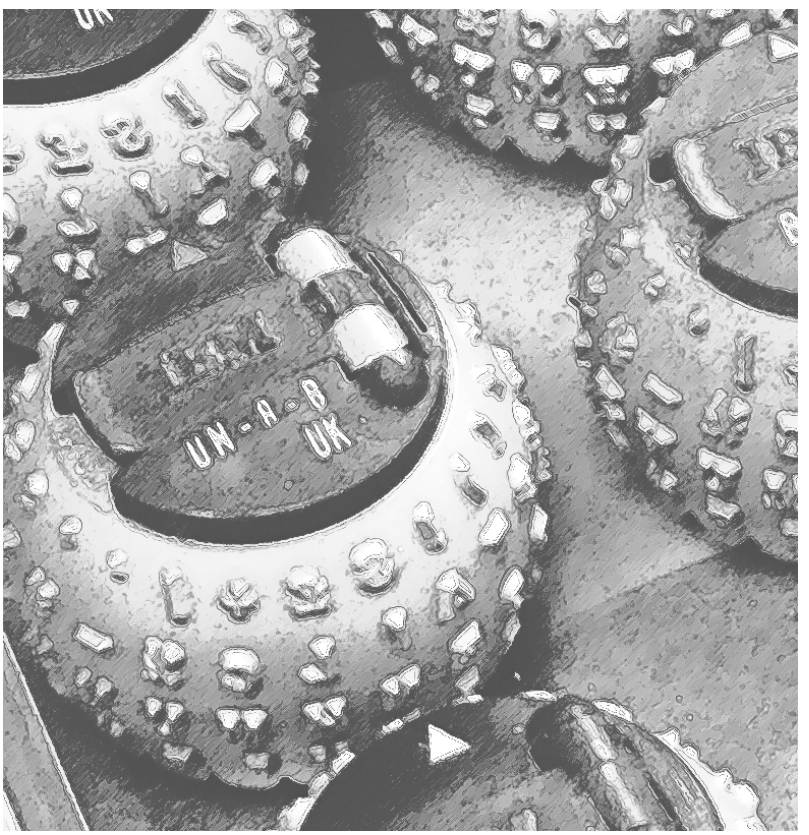
A World of Peripherals

8



Golf-Ball Print head.

8.1 Everyday peripherals

Peripherals are devices that plug into the main computer system to extend its capabilities in specific ways. Many peripherals can be classified as input devices or output devices, though as we will observe later, the lines are becoming blurred as new variations of peripherals are being developed.

A very typical system may be found to be similar to that shown in Figure 8.1. We can see that even at this level, a computer system can have multiple types of connectivity to communicate with a variety of external devices. Some key peripherals are listed in the following subsections.

8.1.1 Input devices

Input peripherals are usually designed to support user interaction, either information input or control over computer functionality and behaviour. Some common input devices are listed in Table 8.1 which shows common devices, interface standards widely used at present, and likely data rates of such a device.

Keyboard: Almost everyone is familiar with this peripheral. Keys are wired into a signal matrix, a little like a row and column grid, and every time a key is pressed, a unique combination of row and column data is generated. A decoder chip can then work out which unique key has been pressed. Some keyboards also include indicator lights and other features, technically making them output devices too. Data rates for keyboards are very low: if you could type twenty characters per second (cps), then this is potentially of similar magnitude to how many bytes are being transferred to the computer per second too[115].

[115] In practice, whilst the operating system may appear to present convenient forms of keyboard input such as one-byte **ASCII** character codes, at a lower level most keyboards rely upon **scancodes**, which can be one to three bytes per keyboard event typically. There are a number of scancode 'sets' and these offer much more information than simply indicating which character is typed after a key is pressed.

Mouse: The computer mouse was initially invented to allow movement of an on-screen text cursor, and later became indispensable for on-screen pointer/cursor movement in graphically based operating systems. Mice use a variety of techniques to detect movement on a desk or mat, sometimes mechanical, sometimes optical. Every movement of a mouse is broken down into many much smaller measurements of movement, and each of these results in data transmission to the computer. So a mouse may be sending 100's of bytes per second for a single typical mouse movement event.

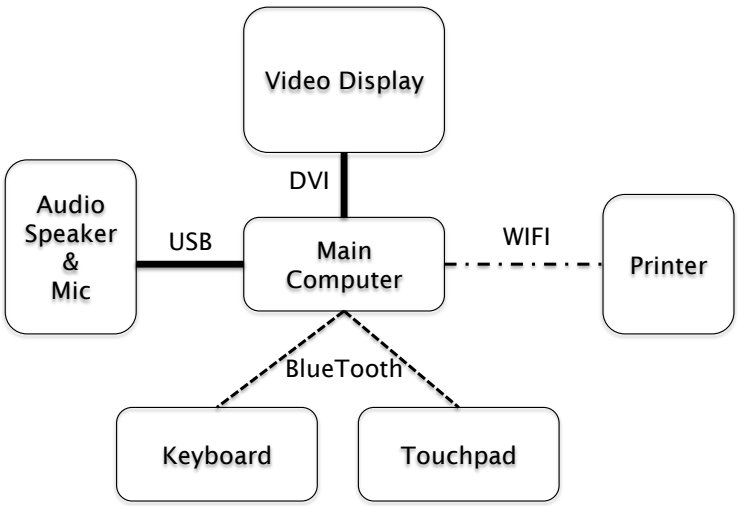


Figure 8.1: A System with typical Peripherals and common connection methods. Different peripherals may use different connection methods.

Touchpad/Trackpad: The touchpad is a touch-sensitive surface, somewhere around the size of a drinks coaster, which detects the motion of a finger or fingers, and translates these into mouse-like motion data streams, which the computer software treats as mouse motion. This is an alternative method of controlling on-screen cursors and pointers.

Touchpads permit more sophisticated inputs than a standard mouse. For example, two fingers can be used to indicate rotation, zoom, and so on (**multi-touch interface**). Users of APPLE Macintosh computers, and some smartphones, will be familiar with this concept.

TouchScreen: A touch screen is really just a touchpad capability overlaid on top of a visible screen panel, whereby the touchpad capability is transparent and does not obscure the view of the screen. Users can interact directly with screen content in this system, which is currently used heavily in tablet computers and smartphones. Generally, this capability is integrated into the device rather than being a true peripheral.

Tablet Stylus: Similar to the concept of mousepad and touch-screen, a stylus (a pen-like device) can be used to interact with the surface of the device. Where a mouse-pad technology is used this is normally called a graphics tablet, where a touchscreen is used it is just a tablet computer used with a stylus.

Microphone: Microphones are used for various purposes, including recording audio for audio production purposes. However, with suitable software,

Table 8.1: Standard Input Devices

Device	Typical Interfaces	Typical data rates
Keyboard	USB/Bluetooth	very low data rates
Mouse	USB/Bluetooth	very low data rates
Touchpad	USB/Bluetooth	very low data rates
Tablet Stylus	USB/Bluetooth	very low data rates
Microphone	USB/Bluetooth	typical data rates 24/48/96 Kilobyte/sec
Camera	USB/Bluetooth/WiFI	10Kilobyte/sec – 10Megabyte/sec
TouchScreen	Integrated	Relatively low data rates.

they are also used for detecting voice commands, for audio dictation to text, and for audio connectivity during video-conferencing[116].

Camera: As with audio, the obvious use is for capturing pictures or video for production purposes, but similarly, in a typical user context, a camera can be used for still-frame capture, video-conferencing, and potentially for activities such as motion tracking, gesture recognition, face recognition, and a few other applications[117] . In embedded applications, cameras have a much wider variety of uses. Recent advances in camera technology have resulted in 3D image capture systems, an example of which is the Kinect system used in gaming systems, but having a variety of other applications.

[116] Video-Conferencing is a process by which two way (or more) communication of audio and video can be established between parties on different computers.

[117] Examples include reading QR codes and barcodes for example.

8.1.2 Output devices.

Output devices are a little less numerous, since the primary human communication mediums are vision and sound. We could add touch, taste, and smell, and indeed, experimental systems exist for all of these as human-computer interfaces. However, they are not yet widely used. Examples of more traditional output devices are listed in Table 8.2.

The major output devices are video, audio, and print. They have different requirements in terms of data volumes, and formats. It is also the case that print content, video, and audio can be quite demanding on file storage space, and techniques have been developed to deal with this, as we will see later.

Table 8.2: Standard Output Devices.

Device	Typical Interfaces	Typical data rates
Video Display	VGA,DVI,HDMI	very high data rates
Video Projector	VGA,DVI,HDMI	very high data rates
Printer	USB/Bluetooth/WiFI	Large data quantities (may/not be fast)
Audio card	USB/Bluetooth	Similar to microphones for simple cases, perhaps 5-10 times more for complex audio such as Dolby 5.1
3D printer	USB	Medium Data rates

Video: Video displays these days are primarily based upon flat-panel display systems, very close cousins of flat-panel TVs. The variety of technologies for flat-panel displays is ever expanding. At the time of writing (2019) there are multiple coexisting technologies, LCD, TFT, OLED, QLED, each with particular capabilities. All of these systems use the concept of generating minute spots of colour (pixels) to build large scale images. Each pixel can have 24-bits of data associated with it, sometimes more. Therefore, a single frame of a video image might contain perhaps 1000 x 800 pixels, at 3 bytes per pixel = 2.4 million bytes of data. At 50 frames per second, data rates could easily approach around 120 million bytes/sec, and ultra-high definition images might contain five or ten times as many pixels. As a result, video displays require specialised connectors such as HDMI.

Video projectors: essentially similar in functionality to a flat-panel display, but images are projected via lenses onto surfaces, such as walls, screens, etc. Technologies for generating projected images include shining light through a flat-panel display matrix, or using arrays of micro-mirrors (**Micro-Mirror Array**) to create projected pixels individually.

Video Cards and GPUs: It may occur to the reader that generating such large volumes of video data at such high data rates is a rather demanding requirement for a computer system. One method of dealing with this is to move the burden to a separate processor, known as a GPU (graphical processing unit) which not only manages image generation but has many extra features to accelerate graphics creation, such as line drawing, shapes and shading algorithms, etc, all built into hardware circuits.

A standard CPU running old-school graphics routines simply could not manage the performance required for this task, and also do everything

Figure 8.2: Print heads from a golf-ball printer, which used a typeface mounted on a spherical print head. To change fonts, the operator had to manually change the print head with another one in the set. (photo with thanks to Jim Austin Computer Museum.)

else the CPU is needed for. When GPU's are mounted on plug-in modules they are referred to as graphics cards or GPU cards. Because these have very high demands on system resources, they often have their own dedicated connection to the system and CPU (the advanced graphics port: AGP, which we encountered briefly in Section 6.2).

Printers: Printers allow us to generate hard-copy or print-on-paper output. They operate in several ways. One method, the laser-printer, uses a laser to scan a drum in order to create an image using powdered ink (toner); another method uses a movable head to scan a page and spray dots of ink onto paper (ink-jet). There are other, older technologies, which we will not explore further here, but you may be interested in reading about pen-plotters, dot-matrix, and line printers if you like really old technologies. An example of a now fairly obscure printing system is shown in Figure 8.2.

Both laser and ink-jet printers have evolved to the point where they can generate very high quality monochrome or colour images, at relatively low cost. Laser printers are seen as the top-level for quality in most general purpose applications, though highly specialised printers for artwork may use very refined ink-jet technologies.

As for video images, a page, just like a video frame, is made up of millions of pixels (in print we refer to dots). A high quality image contains 600x600 dots-per-inch. An A4 page might contain over 34 million coloured dots. In order to represent image content, data is usually transferred in a format known as a PDF (Portable Document Format) file or a PS (postscript) file. Due to the way these files are defined, often a printed page will be much smaller than many tens of megabytes. After all, most printed pages are largely filled with 'white space' and only a fraction of the page is ink.

Since printers are connected via USB, Bluetooth, WiFi or direct Ethernet connection, then the process of printing usually involves transferring a PDF or PS file to the printer via that connection. This is known as spooling. In organisations where printers are shared between users, there may be a central printer management application, called a print spooler, which manages these jobs in a particular way, perhaps managing priority, permission to use various formats such as colour, payment logging, quotas, and so on.

Audio: Audio generation is another task that is generally left to a separate chip, usually an audio DSP (Digital Signal Processor). In the simplest form, audio can be monophonic, and have only one channel, and have data rates of a few thousand bytes per second. More typically, CD-quality audio requires data rates of around 84 Kilobytes/sec.[118] However, this is the data rate needed for the final stage of audio generation and the file containing the stored audio representation can make use of various techniques to reduce the size of the file without any noticeable audio quality impact.

[118] Standard CD audio has a sample rate of 44100 samples per second, where each sample is 16 bits (2 bytes). However, professional recording systems often use 96 Kilosamples/sec per channel and 24-bits per sample.

Like the white space on a page, there are elements within audio that can be removed without any noticeable difference. One of the most successful of these techniques is the **MP3** (**MPEG audio Layer 3**) file format, which can easily compress audio by a factor of 10 or 20, making storage of the audio file 10 or 20 times smaller than the raw (uncompressed) audio data. Audio cards are nowhere near as sophisticated as GPU's. However, they can still do sophisticated things like filtering, audio effects, encoding and decoding mp3 in hardware, and so on.

Typically, an audio card will plug into a PCI bus as it does not have huge data transfer requirements, and some audio modules can even operate via USB or Bluetooth quite satisfactorily.

8.2 Novel peripheral interfaces

There are several human-computer interaction modes that are as yet still more experimental than mainstream, but will no doubt be more prevalent in computing in the future. It is therefore worth briefly highlighting a few of these.

Haptics is the domain of human-computer interaction involving touch sense. We have already identified a few examples of this: a mouse, and a touchscreen, for example. However, there are others.

The dataglove is one technology that is emerging, particularly for virtual reality applications. This is, as it suggests, a glove worn on the hand but able to detect motions and joint positions as they move, allowing a user's hands to be projected into a 3-dimensional model of space, usually supported with a VR headset.

Another model is to use video tracking of limbs, hands, fingers, etc, to track motions, and gestures. This relies upon a number of technologies, both 2D and 3D camera technology, mathematical algorithms and artificial intelligence.

In order to make these systems more interactive, haptic feedback systems have been developed. The simplest case is a gaming handset with vibration to simulate some form of event-related feedback. However, more sophisticated concepts have been developed, including pressure-based sensation, or ultrasonic stimulation, to mimic the sensation of a finger in free space actually being in contact with an object that is not really there. When combined with vision and sound, these can be very convincing.

8.3 Networking connectivity

Network connectivity is an almost essential component of a modern computer system. The vast majority of computers in use today, with the exception of deeply embedded single board computers, have some form of internet or local network connectivity. Even fancy lightbulbs can be controlled by a smartphone app these days, and this is achieved via networks too.

Network connectivity is achieved by a network interface component, often a plug-in device, or a chip built into the computer motherboard. There are also different classes of network connectivity:

- Wired network connections, via cables and sockets,
- Wireless connectivity via WiFi[119] technologies.

[119] WiFi is actually a trademark, and doesn't really stand for anything technical.

8.3.1 Wired networks

By far the most common wired network connection is **Ethernet**, a standard developed to permit many computer systems to connect to a single shared network cable (in effect, Ethernet is like a bus system, at least in the simplest modes of operation).

Another system, **ATM** (**Asynchronous Transfer Mode**) is less popular but used in certain scenarios. We will not go into deep technical detail about how these systems differ. More about networks will be covered in later chapters. However, Ethernet uses the concept of competitive use of the shared Ethernet bus, such that devices may end up wanting the bus at the same time, and potentially making service levels unpredictable. ATM reserves a fixed time-slice of the network bus for each device, guaranteeing quality of service (often referred to as QOS).

An Ethernet controller is effectively a chip or digital module of a chip, which performs the Ethernet data transfer protocol. This controller may even be integrated into a processor chip to minimise extra chips on a circuit board and possibly achieve tighter coupling of Ethernet to system memory. This is important where (a) CPU effort dealing with networks needs to be minimal to save power, or (b) where data transfer between memory and network is expected to be very high.

Ethernet controllers utilise a technique known as **Carrier Sense Multiple Access / Collision Detection**, (**CSMA/CD**). In this system, devices can start to transmit at any time, without any central control. Of course this results in the possibility of **collisions** (several devices attempting to transmit at the same time). To minimise this, the CSMA/CD protocol is used as follows:

- Any device seeking to use the Ethernet bus must first check to see if the bus is already in use (this is the carrier sensing part).
- If the Ethernet bus is not busy, then any number of devices can attempt to transmit (multiple access).
- During transmission, the transmitting devices monitor the state of the bus to check that it is correct. If it is not, then this indicates that another device tried to transmit at just the same moment, and this is known as collision detection.
- If a collision is detected, all devices attempting to transmit will stop and wait a random amount of time before trying again (known as back-off-and-retry).

You may notice that this system has some similarities with I2C, which we examined in Chapter 6.9.1. This approach is quite common where there is no centralised arbiter of bus access. The negative side of this system is that there is no guarantee that a device will ever get to transmit its data. It could randomly back off and fail at each new attempt, for an indefinite period (the odds are minute, but they are never zero). On this basis, standard Ethernet is not considered predictable in its behaviour

(it is non-deterministic), and might be inappropriate in a safety critical system, unless a suitably modified standard is used.

Data transfer rates are clearly going to be a key concern for networks. On wired networks these are limited by the physical capabilities of the cabling system, the length of the cables and how many devices are competing for network bandwidth. Ethernet standards vary from 10Mbit/sec to multi-giga-bit/sec data rates, and there are many variants of the basic definition, each with different cable technologies, applications, and constraints (literally tens, if not hundreds, of variations).

8.3.2 Wireless networks

Wireless networks, sometimes referred to as **WiFi** networks, use radio frequency data transmissions in place of physical wires. This has a number of advantages, primarily the removal of the need for cabling infrastructure in a building, and the ability to mobilise devices such as laptops, smartphones, etc without having to plug them into a network in order to access internet or local network functionality. If there is a downside, it is that radio signals are prone to interference, and propagation of signals in buildings to give good network availability can sometimes be a problem.[120]

[120] Equally, the unconstrained availability of WiFi signals is also a security risk. Third parties can detect WiFi signals and attempt to eavesdrop on data content.

Wireless Ethernet is simply a variation of Ethernet that permits use of radio channels in place of wires, and shares much of the same functionality. We will be familiar with this in our homes and perhaps our offices, where broadband routers allow a wireless network to exist in that space, and connect that space to the internet. These systems typically work well over tens of metres, but can be boosted and extended in various ways, in a large office for example.

Another popular wireless data networking standard is **Bluetooth**. Bluetooth is generally used over short distances, a few metres being typical. Many peripherals can operate using Bluetooth: keyboard and mouse for example can connect to a computer system using Bluetooth wireless links. However, Bluetooth is not designed for high bandwidth in the same class as wireless Ethernet, and where devices such as shared data servers, laser printers, and so on are being connected to a system, it is wireless Ethernet that is the primary standard of choice.

As with Ethernet, there are many Bluetooth standards, some designed for long distances (100 metres perhaps, though these are rarely used) and others for low power and relatively low data rates (e.g. 2 Megabits/sec, 5-10 metres). Some Bluetooth specifications allow data transfer rates

of 24 Megabits/sec, though others are much lower, and also dependant upon the distance between the two devices (data rates are lowered when the link is too distant for a particular data rate to operate reliably[121]).

Because of the lower data rates, compared to wireless Ethernet, and the design of the radio protocol, Bluetooth can operate a relatively lower power. This makes Bluetooth attractive for mobile devices and devices powered by batteries (e.g. smartphones, wireless keyboards, and personal wearable devices).

[121] Although systems can detect bit errors in transmissions, if this results in a block of data having to be retransmitted, then the available bandwidth reduces because some of it is wasted with duplicate data. It may be more efficient to reduce the data rate and have fewer errors.

8.3.3 Network protocols and overheads

Although the raw electrical behaviour and frequency response of a network connection, be it wired or wireless, will dictate the maximum possible bit rate of that physical link, in practice, this is rarely achievable.

Firstly, the low-level hardware requirements of a network, such as CSMA/CD mechanism may introduce the need for additional bits or clock cycles to be incorporated into the transmission of data. Secondly, the transmission of data may be packaged into distinct portions, typically referred to as packets. Such packets may include an information block, known as a packet header, as well as the actual data of interest, known as the payload.

Consequently, if a particular network requires a header of **m** bits, then transmitting **n** bits of data will require **m+n** transmission bit periods. We can define the data transfer efficiency of a network protocol as follows:

Definition 8.3.1 ***Data Transfer Efficiency***

$$Efficiency = \frac{PayloadSize}{PacketSize}$$

Where PacketSize = PayloadSize + HeaderSize.

So, as an example, if a network had a payload of 128 bytes, and a header of 8 bytes, the packet size will be 136 bytes for a payload of 128 bytes. Then, its efficiency will be 94%, since $128/136 = 0.94 = 94\%$.

Now let us take Ethernet as an example. The most common Ethernet format, 'Frame Type-II', has the format parameters outlined in the following summary detail.

Ethernet Frame Type-II info

- **HEADER**
  6 bytes MAC Source address
  6 bytes MAC Destination address
  2 bytes Control Field
- **PAYLOAD**
  Min 46 bytes, Max 1500 bytes
- **CRC**
  4 byte CRC error checksum

Now, in this case, there are 18 bytes of protocol information (header and tail-end CRC values) alongside n bytes of payload data. The worst and best case efficiency would therefore be:

Ethernet example calculations

Worst Case : 46 payload bytes, 14 header bytes, and 4 CRC bytes,
packet size = 64
efficiency = 46/64 = 72%

Best Case: 1500 payload bytes, 14 header bytes, and 4 CRC bytes
packet size = 1518
efficiency = 1500 / 1518 = 98.8%

What does this mean in practice? Well, a network with 1 Gigabit/sec raw bandwidth and a protocol yielding 72% efficiency can only transfer a maximum of 720 million bits/sec of actual data (686 Mbits/sec)[122].

[122] Again, remember for transfer bandwidth we generally use decimal megabytes here.

Whilst it may therefore seem desirable to have longer payloads, and higher efficiency, this also means that network devices have to wait longer for each opportunity to use the network due to the time taken for the currently active device to complete a packet transfer, known as transmission delay.[123] Therefore, simply having very big payloads is not always a good system optimisation.

[123] At least in a simple case. Total network latency might involve routers and other factors that accumulate multiple delay components.

A useful analogy here is a crossroads with traffic lights. If the traffic switches between north-south to east-west once per minute, and the lights take 10 seconds to change, then the efficiency of traffic flow (the bandwidth) is 50/60 = 83%, and the maximum time a driver has to wait is 1 minute (the service latency).

We could decide to have the lights change every 10 minutes, and have a traffic flow efficiency of 590/600 = 98%, but one can imagine the drivers

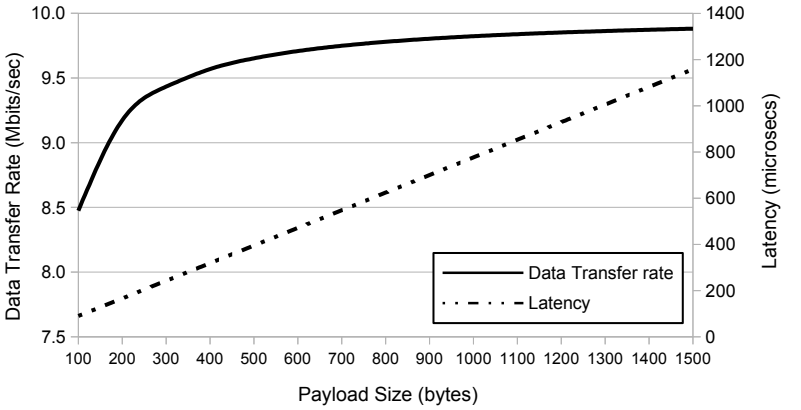


Figure 8.3: Network Efficiency and Latency (delay) tradeoff. Assumes a simple 10 Megabit raw network bandwidth, a header size of 18 bytes and a single network segment.

being rather annoyed at the 10 minute wait for each light change! The message here is that quality of service is a combination of bandwidth and latency.

Returning to our network scenario, we can attempt to represent this efficiency versus payload-size tradeoff via a graph. We can see in Figure 8.3 that as we increase payload size, efficiency of data transfer improves, approaching the maximum bandwidth, but at the same time, delay increases too.

In this example we can see that above about 500 bytes per payload, the transfer rate only improves marginally, but at the cost of rapidly increasing latency penalty. A payload size of between 500 and 1000 bytes would be a good compromise here if bandwidth was important but delay was also a secondary concern. In that case we would achieve nearly all of the peak data transfer rate potential of the system, but with much lower than maximum latency.

If, on the other hand, delay was more important than bandwidth, then a good compromise might be a payload size of no more than 500 bytes, as at this point bandwidth is not dropping off sharply but delay is much lower that the first case mentioned.

There are further subtleties here too. Consider the limitation that a packet cannot be used by a receiving processor until fully received. CRC cannot be verified until the whole packet is visible, for example. Instead of sending 1000 bytes in a single payload, sending 100 bytes ten times in a row might allow the CPU to work on the first 100 bytes whilst the next 100 is being sent (in a sense, a form of pipelining). This is even more

pertinent if a multi-core processor was assigning successively received packets to different CPU cores in turn for processing.

Likewise, rapidity of information updates may be more important than flow rate. In online gaming, players often complain of **lag**: they do something in the game but it responds too slowly, spoiling the feeling of real-time interactivity. Here, shorter packets will allow a finer grain of responsiveness even though total bandwidth is reduced.

8.4 Summary

Peripherals are, in some senses, the most important parts of the computer system. They allow us to customise its configuration, interact with the computer via input and output devices, and to produce video, audio and printed documents. However, the wide variety of devices, and the variations in the ways they connect to the computer system have resulted in many connectivity options being developed, be they wireless or wired connection, high speed busses, low speed point-to-point wires, and so on.

Although we have not explored in detail how peripherals work, for example how a mouse detects movement, we have explored how these devices demand certain degrees of data bandwidth to function sensibly, and how the connection choices and the underlying mechanisms of those connections (such as Ethernet protocol) will lead to performance constraints and tradeoffs that need careful design consideration.

As with topics mentioned in previous chapters, things will continue to evolve. New connection standards will emerge every year or two for the foreseeable future, not least because new peripheral technologies continue to be developed (the maturing capabilities and decreasing costs of VR headsets is a good recent example, driving new sets of requirements). Will we recognise computer systems in twenty years time? They may be very different.

8.5 Terminology introduced in this chapter

ATM network	Bluetooth
CSMA/CD	Dots-per-inch
Ethernet	Golf-ball print head
Lag (network)	Micro-mirror Array
Multi-touch interface	Packet collision
Pixel	Tablet stylus
WiFi	Wireless network

These terms are defined in the glossary, Appendix A.1.