**Learning Outcomes**

|  |
| --- |
| * Demonstrate understanding of the basic aspects of a computer network, its components, and their principles of operation * Show appreciation of how networked systems cooperate to achieve tasks involving local and remote computer systems |

**Summary**

|  |
| --- |
| Communication between computers utilises the OSI reference model to abstract the technicalities of communication away. Within the network itself, bridges and switches can be used to direct traffic, adapt to faults, and provide network security.  Every network capable device has its own MAC address code, and devices use IP addresses to identify each other. Where multiple processes are using the internet concurrently, socket IDs are also needed to ensure information flows to the right destination.  Cloud computing provides affordable data storage and access to powerful systems for information processing. |

**Lesson 1: Network Basics**

|  |  |  |
| --- | --- | --- |
| **Modern Networks**     * **OSI Reference Model** * **TCP/IP** * **Security** |  | Early computer networks were simply systems linked through a communication channel, such as wires:   * Connecting to systems in the immediate area are called **local area networks** (LAN) * More distant connections are called **wide area networks** (WAN) * Modern networks can connect across the earth   **Virtualisation** allows multiple environments and networks to coexist on the same hardware and be managed independently. This provides customisable, flexible access to a central resource.  The **Open Systems Interconnection** (OSI) reference model defines a network as having several layers relating to the hardware and software needed to achieve a network.     * Physical connections are at the lowest level and user applications at the highest * The **transport layer** ensures multiple data flows can exist, not just one * This is achieved by interacting with the **network layer** on behalf of the **session layer** * Thus, a transport layer may communicate with multiple application-session instances * Layers under the transport layer handle packet data transmissions and error corrections * **Reliable service provision** is therefore maintained, as higher levels never see the errors * The network layer also ensures the correct routing and address format for systems   As writing functional modules relating to OSI layers require vast knowledge of networking, it is common to use appropriate and maintained software libraries to ensure adherence to the needed standards.  While OSI defines a theoretical system of layers, practical implementations result in layers being merged.  An example of this is the **Transmission Communication Protocol** (TCP) and **Internet Protocol** (IP), which operate together to form the TCP/IP stack.    This was introduced in 1974 and has become one of the primary ways for systems to communicate.  Encryption can occur at multiple points in the OSI model. Applications can send and receive data with their own encryption, and other layers can incorporate encryption even if the application layer doesn’t. |
| **Network Structures**     * **Bridges** * **Switches** * **Routing** |  | The simplest network is over a shared wireless channel or a single network cable.     * This operates in a similar way to a standard bus architecture * All devices, or **network nodes**, can theoretically see each other * Any node can communicate with another by sending a packet to the destination address * However, nodes must compete for bandwidth, effective quality of service and scalability   A **bridge** is a network component that can join two networks together.  However, it can also be used to split one network into two sections, known as **network segments**.     * The bride will still allow traffic to flow from nodes on Ethernet 1 to Ethernet 2 * However, traffic may not need to cross the bridge, resulting in filtering * Therefore, when packets do not cross the bridge, system bandwidth is effectively doubled * Knowing traffic behaviour in a network allows networks to be designed for performance   Where a network has a 100 Megabyte/sec bandwidth and has been divided into two segments with 10 Megabyte/sec to cross the bridge, the improvement to network capacity can be calculated as:    A bridge has a minimum of 2 network ports: one either side of the bridge. However, it can be a multi-port bridge which is commonly known as a network switch or hub.  A **hub** is one of the least sophisticated network components, as it simple receives packets on one port and sends them out on all the other ports. No filtering takes place.  A **network switch** is like a bridge, and acts as an interchange between three or more segments:     * It retains the traffic filtering effect seen in bridges * However, it requires more information to the correct segment to output to * A lookup algorithm based on a table of destinations is likely used to do this   It is possible have multiple segments connected with multiple switches, allowing more than one path to a destination. This provides resilience to network failures and increases data capacity at a cost to complexity.  **Routing** is the process by which hardware decides which available path should be used to send data.  The destination segment is referred to as a **subnet**.  Routing of packets is based upon a fixed set of rules, known as a **static routing algorithm**. This produces fast and predictable decisions but cannot adapt to faults in the network.  An alternative approach is **dynamically adaptive routing**, where additional bandwidth is used to constantly pass information about the network between switches.   * This allows choices to be made on real time behaviours and statuses * Information could include traffic loads between various points * Potentially this could work around network faults   For the general user, bridges, switches, and routers are almost the same, as all direct traffic to some extent.  A dedicated router is used to connect to local area network to a larger system, such as the internet. Some important customers and services may even have dedicated hard line to provide constant levels of service. |

**Online**:Section 6.3-6.4, Computer Architecture and Operating Systems, University of York

**Print**:Chapter 13.1-13.5, Computer Architecture and Operating Systems, Crispin-Bailey

Chapter 11.1-11.2, Computer Systems: A Programmer’s Perspective, Bryant et al

**Lesson 2: Getting Connected**

|  |  |  |
| --- | --- | --- |
| **Networks and Services**       * **Network Services** * **Sockets Interface** |  | A device needs a unique identifier to be recognised on a network:   * **Media Access Control address** (MAC) – a permanent unique identifier assigned to the network interface device by a manufacturer * **Internet Protocol address** (IP) – a dynamic identifier assigned to any device wishing to be visible on the internet   The most widely used IP standard is the **IPV4** model, which supports 2564 addresses. However, as these have been exhausted already, the **IPV6** scheme, which supports 25616 addresses, is becoming standard.   * The need for additional addresses is partly driven by the expansion of the Internet of Things * Adoption has been hindered by the need for investment in hardware and configuration   Assigning IP addresses is a dynamic process, and an internet provider will have acquired large blocks of IP addresses and will assign these to customers using their own equipment.  An IP address will allow a device to be visible on the internet. However, this can be made simpler for users:   * An address can be linked to a **domain name** * The link is stored in a globally accessible database called the **Domain Name Server** (DNS) * This server translates textual internet domains into IP addresses * The URL can subsequently be used to reach the associated device at the IP address   If a device was moved to another service provider, they would receive a different IP address. If the DNS is updated with the new address, then users could continue to access the device with a domain name.  Within a local network, devices may have IP addresses independent of global internet IP addressing.   * This is because internal devices do not need to be individually publicly accessible * Certain IP ranges are reserved for this purpose * These private ranges are 10.x.x.x, 176.16.x.x, 192.168.x.x, where x can be 0-255 * A system known as **Dynamic Host Configuration Protocol** (DHCP) assigns these * Alternatively, they can be manually set by an administrator * If a device leaves and re-joins a network, this internal IP address may be different   Care must be taken when typing IP addresses directly, as 192.168.0.46 is not the same as 192.168.0.046.   * IPV4 addresses can be entered decimal (base 10) or octal (base 8) * The leading 0 in 192.168.0.046 tells a browser the number is octal 046 * This is converted to the decimal equivalent: 192.168.0.38   A network service is anything that facilitates a service using a network. Typically, these are user applications initiated for a particular purpose or running on a remote server ready to provide a resource.   * Email systems * Web browsers * Remote file systems * Video and audio conferencing * Online gaming * Video streaming   These services rely upon underlying OS components or modules, such as:   * **Secure Shell** (SSH) – permits login on remote computers * **File Transfer Protocol** (FTP) – facilitates exchange of data * **Hyper Text Transfer Protocol** (HTTP) – supports web browser services * **Simple Mail Transfer Protocol** (SMTP) – supports email transmission and reception * **Post Office Protocol** (POP) – supports email transmission and reception * **Common Gateway Interface** (CGI) – facilitates servers and browsers exchanging data * **X-Windows** (X11) – provides viewing of a windows-based OS remotely   As a computer can run many processes simultaneously, an IP address alone isn’t enough to connect two systems. Many connection schemes rely upon the idea of **sockets**. This can be seen in the example:     * A **socket ID** is assigned to a service running on a server when it starts * Similarly, a socket ID is assigned to a program running on the user’s system * It will attempt to link to the service on the server * IP addresses identify the systems and socket IDs identify processes possessing the sockets * eg: Server 144.32.128.230: 1228, Client 144.146.128.200:3007   Certain sockets are reserved, commonly all socket IDs up to 1023. This allows connections to be made to other systems knowing in advance what service this will connect to. |
| **Distributed Systems**     * **Web-based** |  | **Distributed systems** describe applications with functionality split across two or more different systems.  An example of this could be in a small computer used to control a heating system remotely:     * This could be accessed directly, or the system may have SSH capability * This would allow a remote connection, allowing command line instructions to be given * Alternatively, a GUI could be made for the connecting computer to simplify instructions * If sophisticated enough, it could graphically display command shell responses as well * This simple remote service application hides the intricacies of SSH and command lines * The diagram above shows the server side on the right and the client-side on the left * This is a **client-server** network application   **Downtime** is a potential drawback, as both client and server software components may need to be modified when significant changes are made to parts of the two systems.  The previous example could be configured differently. Setting up the remote system as a webserver, it could receive queries from a web browser rather than receiving direct commands from an application.     * Entering a web address would display a web page generated by the remote computer * Changes to temperature etc would be sent through an HTTP request * The remote machine would decode the message, perform tasks, and return a response * The response is likely displayed in a new or refreshed page   This allows any web browser to communicate with the server, not one specific application. All the work is done by the remote computer: providing web pages, interpreting requests, and performing tasks.   * Consequently, downtime is minimised as maintenance is only needed server side * Potentially no interruption of service is observed by the user * Furthermore, the previous application had a role in sending commands * Whereas with this model, the web browser has no knowledge of the remote system   Many programming languages are designed to deal with client-server scenarios:   * Client facing side – Javascript can provide the needed interactivity on webpages * Server facing side – PERL could manage requests, responses, and initiate programs * The local machine needs no special code or knowledge, just an appropriate browser |

**Online**:Section 6.6-6.7, Computer Architecture and Operating Systems, University of York

**Print**:Chapter 13.6-13.9, Computer Architecture and Operating Systems, Crispin-Bailey

Chapter 11.3, Computer Systems: A Programmer’s Perspective, Bryant et al

**Lesson 3: Going Virtual**

|  |  |  |
| --- | --- | --- |
| **Cloud Computing**     * **Traditional vs Cloud** * **Virtual Machines** |  | Cloud computing refers to remote distributed computing systems. Variations include:   * **Cloud storage** – data storage and retrieval, providing reliability without local costs * **Cloud computing** – computational services, providing low-cost scalability * Hybrid – a combination of the above according to a specific need   The concept of cloud computing is tied to the economics and logistics of deploying computer systems.  Traditional in-house infrastructure might involve having a physical server installed. Drawbacks include:   * Initial cost of the server may be high * Staff members are required to maintain the server * Regular data backups are needed for contingency * Data back ups may need to be kept at a second location for data resilience * The server and remote backups require both physical and digital security measures * A single server would not be suitable for a distributed workforce   By comparison, leasing remote computer services would provide several advantages:   * Long term costs may be higher, but up-front costs are much lower * The cloud service provider handles all maintenance, removing need for specialist staff * Data backups are standard with most cloud platforms, possibly even distributed * A reputable cloud provider will have specialist cyber security staff already in place * There are alternate cloud servers ready to take over if a specific server fails * The cloud platform can be accessed anywhere * Typically, payment is only required for what is used   Generally, these advantages centre around consistency, reliability, and cost.  The comparison above is based on the need for storing and retrieving data in large volumes. However, retrieving data for analysis or processing on local machines over a standard network can be a bottleneck.   * More powerful local systems could be purchased, but the problem will inevitably reoccur * These systems would also sit idle when not used, reducing their cost efficiency   Instead, modest local machines could be used for access to a remote **compute server**, where analysis and processing takes place.   * This is like the older concept of a terminal * The remote server can utilise multiple processers to speed up processing * Consequently, the efficiency of staff is improved * An integrated solution would let the data be processed without being transferred locally * Viewing only the result would reduce traffic on the local network |
| **Virtualisation** |  | **Virtualisation** describes an OS running on an emulation of a real machine. A suitably powerful computer can run multiple instances of these virtual machines (VM), the number depending on tolerable performance.  This allows users to have an interactive desktop and cloud providers do not need a machine per customer.  There are two methods to connect to this remote virtual machine:   * A user can connect through SSH to connect to the machine’s command line. This might be suitable for long computational tasks or scripts or change system configurations * A user can connect through X-Windows, duplicating the desktop for the user. This will look and behave like a normal desktop, but on more a powerful machine   Virtualisation can be supported by a machine in two different ways:     * Each requires a software agent called **hypervisor** which coordinates system resources * **Native virtualisation** uses a system native hypervisor to provide multiple VMs, each appearing to be duplicates or clones of the underlying system * **Hosted virtualisation** uses a native OS with a hypervisor as an application, which can support and run various VMs, eg Windows, Linux etc, concurrently * **Local virtualisation** doesn’t involve remote access, instead it uses a virtual machine hypervisor to run an instance of a different OS on their local machine for testing etc   Local virtualisation is useful for software engineers maintaining software, as it removes the need to keep an infrequently used specifically configured machine available or rebooting a machine into different set ups. |

**Online**:Section 6.8, Computer Architecture and Operating Systems, University of York

**Print**:Chapter 13.10-13.3, Computer Architecture and Operating Systems, Crispin-Bailey