V        x                   V                           snd FIN V
 +---------+                  +---------+                   +---------+
 |FINWAIT-2|                  | CLOSING |                   | LAST-ACK|
 +---------+                  +---------+                   +---------+
   |                rcv ACK of FIN |                 rcv ACK of FIN |
   |  rcv FIN       -------------- |    Timeout=2MSL -------------- |
   |  -------              x       V    ------------        x       V
    \ snd ACK                 +---------+delete TCB         +---------+
     ------------------------>|TIME WAIT|------------------>| CLOSED  |
                              +---------+                   +---------+
```

LISTEN - represents waiting for a connection request from any remote TCP and port.

SYN-SENT - represents waiting for a matching connection request after having sent a connection request.

SYN-RECEIVED - represents waiting for a confirming connection request acknowledgment after having both received and sent a connection request.

ESTABLISHED - represents an open connection, data received can be delivered to the user. The normal state for the data transfer phase of the connection.

FIN-WAIT-1 - represents waiting for a connection termination request from the remote TCP, or an acknowledgment of the connection termination request previously sent.

FIN-WAIT-2 - represents waiting for a connection termination request from the remote TCP.

CLOSE-WAIT - represents waiting for a connection termination request from the local user.

CLOSING - represents waiting for a connection termination request acknowledgment from the remote TCP.

LAST-ACK - represents waiting for an acknowledgment of the connection termination request previously sent to the remote TCP (which includes an acknowledgment of its connection termination request).

From RFC793

SER Socket State Control Functions

socket_open()

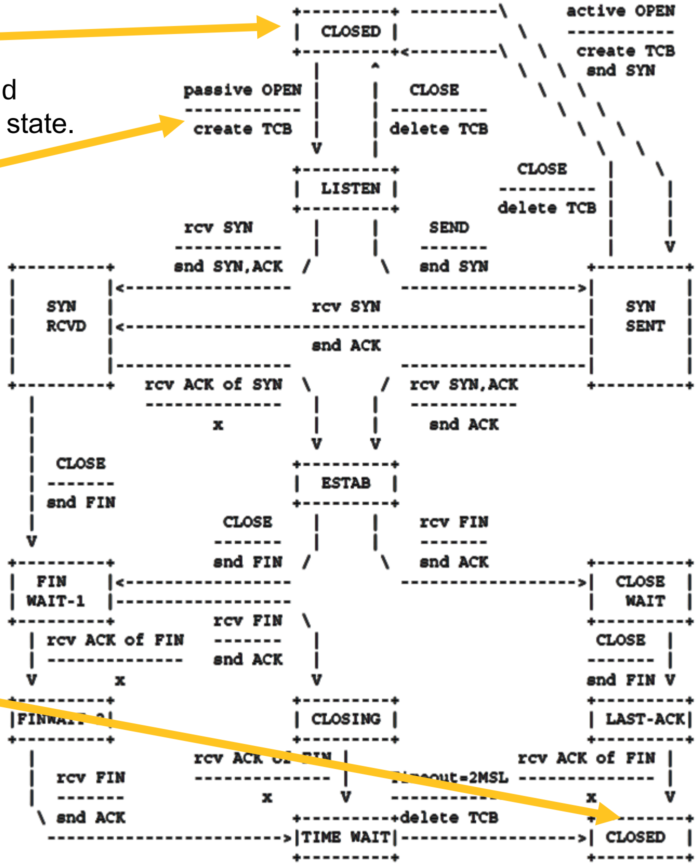
Initializes the socket and places it in the “closed” state.

socket_listen()

Places the socket in listen mode – transitions from “CLOSED” state to “LISTEN” state.

socket_close()

Force the port to be closed immediately, breaking any established connections. The socket returns to the “CLOSED” state.



From RFC793

Socket State Control Functions

socket_is_closed()

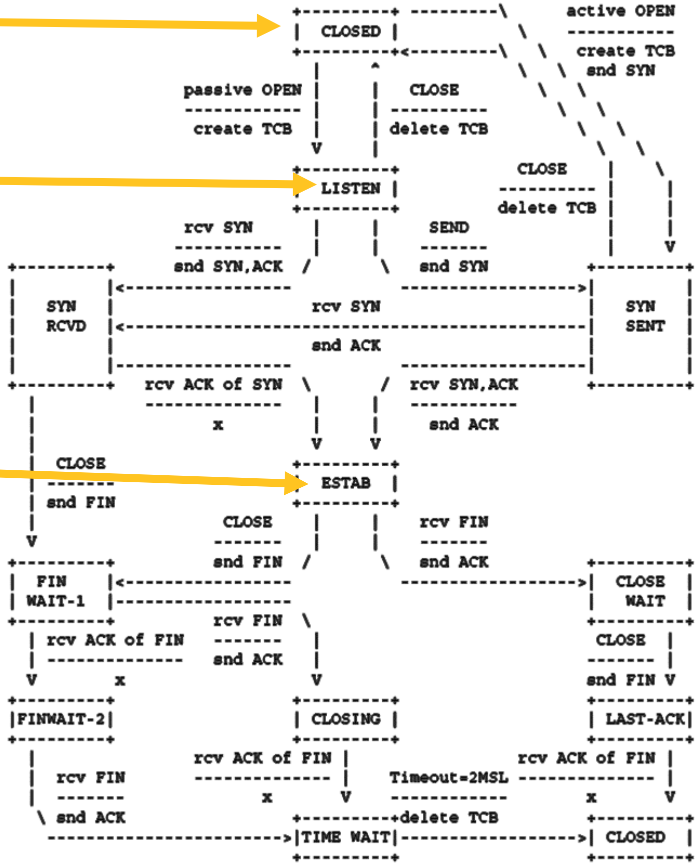
The socket is in the “CLOSED” state

socket_is_listening()

The socket is in the “LISTEN” state.

socket_is_established()

The socket is in the “ESTAB” state.



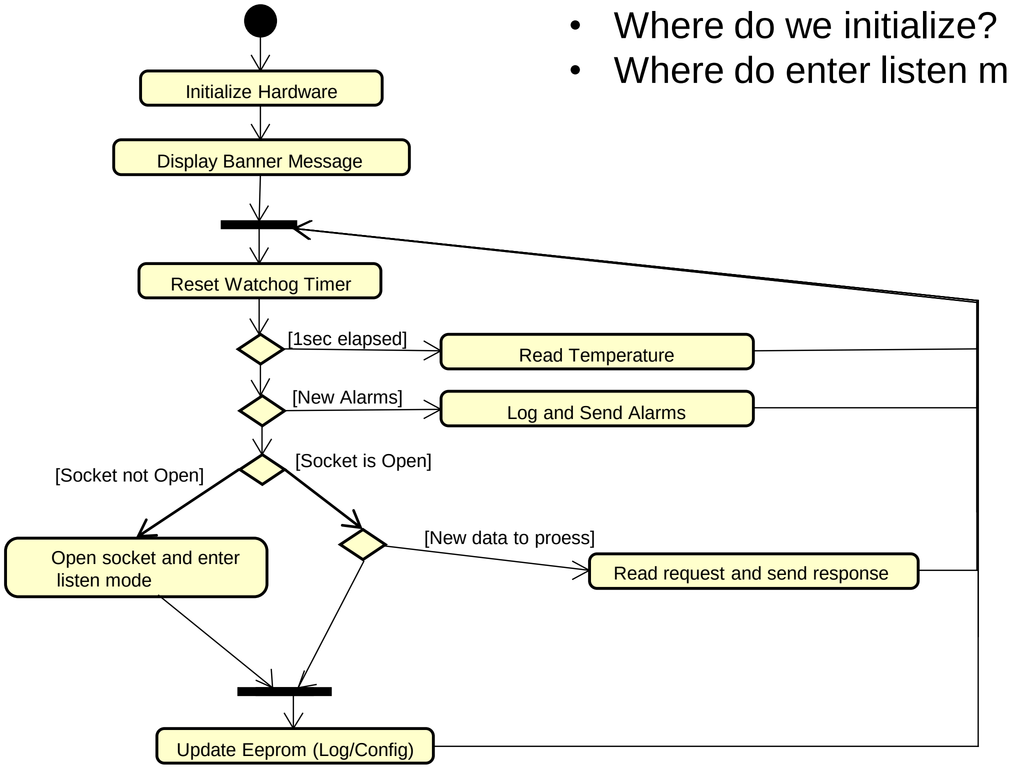
```
                              +---------+ ---------\      active OPEN
                              |  CLOSED |            \    -----------
                              +---------+<---------\   \   create TCB
                                |     ^              \   \  snd SYN
                   passive OPEN |     |   CLOSE        \   \
                   ------------ |     | ----------       \   \
                    create TCB  |     | delete TCB         \   \
                                V     |                      \   \
                              +---------+            CLOSE    |    \
                              |  LISTEN |          ---------- |     |
                              +---------+          delete TCB |     |
                   rcv SYN      |     |     SEND              |     |
                  -----------   |     |    -------            |     V
 +---------+      snd SYN,ACK  /       \   snd SYN          +---------+
 |         |<-----------------           ------------------>|         |
 |   SYN   |                    rcv SYN                     |   SYN   |
 |   RCVD  |<-----------------------------------------------|   SENT  |
 |         |                    snd ACK                     |         |
 |         |------------------           -------------------|         |
 +---------+   rcv ACK of SYN  \       /  rcv SYN,ACK       +---------+
   |           --------------   |     |   -----------
   |                  x         |     |     snd ACK
   |                            V     V
   |  CLOSE                   +---------+
   | -------                  |  ESTAB  |
   | snd FIN                  +---------+
   |                   CLOSE    |     |    rcv FIN
   V                  -------   |     |    -------
 +---------+          snd FIN  /       \   snd ACK          +---------+
 |  FIN    |<-----------------           ------------------>|  CLOSE  |
 | WAIT-1  |------------------                              |   WAIT  |
 +---------+          rcv FIN  \                            +---------+
   | rcv ACK of FIN   -------   |                            CLOSE  |
   | --------------   snd ACK   |                           ------- |
   V        x                   V                           snd FIN V
 +---------+                  +---------+                   +---------+
 |FINWAIT-2|                  | CLOSING |                   | LAST-ACK|
 +---------+                  +---------+                   +---------+
   |                rcv ACK of FIN |                 rcv ACK of FIN |
   |  rcv FIN       -------------- |    Timeout=2MSL -------------- |
   |  -------              x       V    ------------        x       V
    \ snd ACK                 +---------+delete TCB         +---------+
     ------------------------>|TIME WAIT|------------------>| CLOSED  |
                              +---------+                   +---------+
```

From RFC793

Pseudo Code for Socket Connection

```
WHILE FOREVER
    IF  socket_is_closed()
         socket_open();
         socket_listen();
    ELSE
        ## process inputs if there are any
       IF DONE
            socket_disconnect()
       ENDIF
    ENDIF
WEND
```

TCP Connection Server Connection Flow



- Where do we initialize?
- Where do enter listen mode?

We will need to make sure that our packet reception is non-blocking

[END]

```
/* Brainfuck.c
 * BrainFuck Interpreter */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int main(int argc, char **argv)
{
    int memory[32768], space[32768];
    int fd, len, buf, spointer, mpointer;
    int kl = 0;

    if (argc <= 1)
    {
        printf("Usage: %s <brainfuck file>\n", argv[0]);
        exit(1);
    }

    /* Open brainfuck file */
    if ((fd = open(argv[1], O_RDONLY)) < 0) exit(1);

    /* Read the file byte by byte */
    len = spointer = 1;
    while (len > 0)
    {
        len = read(fd, &buf, 1);
        space[spointer] = buf;
        spointer++;
    }

    close(fd);

    for (mpointer = 0; mpointer < 32768; mpointer++) memory[mpointer] = 0;

    len = spointer;
    spointer = mpointer = 0;

    for (spointer = 0; spointer < len; spointer++)
    {
        switch(space[spointer])
        {
            // Increment pointer value
            case '+':
                memory[mpointer]++;
                break;
            // Decrement pointer value
            case '-':
                memory[mpointer]--;
                break;
            // Increment pointer
            case '>':
                mpointer++;
                break;
            // Decrement pointer
            case '<':
                mpointer--;
                break;
            // Print current pointer value
            case '.':
                putchar(memory[mpointer]);
                break;
            // Read value and store in current pointer
            case ',':
                memory[mpointer] = getchar();
                break;
            // Start loop
            case '[':
                if (memory[mpointer] == 0)
                {
                    /* Find matching ] */
                    spointer++;
                    while (kl > 0 || space[spointer] != ']')
                    {
                        if (space[spointer] == '[') kl++;
                        if (space[spointer] == ']') kl--;
                        // Go in right direction
                        spointer++;
                    }
                }
                break;
        // End Loop
            case ']':
                if (memory[mpointer] != 0)
                {
                /* Find matching [ */
                    spointer--;
                    while (kl > 0 || space[spointer] != '[')
                    {
                        if (space[spointer] == '[') kl--;
                        if (space[spointer] == ']') kl++;
                    // Go left
                    spointer--;
                    }
                spointer--;
                }
                break;
        }
    }
    putchar('\n');
    return (0);
}
```

SER486: Embedded C Programming

Networking: HTTP

Arizona State University

Topics to Cover:

What is HTTP?

1

How to Create HTTP requests

2

Parsing HTTP

3

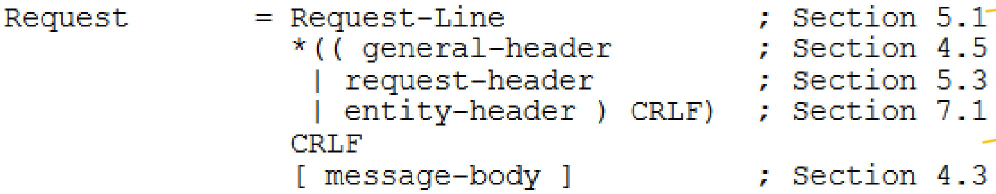
What is HTTP?

OUR Message format: HTTP

HTTP is defined by IETF RFC 2616

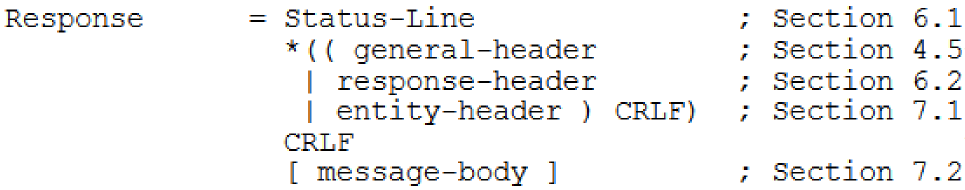
- **The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems.**
- **It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers.**
- **A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred.**

HTTP Format



```
Request       = Request-Line              ; Section 5.1
                *(( general-header        ; Section 4.5
                 | request-header         ; Section 5.3
                 | entity-header ) CRLF)  ; Section 7.1
                CRLF
                [ message-body ]          ; Section 4.3
```

GET 192.168.1.100/device HTTP/1.1
Accept-Language: en-us
…
<CR><LF>

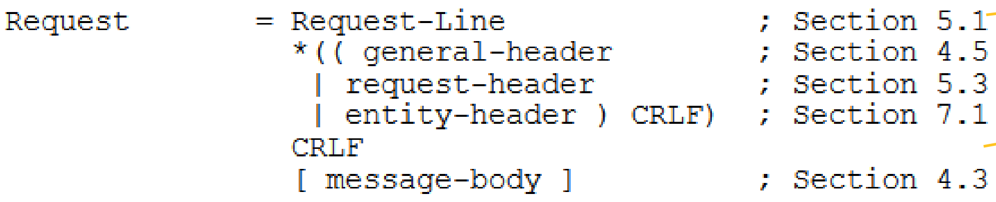


```
Response      = Status-Line               ; Section 6.1
                *(( general-header        ; Section 4.5
                 | response-header        ; Section 6.2
                 | entity-header ) CRLF)  ; Section 7.1
                CRLF
                [ message-body ]          ; Section 7.2
```

HTTP/1.1 200 OK
Connection: Closed
…
<CR><LF>
<html>
<body>
/device
&nbsp &nbsp &nbsp &nbsp &nbsp/vpd
…
</body>
</html>
<CR><LF>

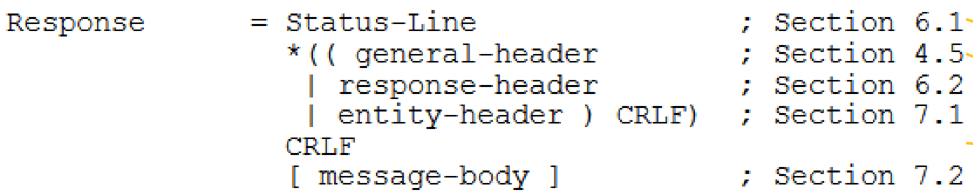
Contents of this slide included from IETF RFC 2616

HTTP Format Continued



```
Request       = Request-Line              ; Section 5.1
                *(( general-header        ; Section 4.5
                 | request-header         ; Section 5.3
                 | entity-header ) CRLF)  ; Section 7.1
                CRLF
                [ message-body ]          ; Section 4.3
```

PUT 192.168.1.100/device/config/tcrit_hi HTTP/1.1
Accept-Language: en-us
…
<html>
<body>
100
</body>
</html>



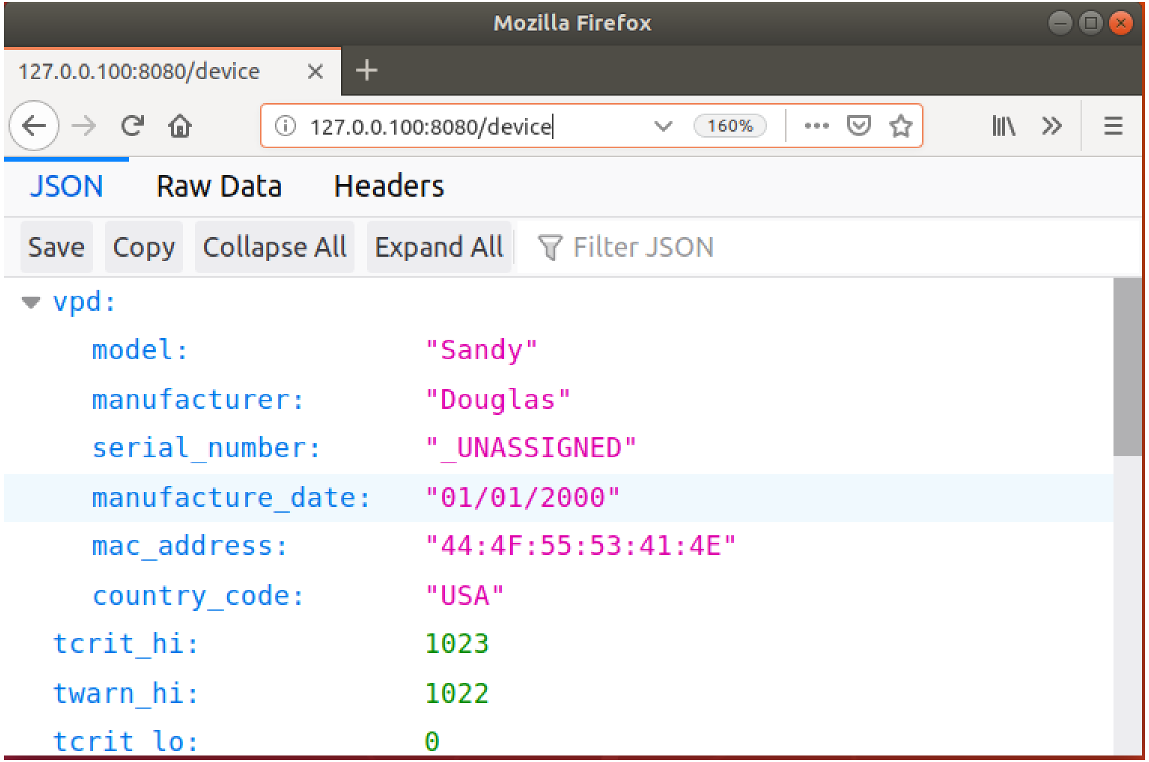
```
Response      = Status-Line               ; Section 6.1
                *(( general-header        ; Section 4.5
                 | response-header        ; Section 6.2
                 | entity-header ) CRLF)  ; Section 7.1
                CRLF
                [ message-body ]          ; Section 7.2
```

HTTP/1.1 200 OK
Connection: Closed
<CR><LF>

Contents of this slide included from IETF RFC 2616

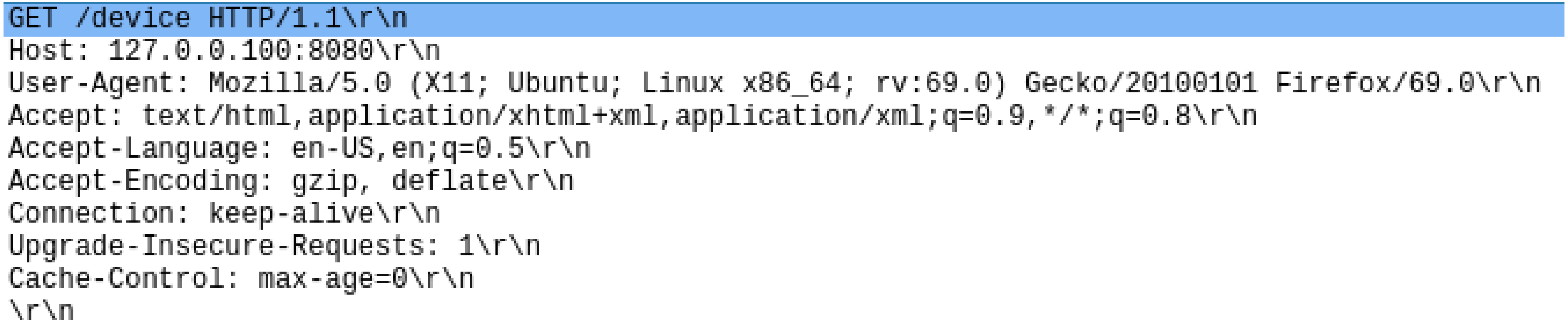
Sending HTTP Requests

Sending an HTTP GET request from your browser



```
▾ vpd:
    model:              "Sandy"
    manufacturer:       "Douglas"
    serial_number:      "_UNASSIGNED"
    manufacture_date:   "01/01/2000"
    mac_address:        "44:4F:55:53:41:4E"
    country_code:       "USA"
  tcrit_hi:             1023
  twarn_hi:             1022
  tcrit_lo:             0
```

What happened – GET Request



```
GET /device HTTP/1.1\r\n
Host: 127.0.0.100:8080\r\n
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:69.0) Gecko/20100101 Firefox/69.0\r\n
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
Accept-Language: en-US,en;q=0.5\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: keep-alive\r\n
Upgrade-Insecure-Requests: 1\r\n
Cache-Control: max-age=0\r\n
\r\n
```

What happened – GET Response

```
HTTP/1.1 200 OK
Content-Type: application/vnd.api + json
…
<CR><LF>
{"vpd":{"model":"Sandy","manufacturer":"Douglas","serial_number":"_UNASSIGNED","manufacture_date":
"01/01/2000","mac_address":"44:4F:55:53:41:4E","country_code":"USA"},"tcrit_hi":1023,"twarn_hi":1022,"t
crit_lo":0,"twarn_lo":1,"temperature":75,"state":"NORMAL","log":[{"timestamp":"01/01/2000
00:00:00","event":3},{"timestamp":"01/01/2000 00:00:00","event":4},{"timestamp":"01/01/2000
00:00:00","event":0},{"timestamp":"01/01/2000 00:00:07","event":2},{"timestamp":"01/01/2000
00:00:01","event":3},{"timestamp":"01/01/2000 00:00:00","event":4},{"timestamp":"01/01/2000
00:00:00","event":0}]}
<CR><LF>
```

Sending a GET request with CURL

```
curl –X GET ‘http://127.0.0.100:8080/device’
```

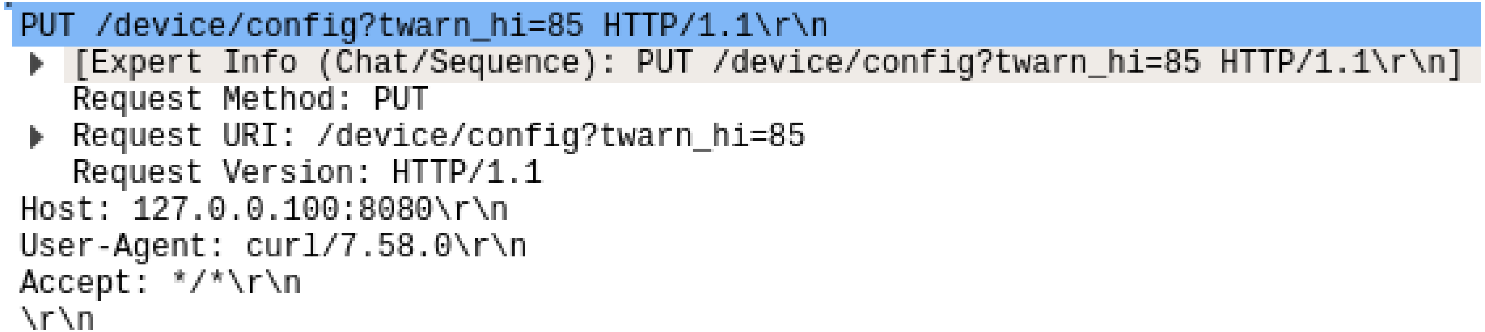
(type this into the terminal window – don’t copy-paste)

```
{"vpd":{"model":"Sandy","manufacturer":"Douglas","serial_number":"_UNASSIGNED","manufacture_date":"01/01/2000","mac_address":"44:4F:55:53:41:4E","country_code":"USA"},"tcrit_hi":1023,"twarn_hi":1022,"tcrit_lo":0,"twarn_lo":1,"temperature":75,"state":"NORMAL","log":[{"timestamp":"01/01/2000 00:00:00","event":3},{"timestamp":"01/01/2000 00:00:00","event":4},{"timestamp":"01/01/2000 00:00:00","event":0},{"timestamp":"01/01/2000 00:00:07","event":2},{"timestamp":"01/01/2000 00:00:01","event":3},{"timestamp":"01/01/2000 00:00:00","event":4},{"timestamp":"01/01/2000 00:00:00","event":0}]}
```

Sending a PUT Request

curl -X PUT ‘http://127.0.0.100:8080/device/config?twarn_hi=85’

(type this into the terminal window – don’t copy-paste)



```
PUT /device/config?twarn_hi=85 HTTP/1.1\r\n
 ▶ [Expert Info (Chat/Sequence): PUT /device/config?twarn_hi=85 HTTP/1.1\r\n]
   Request Method: PUT
 ▶ Request URI: /device/config?twarn_hi=85
   Request Version: HTTP/1.1
Host: 127.0.0.100:8080\r\n
User-Agent: curl/7.58.0\r\n
Accept: */*\r\n
\r\n
```

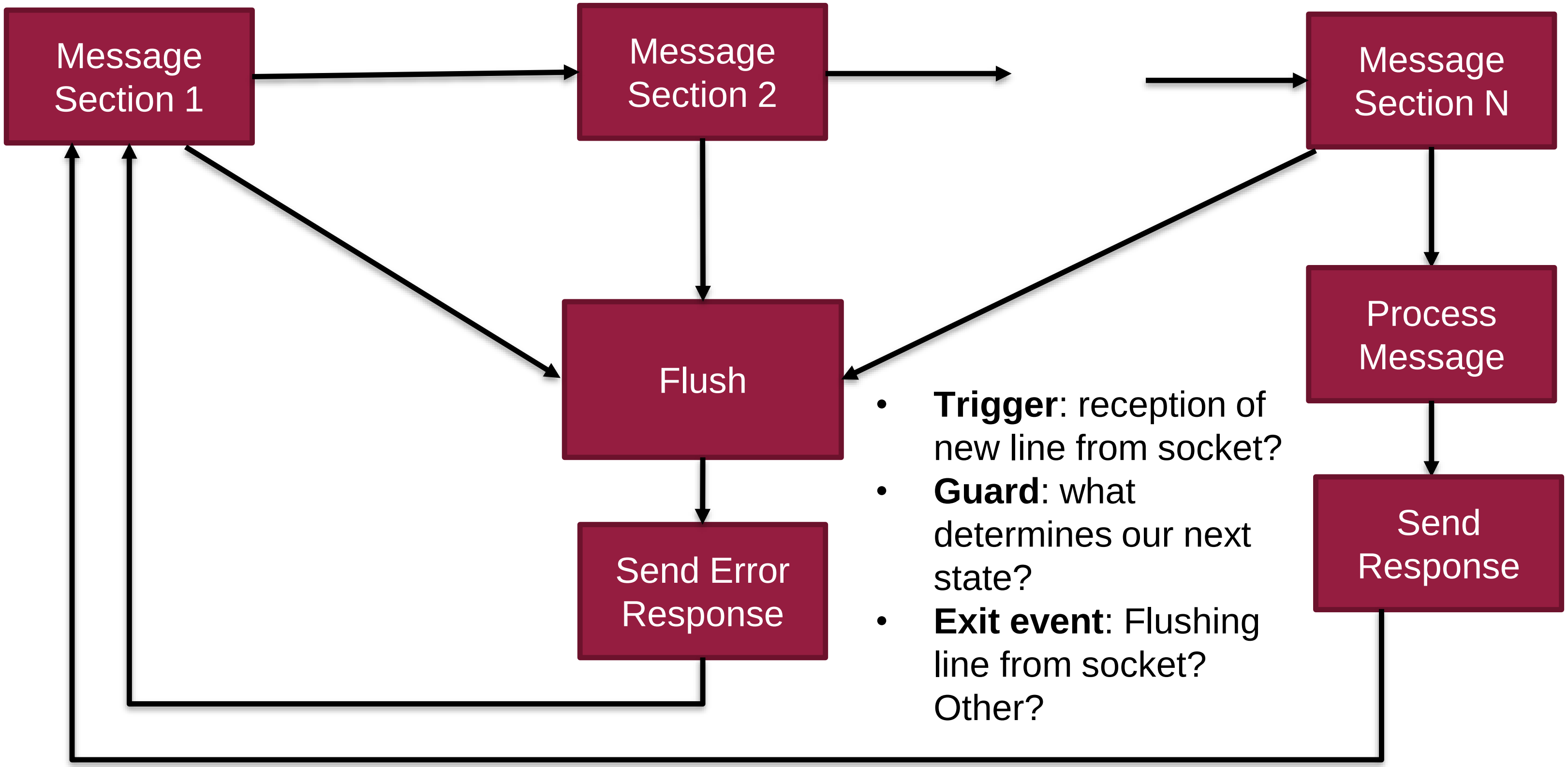
Sending a DELETE Request with CURL

```
curl -X DELETE ‘http://127.0.0.100:8080/device/log’
```

(type this into the terminal window – don’t copy-paste)

Parsing HTTP

Generic Text-based Protocol Parsing FSM



- **Trigger**: reception of new line from socket?
- **Guard**: what determines our next state?
- **Exit event**: Flushing line from socket? Other?

Observations

HTTP can be processed one line at a time

- Don’t block other processes while waiting for a line of text
- Can use
- Delimiter = CRLF

HTTP line interpretation based on context

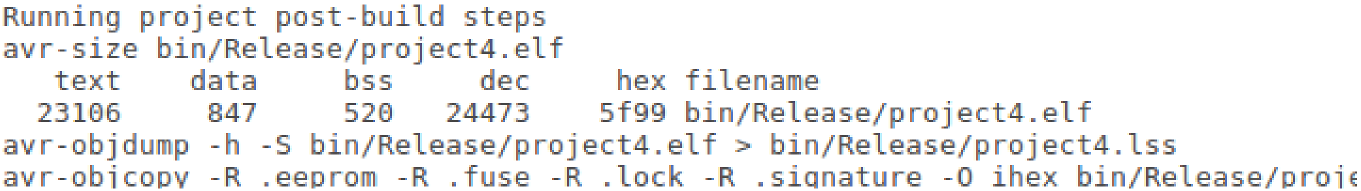
- First line is the request line
- Optional body lines will be included after the header concluding CRLF

JSON body should be expected for PUT requests but not for GET or DELETE

Responses must be sent AFTER fully receiving the HTTP message (and optional body)

- Was the URI correct?
- Was the command well-formed?
- Was the PUT value of the expected type and within range?

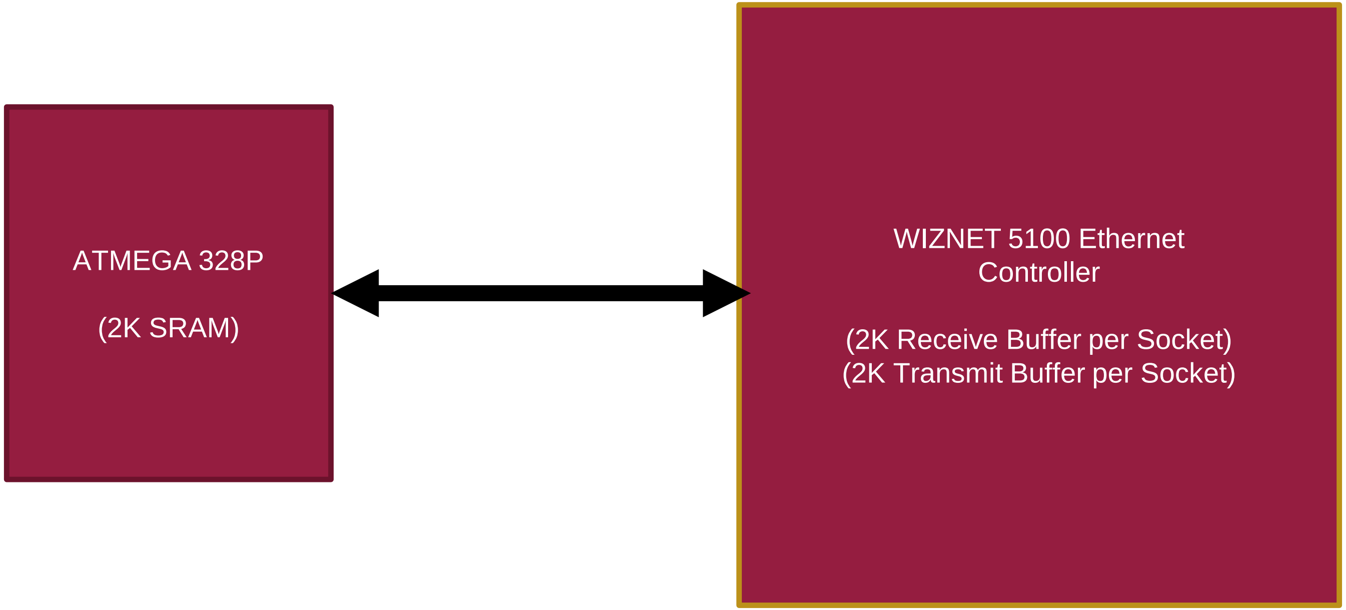
The problem of Limited Memory



```
Running project post-build steps
avr-size bin/Release/project4.elf
   text    data     bss     dec     hex filename
  23106     847     520   24473    5f99 bin/Release/project4.elf
avr-objdump -h -S bin/Release/project4.elf > bin/Release/project4.lss
avr-objcopy -R .eeprom -R .fuse -R .lock -R .signature -O ihex bin/Release/proje
```

How does this change our parser?

Fortunately There is More Memory Elsewhere


ATMEGA 328P
(2K SRAM)
WIZNET 5100 Ethernet Controller
(2K Receive Buffer per Socket)
(2K Transmit Buffer per Socket)

To make of the added memory, we must perform string operations on the data in the Ethernet socket rather than copying into the ATMEGA

A very useful function

```
/* Compare the first bytes of the receive buffer with the specified string.
* If they match, the bytes are removed from the buffer and the function returns
* a value of 1.  Otherwise, the contents of the receive buffer are not altered
* and the function returns 0.
*/
unsigned char socket_recv_compare(SOCKET s, const char*str);
```

Our IIoT Endpoints

Endpoint	Description	Operations Supported
\device	A resource that represents the entire IIoT device	GET, PUT
\device\config	A resource that represents the IIoT device configuration	PUT
\device\log	A resource that represents the IIoT device log	DELETE

```c
/* Brainfuck.c
 * BrainFuck Interpreter */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int main(int argc, char **argv)
{
    int memory[32768], space[32768];
    int fd, len, buf, spointer, mpointer;
    int kl = 0;

    if (argc <= 1)
    {
        printf("Usage: %s <brainfuck file>\n", argv[0]);
        exit(1);
    }

    /* Open brainfuck file */
    if ((fd = open(argv[1], O_RDONLY)) < 0) exit(1);

    /* Read the file byte by byte */
    len = spointer = 1;
    while (len > 0)
    {
        len = read(fd, &buf, 1);
        space[spointer] = buf;
        spointer++;
    }

    close(fd);

    for (mpointer = 0; mpointer < 32768; mpointer++) memory[mpointer] = 0;

    len = spointer;
    spointer = mpointer = 0;

    for (spointer = 0; spointer < len; spointer++)
    {
        switch(space[spointer])
        {
            // Increment pointer value
            case '+':
                memory[mpointer]++;
                break;
            // Decrement pointer value
            case '-':
                memory[mpointer]--;
                break;
            // Increment pointer
            case '>':
                mpointer++;
                break;
            // Decrement pointer
            case '<':
                mpointer--;
                break;
            // Print current pointer value
            case '.':
                putchar(memory[mpointer]);
                break;
            // Read value and store in current pointer
            case ',':
                memory[mpointer] = getchar();
                break;
            // Start loop
            case '[':
                if (memory[mpointer] == 0)
                {
                    /* Find matching ] */
                    spointer++;
                    while (kl > 0 || space[spointer] != ']')
                    {
                        if (space[spointer] == '[') kl++;
                        if (space[spointer] == ']') kl--;
                        // Go in right direction
                        spointer++;
                    }
                }
            break;
        // End Loop
            case ']':
                if (memory[mpointer] != 0)
                {
                /* Find matching [ */
                    spointer--;
                    while (kl > 0 || space[spointer] != '[')
                    {
                        if (space[spointer] == '[') kl--;
                        if (space[spointer] == ']') kl++;
                    // Go left
                    spointer--;
                    }
                spointer--;
            }
            break;
        }
    }
    putchar('\n');
    return (0);
}
```

[END]

SER486: Embedded C Programming

Networking: JSON and Our IIoT Data Model

Arizona State University

```
“Name”: “CNC Machine 1”
“Asset#”: 5551234
“Actuators”: [
               {    “Name”: “Motor1”, ... }
               {    “Name”: “Motor2”, ... }
               {    “Name”: “Motor3”, ... }
],
“Sensors”: [
{
               “ID”: 1
“Type”: “Temperature”,
“Model#”: “PICMG 125”,
“Manufacturer”: “PICMG”,
               “MaximumValue”: 150,
               “MinimumValue”: 0,
               “Units”: “Degrees F”,
               “Reading”: 105.8
} ,
...
],
“Telemetry”: [
} ...
“SensorID”: 1
"Status": {
"State": "Enabled",
"Health": "OK"
               } ,
"ReadingCelsius": 41,
"UpperThresholdCritical": 45,
"PhysicalContext": “Motor1 Assy"
}, ...
]
```

JSON

Characteristics

- “Lightweight” data exchange format
- Commonly used for IT data modeling
- Human readable
- Hierarchical

Key/Value pairs

- **Formatted as “key”:value**

Value types:

- Integer
- String
- True/False
- Array
- Object

Arrays

- Enclosed in []
- Contain values separated by commas

Objects

- Enclosed in {}
- Contain Key/Value Pairs separated by commas

More information available in IETF RFC 8259

Our Device Object Model

Key	Value Type	Example
“vpd”	VPD object	{“model”:“XYZ-2000”, “manufacturer”:“Dougtronic”, “serial_number”:“asurite”, “manufacture_date”:“01/01/2019”, “mac_address”:“44:4F:55:53:41:4E”, “country_code”:“USA”}
“tcrit_hi”	Integer	1023
“twarn_hi”	Integer	1022
“tcrit_lo”	Integer	0
“twarn_lo”	Integer	1
“temperature”	Integer	88
“state”		“NORMAL”
“log”	LogEntry[]	[{“timestamp”:"04/01/2018 00:01:01",“event”:1}, {“timestamp”:"01/01/2018 00:01:00",“event”:0}]

VPD and LogEntry Data Models

VPD Object

Key	Value Type	Example
“model”	String	“XYZ-2000”
“manufacturer”	String	“Dougtronic”
“serial_number”	String	“asurite”
“manufacture_date”	String	“01/01/2019”
“mac_address”	String	“44:4F:55:53:41:4E”
“country_code”	String	“USA”

LogEntry Object

Key	Value Type	Example
“timestamp”	String	"01/01/2000 00:00:00"
“eventnum”	Integer	0

Example of our data model in JSON

```
{"vpd":{"model":"Sandy","manufacturer":"Douglas","serial_numbe
  r":"_UNASSIGNED","manufacture_date":"01/01/2000","mac_addres
  s":"44:4F:55:53:41:4E","country_code":"USA"},"tcrit_hi":1023
  ,"twarn_hi":1022,"tcrit_lo":0,"twarn_lo":1,"temperature":75,
  "state":"NORMAL","log":[{"timestamp":"01/01/2000
  00:00:00","event":3},{"timestamp":"01/01/2000
  00:00:00","event":4},{"timestamp":"01/01/2000
  00:00:00","event":0},{"timestamp":"01/01/2000
  00:00:07","event":2},{"timestamp":"01/01/2000
  00:00:01","event":3},{"timestamp":"01/01/2000
  00:00:00","event":4},{"timestamp":"01/01/2000
  00:00:00","event":0}]}
```

?

For the final project you will need to respond to an HTTP get with a JSON-formatted representation of the device state.

How can this be done?

More useful socket functions

```
/* send an character to the remote host (not including the terminating null) */
void        socket_writechar(SOCKET s, const char ch);

/* send an ascii string to the remote host (not including the terminating null) */
void        socket_writestr(SOCKET s, const char*str);

/* send the specified ascii string to the remote host, enclosed in double quote characters */
void        socket_writequotedstring(SOCKET s, const char*str);

/* send the specified 8-bit integer to the remote host as a hexadecimal, text representation */
void        socket_writehex8(SOCKET s, const unsigned char x);

/* send the specified 16-bit integer to the remote host as hexadecimal, text representation */
void        socket_writehex16(SOCKET s, const unsigned int x);

/* send the specified integer to the remote host as a decimal text representation */
void        socket_writedec32(SOCKET s, int n);

/* given a RTC date/time number, send a text representation of the date to the remote host */
void        socket_writedate(SOCKET s, unsigned long datenum);

/* given a pointer to an array of 6 unsigned characters representing a mac address, send
* the text representation consisting of 6 8-bit hexadecimal numbers, separated by colons */
void        socket_write_macaddress(SOCKET s, unsigned char *mac_address);
```

```
/* Brainfuck.c
 * BrainFuck Interpreter */

#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>

int main(int argc, char **argv)
{
    int memory[32768], space[32768];
    int fd, len, buf, spointer, mpointer;
    int kl = 0;

    if (argc <= 1)
    {
        printf("Usage: %s <brainfuck file>\n", argv[0]);
        exit(1);
    }

    /* Open brainfuck file */
    if ((fd = open(argv[1], O_RDONLY)) < 0) exit(1);

    /* Read the file byte by byte */
    len = spointer = 1;
    while (len > 0)
    {
        len = read(fd, &buf, 1);
        space[spointer] = buf;
        spointer++;
    }

    close(fd);

    for (mpointer = 0; mpointer < 32768; mpointer++) memory[mpointer] = 0;

    len = spointer;
    spointer = mpointer = 0;

    for (spointer = 0; spointer < len; spointer++)
    {
        switch(space[spointer])
        {
            // Increment pointer value
            case '+':
                memory[mpointer]++;
                break;
            // Decrement pointer value
            case '-':
                memory[mpointer]--;
                break;
            // Increment pointer
            case '>':
                mpointer++;
                break;
            // Decrement pointer
            case '<':
                mpointer--;
                break;
            // Print current pointer value
            case '.':
                putchar(memory[mpointer]);
                break;
            // Read value and store in current pointer
            case ',':
                memory[mpointer] = getchar();
                break;
            // Start loop
            case '[':
                if (memory[mpointer] == 0)
                {
                    /* Find matching ] */
                    spointer++;
                    while (kl > 0 || space[spointer] != ']')
                    {
                        if (space[spointer] == '[') kl++;
                        if (space[spointer] == ']') kl--;
                        // Go in right direction
                        spointer++;
                    }
                }
            break;
        // End Loop
            case ']':
                if (memory[mpointer] != 0)
                {
                /* Find matching [ */
                    spointer--;
                    while (kl > 0 || space[spointer] != '[')
                    {
                        if (space[spointer] == '[') kl--;
                        if (space[spointer] == ']') kl++;
                    // Go left
                    spointer--;
                    }
                spointer--;
            }
            break;
        }
    }
    putchar('\n');
    return (0);
}
```

[END]