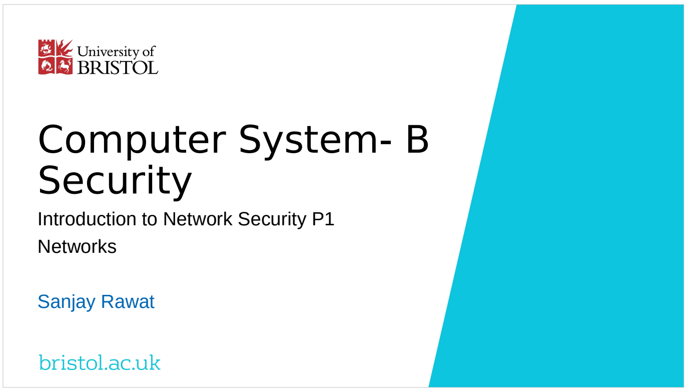


University of BRISTOL

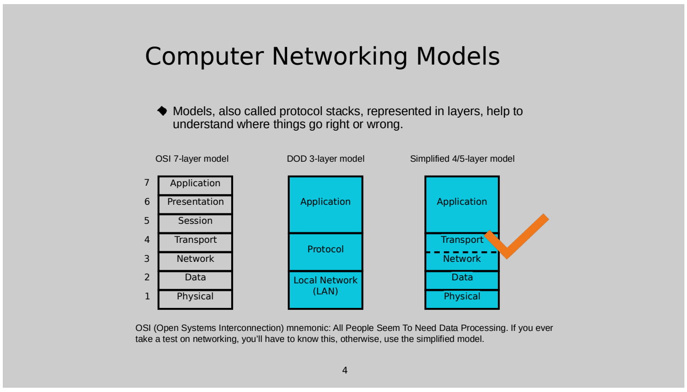
Computer System- B Security

Introduction to Network Security P1

Networks

Sanjay Rawat

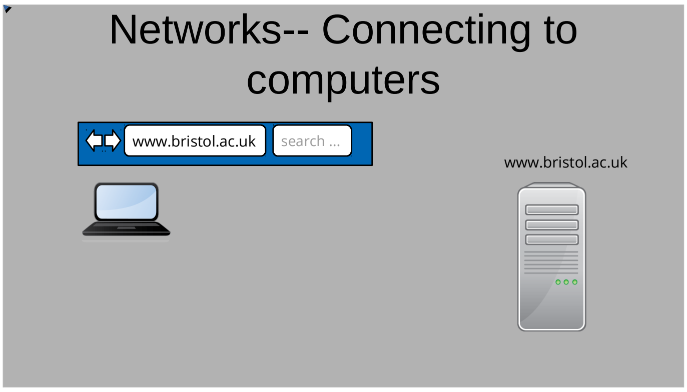
bristol.ac.uk



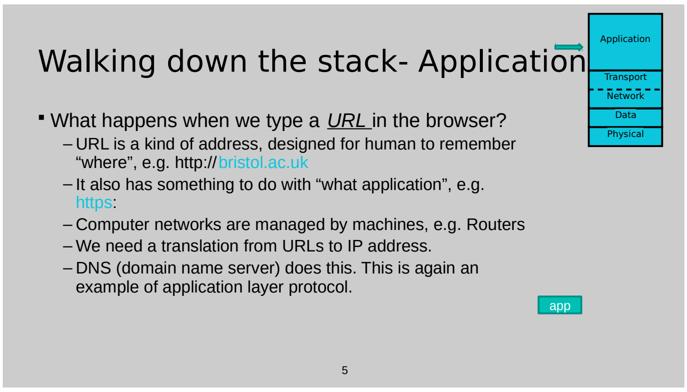
Computer Networking Models

- Models, also called protocol stacks, represented in layers, help to understand where things go right or wrong.

OSI (Open Systems Interconnection) mnemonic: All People Seem To Need Data Processing. If you ever take a test on networking, you'll have to know this, otherwise, use the simplified model.

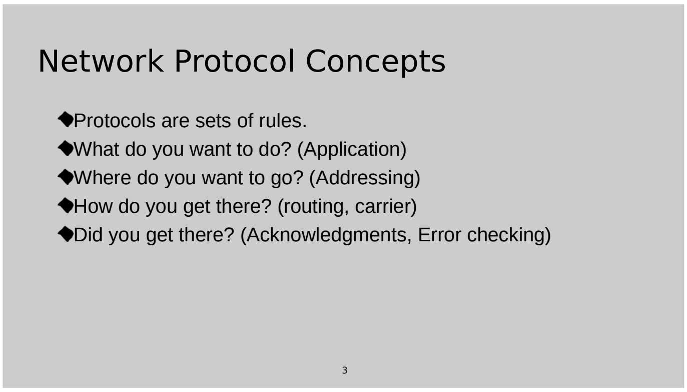


Networks-- Connecting to computers



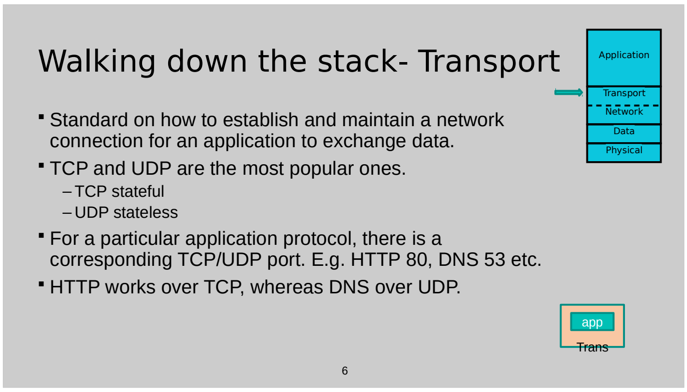
Walking down the stack- Application

- What happens when we type a *URL* in the browser?
  - URL is a kind of address, designed for human to remember "where", e.g. http://bristol.ac.uk
  - It also has something to do with "what application", e.g. https:
  - Computer networks are managed by machines, e.g. Routers
  - We need a translation from URLs to IP address.
  - DNS (domain name server) does this. This is again an example of application layer protocol.



Network Protocol Concepts

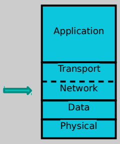
- Protocols are sets of rules.
- What do you want to do? (Application)
- Where do you want to go? (Addressing)
- How do you get there? (routing, carrier)
- Did you get there? (Acknowledgments, Error checking)



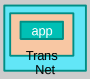
Walking down the stack- Transport

- Standard on how to establish and maintain a network connection for an application to exchange data.
- TCP and UDP are the most popular ones.
  - TCP stateful
  - UDP stateless
- For a particular application protocol, there is a corresponding TCP/UDP port. E.g. HTTP 80, DNS 53 etc.
- HTTP works over TCP, whereas DNS over UDP.

Walking down the stack- Network



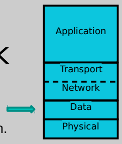
- Standard for how to reach to a connected machine (via intermediate routers).
- IP v4 and v6 address schemes.
- Other examples: ICMP (ping), IGMP etc.
- Remember, we use DNS to get this IP address of Bristol.ac.uk, which is 137.222.0.38.



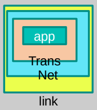
IP Addressing

- IPv4 addresses consists of 4 “octets” such as: 172.16.1.20 (32-bit long)
- Each “octet” consists of numbers between 0 and 255.
- It works sort of like the phone system, with “area codes” to the left, then “prefix” etc. but more flexible. On campus, your computer will know that “172.16.” means “BuildingX” while it will figure out that “1” means “Floor1” and will learn that “20” means the computer called “ABC.” It does this via subnet masking (in this case, 255.255.255.0), which isn’t covered in this class.
- We have a range, used for private networks.
  - 10.0.0.0 – 10.255.255.255
  - 172.16.0.0 – 172.31.255.255
  - 192.168.0.0 – 192.168.255.255
- IPv4 addresses are getting consumed. So, we have IPv6.
  - 128 bit long
  - eight groups of four hexadecimal digits, each group representing 16 bits. E.g. 2001:0db8:85a3:0000:0000:8a2e:0370:7334

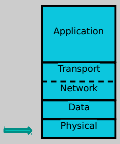
Walking down the stack- Data/link



- Standard on “how two physical devices (i.e. computers) connect and share data.
- Some form of addressing scheme is required to get the packet to the right destination.
  - This is called the Media Access Control (or MAC) address, or sometimes ethernet address, physical address, adaptor address, hardware address, etc.
  - It’s a 12-digit (48 bit) hexadecimal address that is unique to that ethernet adaptor **(but can be changed!)**. Ex. 00:30:65:83:fc:0a.
  - The first three octets of any MAC address are IEEE-assigned Organizationally Unique Identifiers
  - E.g., Cisco 00-1A-A1, D-Link 00-1B-11, ASUSTek 00-1A-92
  - The next three can be assigned by organizations as they please, with uniqueness being the only constraint
  - ARP (address resolution protocol) is used to get MAC, given an IP.
- Switches/hubs operates at this layer



Walking down the stack- Physical



- Nowadays: Pretty much just Cat 5 (or Cat 5e or Cat6) twisted pair copper wire and microwave (wireless).
- Other: Fiber (multi-mode or single-mode) coaxial copper (thick- and thin-net), Cable Modem, plain phone (DSL), microwaves (wireless ethernet), etc.

Transmission Control Protocol

- IP is stateless protocol (each packet is independent of others).
- TCP is a transport layer protocol guaranteeing reliable data transfer, in-order delivery of messages and the ability to distinguish data for multiple concurrent applications on the same host
- Most popular application protocols, including HTTP, FTP and SSH are built on top of TCP
- TCP takes a stream of 8-bit byte data, packages it into appropriately sized segment and calls on IP to transmit these packets
- Delivery order is maintained by marking each packet with a sequence number
- Every time TCP receives a packet, it sends out an ACK to indicate successful receipt of the packet.
- TCP generally checks data transmitted by comparing a checksum of the data with a checksum encoded in the packet

Ports

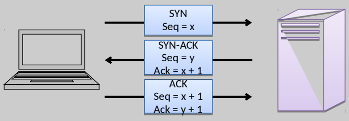
- TCP supports multiple concurrent applications on the same server
- Accomplishes this by having ports, 16 bit numbers identifying where data is directed
- The TCP header includes space for both a source and a destination port, thus allowing TCP to route all data
- In most cases, both TCP and UDP use the same port numbers for the same applications
- Ports 0 through 1023 are reserved for use by known protocols.
- Ports 1024 through 49151 are known as user ports, and should be used by most user programs for listening to connections and the like
- Ports 49152 through 65535 are private ports used for dynamic allocation by socket libraries

TCP Packet Format

Bit Offset	0-3	4-7	8-15	16-18	19-31
0	**Source Port**			**Destination Port**	
32	**Sequence Number**				
64	**Acknowledgment Number**				
96	Offset	Reserved	Flags	**Window Size**	
128	Checksum			Urgent Pointer	
160	Options				
>= 160	Payload				

Establishing TCP Connections

- TCP connections are established through a three way handshake.
- The server generally has a passive listener, waiting for a connection request
- The client requests a connection by sending out a SYN packet
- The server responds by sending a SYN/ACK packet, indicating an acknowledgment for the connection
- The client responds by sending an ACK to the server thus establishing connection



Denial of Service Attacks

- Computer resources are limited (network bandwidth & memory).
- Server starts dropping packets once resources are unavailable.
- DoS attack aims at consuming such resources.
- E.g. several flooding attacks (syn flood, icmp flood etc.)

SYN Flood

- Rely on sending TCP connection requests faster than the server can process them
- Attacker creates a large number of packets with spoofed source addresses and setting the SYN flag on these
- The server responds with a SYN/ACK for which it never gets a response (waits for about 3 minutes each)
- Eventually the server stops accepting connection requests, thus triggering a denial of service.

ARP

- The address resolution protocol (ARP) connects the network layer to the data layer by converting IP addresses to MAC addresses
- ARP works by broadcasting requests and caching responses for future use
- The protocol begins with a computer broadcasting a message of the form

  who has <IP address1> tell <IP address2>
- When the machine with <IP address1> or an ARP server receives this message, its broadcasts the response

  <IP address1> is <MAC address>
- The requestor's IP address <IP address2> is contained in the link header
- The Linux and Windows command arp - a displays the ARP table

```
Internet Address      Physical Address      Type
128.148.31.1          00-00-0c-07-ac-00     dynamic
128.148.31.15         00-0c-76-b2-d7-1d     dynamic
128.148.31.71         00-0c-76-b2-d0-d2     dynamic
128.148.31.75         00-0c-76-b2-d7-1d     dynamic
128.148.31.102        00-22-0c-a3-e4-00     dynamic
128.148.31.137        00-1d-92-b6-f1-a9     dynamic
```

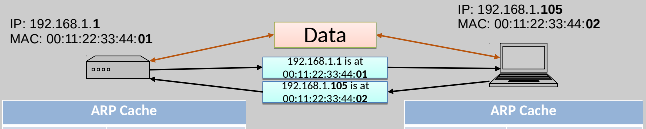
ARP Spoofing

- The ARP table is updated whenever an ARP response is received
- Requests are not tracked
- ARP announcements are not authenticated
- Machines trust each other
- A rogue machine can spoof other machines

ARP Poisoning (ARP Spoofing)

- According to the standard, almost all ARP implementations are stateless
- An arp cache updates every time that it receives an arp reply… even if it did not send any arp request!
- It is possible to “poison” an arp cache by sending gratuitous arp replies
- Using static entries solves the problem but it is almost impossible to manage!

ARP Caches



IP: 192.168.1.**1**
MAC: 00:11:22:33:44:**01**

Data

192.168.1.**1** is at 00:11:22:33:44:**01**

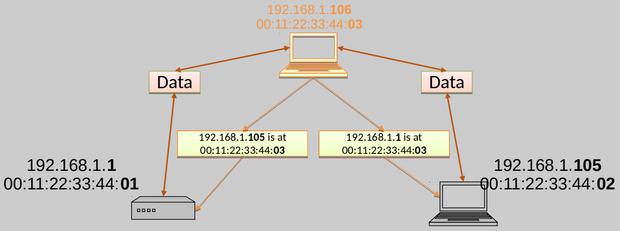
192.168.1.**105** is at 00:11:22:33:44:**02**

IP: 192.168.1.**105**
MAC: 00:11:22:33:44:**02**

ARP Cache	
192.168.1.**105**	00:11:22:33:44:**02**

ARP Cache	
192.168.1.**1**	00:11:22:33:44:**01**

Poisoned ARP Caches



192.168.1.**106**
00:11:22:33:44:**03**

Data

Data

192.168.1.**105** is at 00:11:22:33:44:**03**

192.168.1.**1** is at 00:11:22:33:44:**03**

192.168.1.**1**
00:11:22:33:44:**01**

192.168.1.**105**
00:11:22:33:44:**02**

Poisoned ARP Cache

192.168.1.**105**	00:11:22:33:44:**03**

Poisoned ARP Cache

192.168.1.**1**	00:11:22:33:44:**03**