

TNE20003 - Internet and Cybersecurity for Engineering Applications

Internet-Enabled Programming

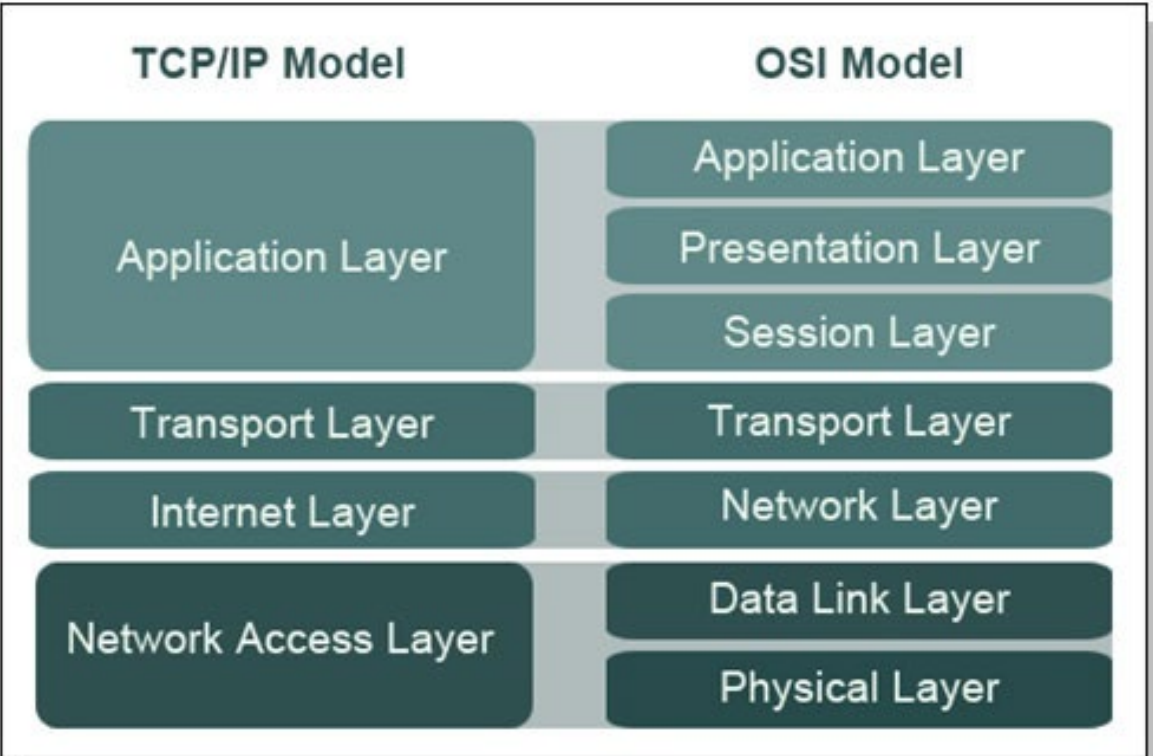


7.1 Transport Layer Protocols

The Network Stack

While we don't use the OSI Model in Practice, it is still referred to when numbering layers (hence Routers are called Layer 3 devices because they operate at the Internet Layer)

In this weeks classes we will examine the basic principles for putting together applications (at the Application Layer) to make use of the existing TCP/IP Stack



TCP – Transmission Control Protocol

TCP is the dominant Transport Layer Protocol running on the Internet today:

- Data is delivered in-order to the application
- **Guaranteed error-free delivery**
- Throughput adjusts based on available bandwidth

TCP is Stream Based

UDP – User Datagram Protocol

UDP is the second most dominant Transport Layer Protocol running on the Internet today:

- Extension of Best Effort Packet Delivery to applications
- **Error Detection only**
- Datagrams queued and delivered ASAP

UDP is Message Based

Network Types

How to Choose



Traditionally TCP is used when we want to transmit a stream of data to the destination and time immediacy is not important



Traditionally UDP is used when time is of the essence and guaranteed delivery less so. UDP is also often used for simple query-response protocols

With Cybersecurity concerns and issues with Firewalls, TCP is becoming the preferred option for most protocols

Other Transport Layer Protocols

Multi-Path TCP – MPTCP

- Support multiple interfaces concurrently, increasing throughput through use of concurrent channels
- Looks like TCP to firewalls so typically allowed

Stream Control Transmission Protocol – SCTP

- Support multiple interfaces, both best effort, guaranteed, and partial guarantee

Datagram Congestion Control Protocol – DCCP

- Unreliable like UDP but with Congestion Control to not overload network

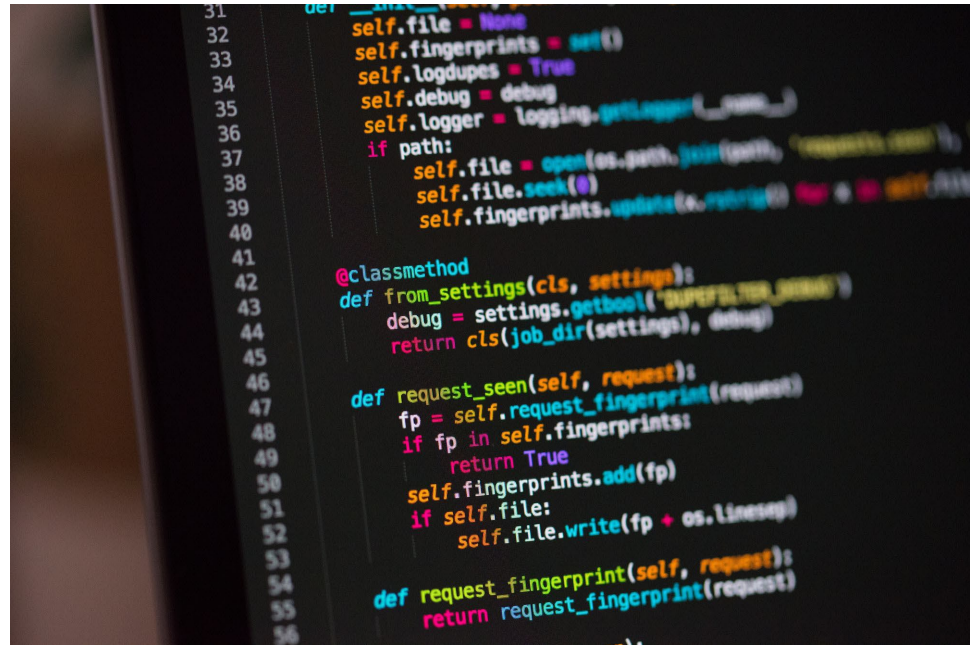
QUIC

- New variant used by Google Chrome to deliver web content. Faster connection establishment

7.2 Network Programming

Accessing the Network from an Application

- The Transport Layer is implemented in the OS
- Need a standard mechanism to access these functions within the OS
- All platforms provide the Sockets API, a C-Library that interfaces with the network stack on the OS
- Standard API written by Berkeley University for Unix. Supported by everyone – even Microsoft
- For other programming languages, they provide a wrapper around the Sockets Library
- Function calls are basically the same for all languages, with same functionality



```
31 def __init__(self):
32     self.file = None
33     self.fingerprints = set()
34     self.logdupes = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'requests.txt'),
39                           'a')
40         self.file.seek(0)
41         self.fingerprints.update(e.request for e in self.requests)
42
43 @classmethod
44 def from_settings(cls, settings):
45     debug = settings.getbool('DEBUG', False)
46     return cls(job_dir(settings), debug)
47
48 def request_seen(self, request):
49     fp = self.request_fingerprint(request)
50     if fp in self.fingerprints:
51         return True
52     self.fingerprints.add(fp)
53     if self.file:
54         self.file.write(fp + os.linesep)
55
56 def request_fingerprint(self, request):
57     return request_fingerprint(request)
```

Blocking vs Non-Blocking Sockets

By default Sockets are Blocking:

- Execution of most functions will block until data can be sent/received, or connections are actually established
- **Your program will stop until the call unblocks**
- Typically need multi-threading to support concurrency

Can Change to Non-Blocking Sockets

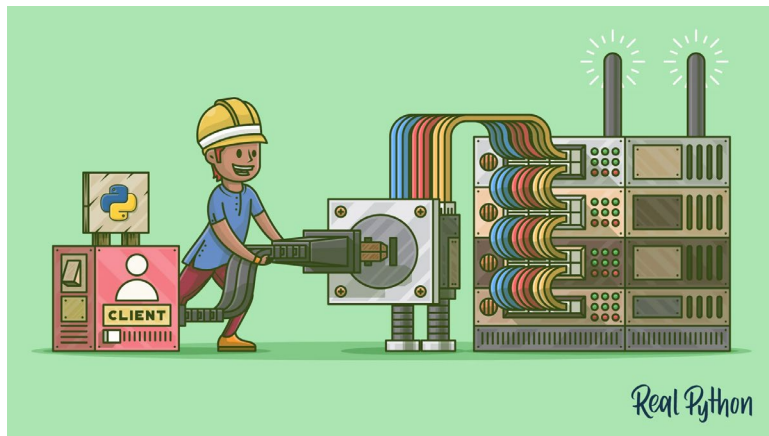
- No Socket function call will block
- **Instead return special error code to signify that it “would block”**
- Need to detect and act on result

Or use select() call:

- select() will block on multiple Sockets and unblock when any one of them needs to be handled

Sample Programs in Week 8 Tutorial

Basic Socket API Functions (1)



socket()

- Create a socket in OS and return a handle to newly created socket
- Handle is used for all future calls on this socket
- Parameters include protocol (eg TCP) and network (eg IPv4)

bind()

- Attach socket to nominated IP address and Port number on host
- Can bind to a wildcard IP (all IP on system)
- Can bind to wildcard port (OS select)



Basic Socket API Functions (2)



listen()

- Tell the OS to start accepting connections on this TCP socket
- Entire TCP connection is handled by the OS
- Program has no part in this other than calling listen()

accept()

- Accept a pending connection on a TCP listening socket
- Call will block until there is a connection pending as complete by the OS
- Function will return a new (second) socket which is used to manage the new connection
- Original socket can be used to accept further connections
- New socket is bound to same local IP/port as listening socket



Basic Socket API Functions (3)



connect()

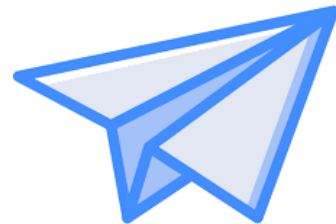
- Attempt to establish a connection to a remote application
- Parameters include remote address and port number
- Will initiate TCP handshake by OS, blocks until connection established
- For UDP remote information is stored, there is no connect process

send()

- Send data over connected socket
- Blocks if no buffer space left

sendto()

- For unconnected UDP sockets to send to nominated remote IP address and Port number



Basic Socket API Functions (4)



recv()

- Retrieve data from connected socket
- Blocks if no data available within OS

recvfrom()

- For unconnected UDP sockets, retrieve UDP datagram **AND** provide information on the remote IP address and port number so you can send a reply

close()

- Closes the socket so no more data can be sent/received
- Releases resources in the OS
- If TCP, OS performs TCP disconnection sequence



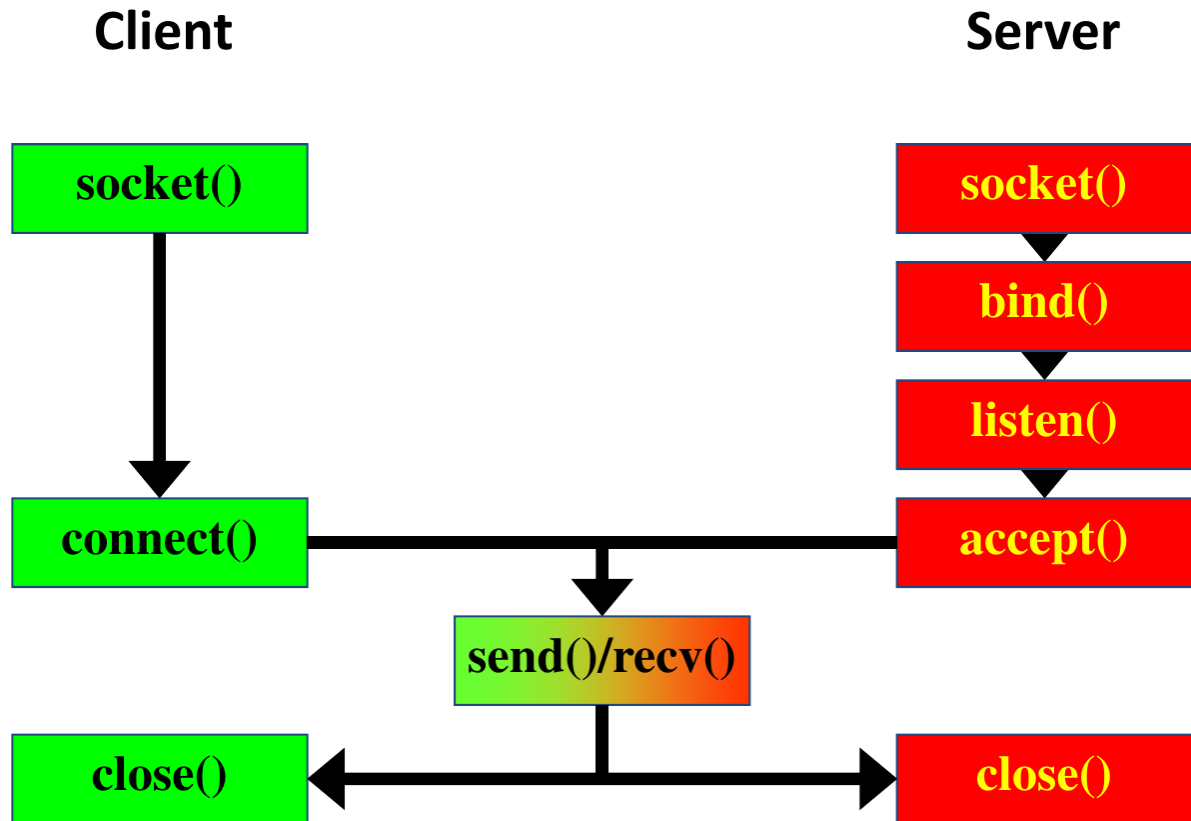
Basic Socket API Functions (5)

- Other functions exist to set (and get) socket options (such as whether a socket is blocking or not)
- **Also functions to perform DNS queries and find an IP address from a name and vice-versa. This is important as you want the user to specify a name (URL) but the socket functions only work with IP addresses**
- Also functions to convert an IP address to a string and vice-versa (for display and/or logging purposes)

TCP Application Flowchart

Sample Socket API process for applications to communicate via TCP

One end must always operate as a server



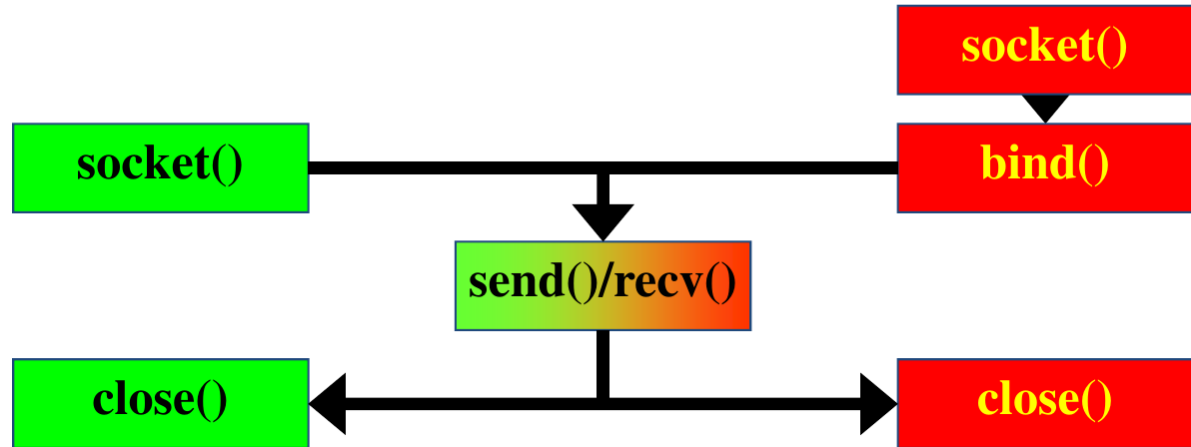
Network Programming

UDP Application Flowchart

Sample Socket API process for applications to communicate via UDP

Client

Server



7.3 Embedded Systems

Embedded Systems

Network Programming

Not all embedded systems are capable of supporting Python
In this case you may need to use the C Sockets Library



Not all embedded systems run a full Operating System
If network support is available, it is typically a socket-like library that you can call directly
Need to be more careful how you code as the OS is not multi-tasking, you will need to ensure the library is called often enough to check for network data

7.4 Laboratory

Lab – Network Programming

In this lab, you will complete the following objectives:

- Write a TCP Python Program to connect to a web server
- Construct a HTTP request for a web page
- Retrieve the downloaded content and display

Credit Task:

- Parse the returned HTTP data and separate the page from the metadata for display

Distinction Task:

- Parse the downloaded HTML for embedded images
- Download and save the images to disk

SWIN
BUR
* NE *

SWINBURNE
UNIVERSITY OF
TECHNOLOGY



CISCO

What Did I Learn in this Module?

- The reasons for selecting different Transport Layer Protocol depending on your required application
- The standard interface for applications to communicate with the OS to establish and use network connections
- The basic functions provided for network communications
- The order and process for calling these functions

Module 7 – New Terms and Commands

- Sockets
- Sockets API
- Application Process