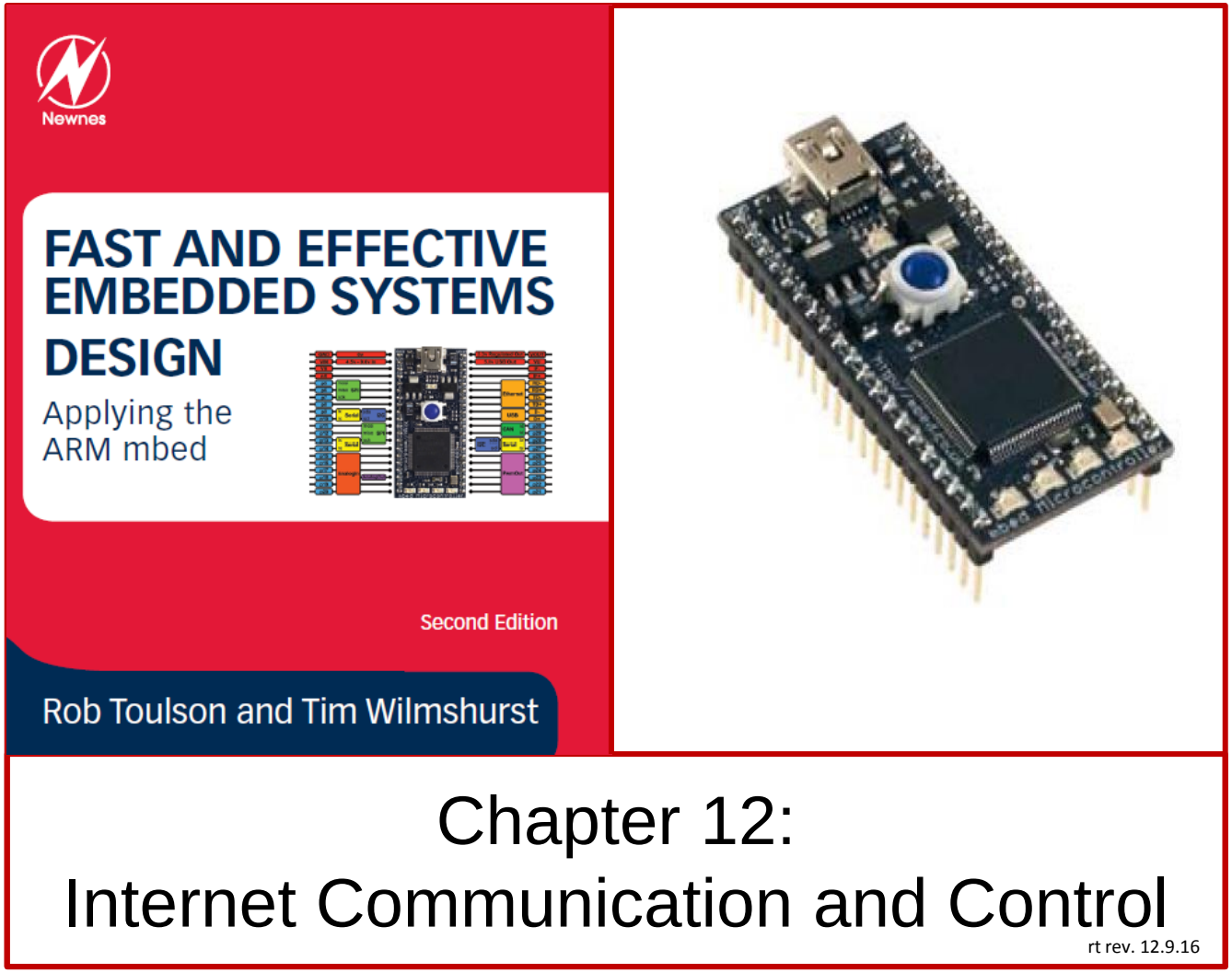


Chapter 12: Internet Communication and Control

rt rev. 12.9.16

If you use or reference these slides or the associated textbook, please cite the original authors' work as follows:

Toulson, R. & Wilmshurst, T. (2016). Fast and Effective Embedded Systems Design - Applying the ARM mbed (2nd edition), Newnes, Oxford, ISBN: 978-0-08-100880-5.

www.embedded-knowhow.co.uk

Introduction to Internet Communications

- Computers and devices connect to the Internet through the Ethernet communications protocol, or through wireless (wi-fi) communications.
- Public data, stored on *servers* as web pages, can be accessed by *client* devices that are connected to the Internet.
- it could also be an embedded system that accesses and utilises data stored on the server and responds to control messages sent by the server.
- The Internet nowadays facilitates advanced interactive applications. The mbed compiler for example is accessed entirely through the Internet, which means that no additional software needs to be installed for a developer to work with the mbed.
- The Internet nowadays allows for machines and devices to communicate with each other autonomously, i.e. with no (or limited) human interaction.
- The concept of connecting literally billions of devices and sensors to the Internet is referred to as the *Internet of Things.*

Introducing Ethernet

Ethernet is a serial protocol which is designed to facilitate network communications. Any device successfully connected to the Ethernet can potentially communicate with any other device connected to the network.

Networks are often described as one of two types:

- Local area network (LAN) – usually for devices connected together in close proximity, perhaps in the same building and often without internet access.
- Wide area network (WAN) – which describes a network of devices over a greater geographic area, usually connected by the internet.

Ethernet communications are defined by the IEEE 802.3 standard and supports data rates up to 100 Gigabits per second.

Ethernet uses differential send (Tx) and receive (Rx) signals, resulting in 4 wires labelled RX+, RX-, TX+ and TX-.

Ethernet messages are communicated as serial data packets referred to as *frames*. Using frames allows a single message to hold a number of data values including a value defining the length of the data packet as well as the data itself.

Introducing Ethernet

The Ethernet frame defines its own size, which means that only the necessary amount of data is communicated with no wasted or empty data bytes.

Ethernet communications need to pass a large quantity of data at high rates, so data efficiency is a very important aspect.

The use of frames allows an efficient method. Each frame also includes a unique source and destination MAC address.

The frame is wrapped within a set of *preamble* and *start of frame* (SOF) bytes and a *frame check sequence* (FCS) which enables devices on the network to understand the function of each communicated data element. The standard 802.3 Ethernet frame is constructed as shown below.

Preamble	Start of frame delimiter	Destination MAC address	Source MAC Address	Length	Data	Frame Check Sequence	Interframe gap
7 bytes of 10101010	1 byte of 10101011	6 bytes	6 bytes	2 bytes	46 – 1500 bytes	4 bytes	

Introducing Ethernet

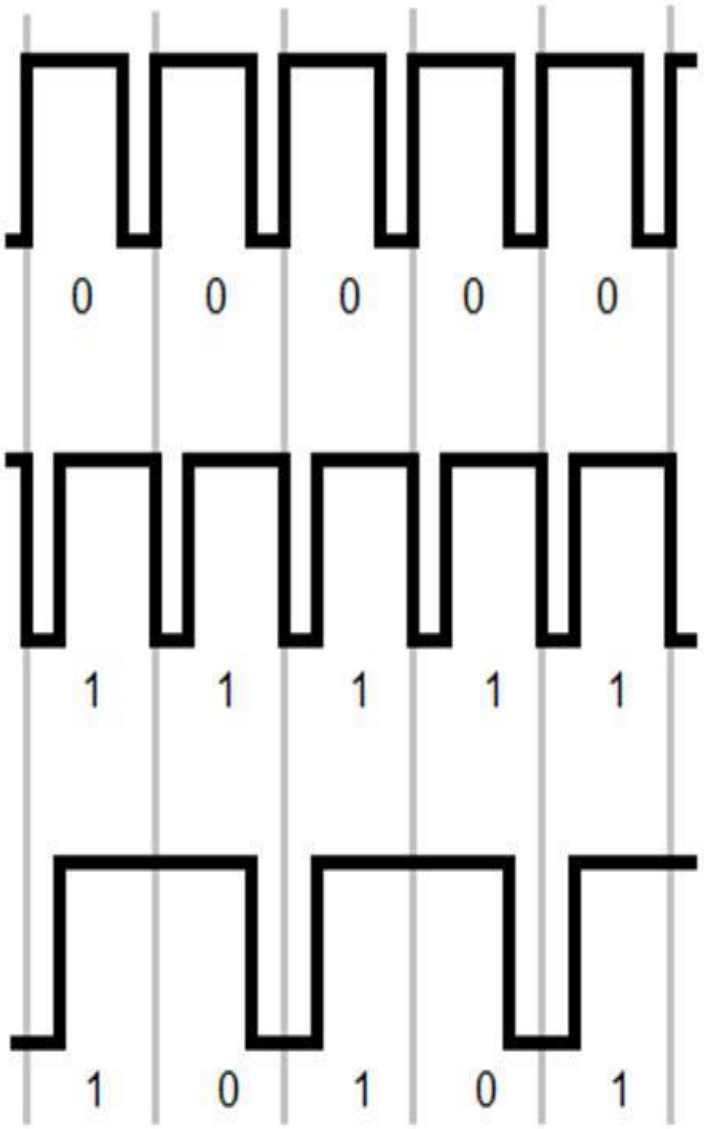
The Ethernet data takes the form of the *Manchester encoding* method, which relies on the direction of the edge transition within the timing window.

- If the edge transition within the timing frame is high-to-low, the coded bit is a 0
- If the transition is low-to-high then the bit is a 1

The Manchester protocol is very simple to implement in integrated circuit hardware

As there is always a switch from 0 to 1 or 1 to 0 for every data value, the clock signal is effectively embedded within the data.

Even when a stream of zeros (or ones for that matter) is being transmitted, the digital signal still shows transitions between high and low states.



Implementing simple mbed Ethernet communications

Function	Usage
Ethernet	Create an Ethernet interface.
write	Writes into an outgoing Ethernet packet.
send	Send an outgoing Ethernet packet.
receive	Receives an arrived Ethernet packet.
read	Read from a received Ethernet packet.
address	Gives the Ethernet address of the mbed.
link	Returns the value 1 if an Ethernet link is present and 0 if no link is present.
set_link	Sets the speed and duplex parameters of an Ethernet link.

```
/* Program Example 12.7: Ethernet write - sends two data bytes every 200 ms
from an mbed's Ethernet port. The two byte values are arbitrarily chosen as
0xB9 and 0x46.
                                                                        */
#include "mbed.h"
#include "Ethernet.h"
Ethernet eth;                       // The Ethernet object
char data[]={0xB9,0x46};            // Define the data values
int main() {
    while (1) {
        eth.write(data,0x02);       // Write the package
        eth.send();                 // Send the package
        wait(0.2);                  // wait 200 ms
    }
}
```

Ethernet communication between mbeds

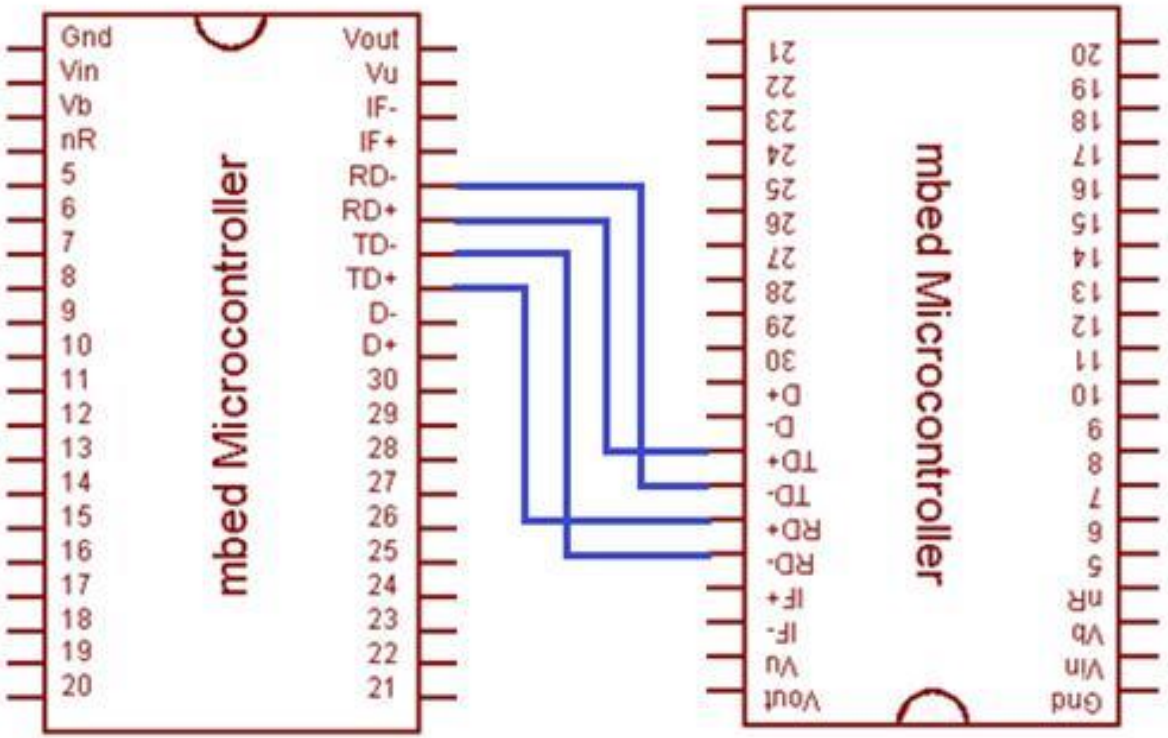
```
/* Program Example 12.8: Ethernet read - allows an mbed to read Ethernet data traffic
and display the captured data to the screen
                                                                              */
#include "mbed.h"
Ethernet eth;                          // Ethernet object
Serial pc(USBTX, USBRX);               // tx, rx for host terminal coms
char buf[0xFF];                        // create a large buffer to store data
int main() {
  pc.printf("Ethernet data read and display\n\r");
  while (1) {
    int size = eth.receive();              // get size of incoming data packet
    if (size > 0) {                        // if packet received
      eth.read(buf, size);                 // read packet to data buffer
      pc.printf("size = %d data = ",size); // print to screen
      for (int i=0;i<size;i++) {           // loop for each data byte
        pc.printf("%02X ",buf[i]);        // print data to screen
      }
      pc.printf("\n\r");
    }
  }
}
```

Note that the Ethernet read program first defines a large data buffer to store incoming data. The program then continuously evaluates the size of any Ethernet data traffic.

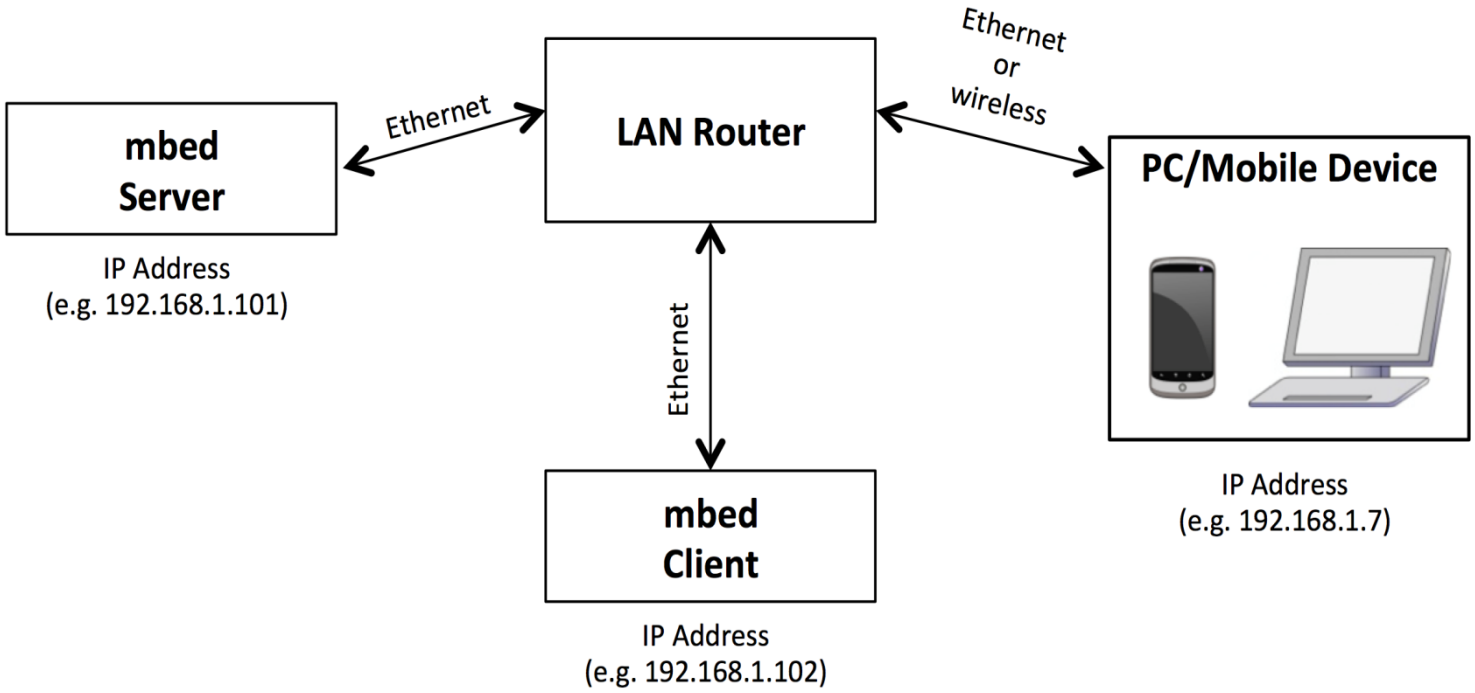
If the data packet has a size greater than zero the display loop is entered. The size of the data package along with the data is then displayed to the host terminal.

Ethernet communication between mbeds

To successfully communicate between the two mbeds, a crossed signal connection is required, as shown



Local Area Network communications with the mbed



Within the LAN shown, it is possible for the PC or mobile device to remotely access files stored on the mbed server. Equally, it is possible for the mbed client to access files and data stored on the mbed server, as well as for the mbed server to control the mbed client through remote control protocols.

The PC client can also request the mbed server to send control messages to the mbed client, hence allowing the PC client to indirectly control the mbed client. This communication can all be implemented as long as the local 32-bit *IP address* - sometimes referred to as the *private IP address* - of each device is known.

Initiating a simple LAN connection with the mbed

A standard Ethernet socket (RJ45) is required to connect the mbed Ethernet port to a network hub or router. If using the mbed application board, it is simple to use the built in Ethernet socket.

Utilising the mbed **EthernetInterface** and **mbed-rtos** libraries, Program Example 12.3 creates an Ethernet communications interface and requests the router to automatically assign an IP address to the mbed. The program then prints the assigned IP address to a host terminal.

```
/* Program Example 12.3: Opening an Ethernet network interface
                                                */
#include "mbed.h"
#include "EthernetInterface.h"

EthernetInterface eth;          // create ethernet interface

int main() {
  eth.init();                   // initialise interface with DCHP
  eth.connect();                    // connect and open communications
  printf("IP Address is %s\n", eth.getIPAddress());  // display IP address
  eth.disconnect();                 // disconnect
}
```

Using the mbed as a Hypertext Transfer Protocol (HTTP) file server

When connected to an Ethernet network, the mbed can be configured to host data files that can be accessed using the Hypertext Transfer Protocol from another PC or device on the LAN. To implement this we can make use of the bespoke **HTTPServer** library.

When using the **HTTPServer** library, it is necessary to specify *request handlers* for the server. Request handlers are specific software definitions that inform the server what types of Ethernet messages to respond to, i.e. requests for the server to do something.

Handlers are defined by using the **addHandler( )** function and specifying the type of handler that is required. The **HTTPServer** library only supports two different types of handler, one of which is a file system handler that uses the handler name **HTTPFsRequestHandler** in program code.

Additionally the file system handler needs to be informed where to locate and store files in physical memory; this is managed by defining a **LocalFileSystem** object.

Once the file server is initiated, it is necessary to inform the server which *Transmission Control Protocol (TCP) port* it should be routed to. HTTP communications are usually defined to use port 80.

Once the server is started, the **poll( )** function is required to be called regularly, in order to respond quickly to server requests from external clients.

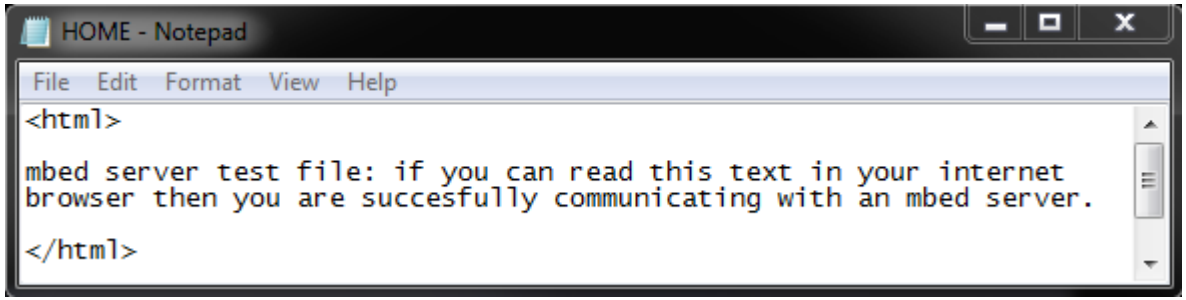
Using the mbed as a Hypertext Transfer Protocol (HTTP) file server

```
/* Program Example: 12.4 mbed file server setup
                                                */
#include "mbed.h"
#include "EthernetInterface.h"
#include "HTTPServer.h"
#include "FsHandler.h"

EthernetInterface eth;            // define Ethernet interface
LocalFileSystem fs("webfs");      // define Local file system
HTTPServer svr;                   // define HHTP server object

int main() {
  eth.init("192.168.1.101","255.255.255.0","192.168.1.1"); // initialise
  eth.connect();                              // connect Ethernet
  HTTPFsRequestHandler::mount("/webfs/", "/"); // mount file server handler
  svr.addHandler<HTTPFsRequestHandler>("/");   // add handler to server object
  svr.start(80, &eth);                    // bind server to port 80
  while(1)
  {
    svr.poll();          // continuously poll for Ethernet messages to server
  }
}
```

In order to compile Program Example 12.4 the **HTTPServer** library should be imported along with the mbed official **EthernetInterface** and **mbed-rtos** and **mbed-rpc** libraries.

Using the mbed as a Hypertext Transfer Protocol (HTTP) file server

Using a standard text editor, create a hypertext markup language file (which requires a **.htm** filename extension) called, for example, **HOME.HTM**. Enter some example text in the file, so that it is obvious when the Internet browser has correctly accessed the file (see example file below).



Create this file and save it to the mbed via the standard USB cable connection.

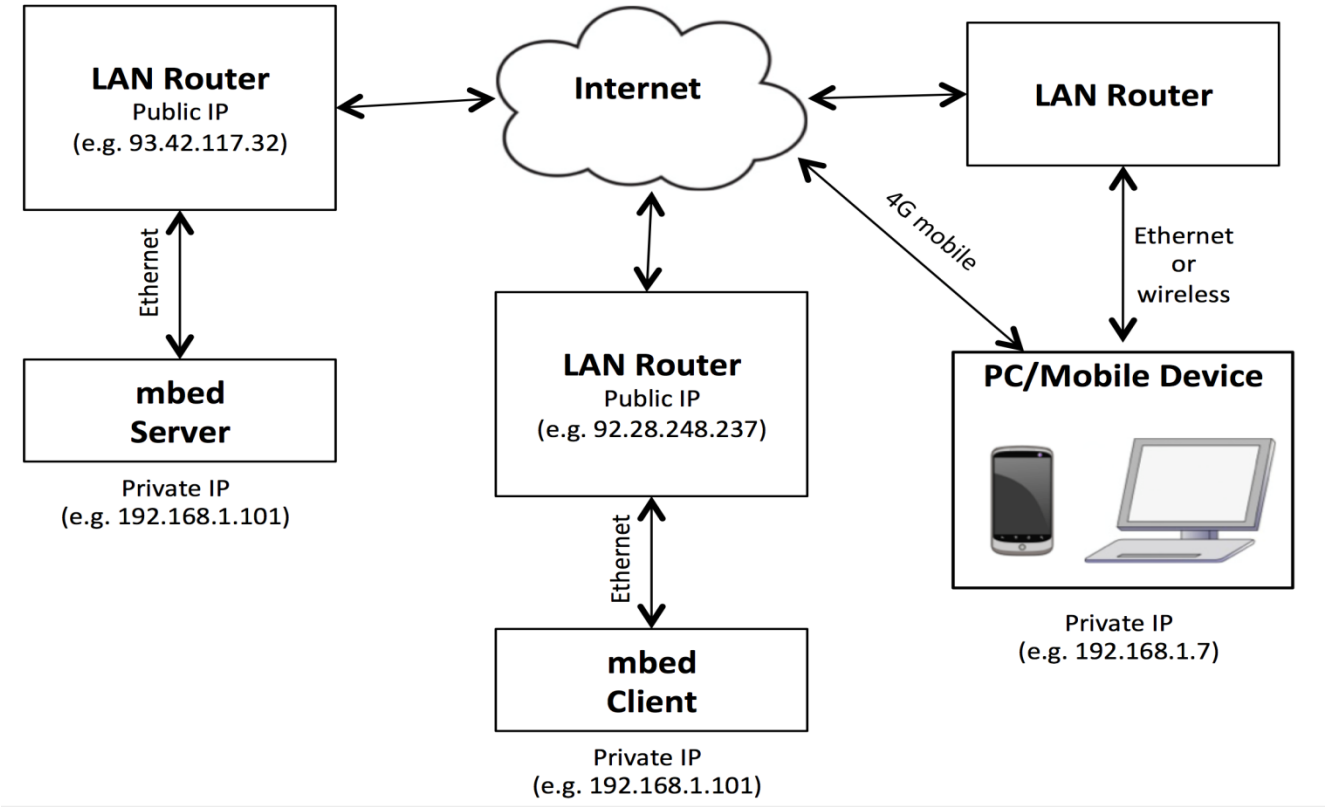
It is now possible to access the **HOME.HTM** file stored on the mbed from any other computer or client device that is successfully connected to the router that manages the LAN. To do this open a web browser application on a connected PC, and type in the following navigation address:

http://192.168.1.101/HOME.HTM

Successful navigation to this network address should bring up the HTML text as contained in the **HOME.HTM** file.

Read further:
Explore and implement the Remote Procedure Calls (RPC) examples shown in Section 12.4

Using the mbed with Wide Area Networks



The main obstacle to overcome with WAN communications, is to understand the *public IP address* of each LAN. All routers and LANs on the Internet are assigned a unique public IP address, which allows connectivity from anywhere in the world, as long as the unique public IP is known.

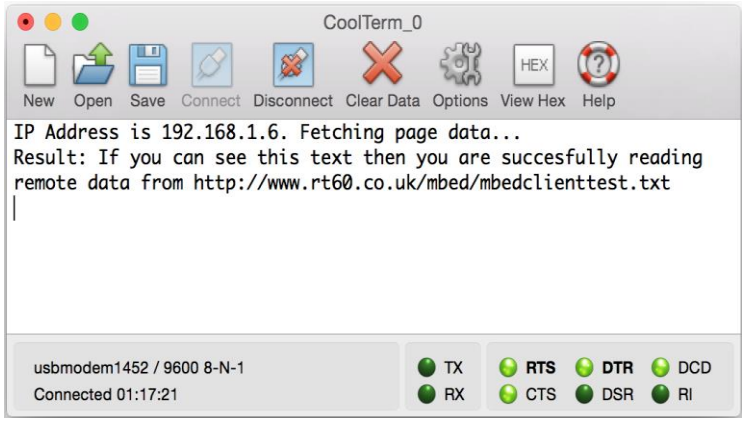
For any device connected to the Internet it is possible to deduce the public IP address by visiting a website such as https://www.whatismyip.com

Using the mbed with Wide Area Networks

It is possible to use the mbed as a HTTP client in order to access data from the Internet. To do this we rely on another network interface library called **HTTPClient**.

Program Example 12.7 enables the mbed to connect as an HTTP client to a remote online server and access and display a text string from within a text file held on the server. The test file we use in this example is stored online at: http://www.rt60.co.uk/mbed/mbedclienttest.txt

```
/* Program Example 12.7: mbed HTTP client test
                                                */
#include "mbed.h"
#include "EthernetInterface.h"
#include "HTTPClient.h"

EthernetInterface eth;
HTTPClient http;
char str[128];

int main() {
  eth.init("192,168,1,101","255,255,255,255","192,168,1,1");
  eth.connect();
  printf("Fetching page data...\n");
  http.get("http://www.rt60.co.uk/mbed/mbedclienttest.txt", str, 128);
  printf("Result: %s\n", str);
  eth.disconnect();
}
```



The Internet of Things

It is therefore possible to make data that the mbed gathers accessible to the outside world, as well as enabling mbed systems to be controlled remotely through the Internet.

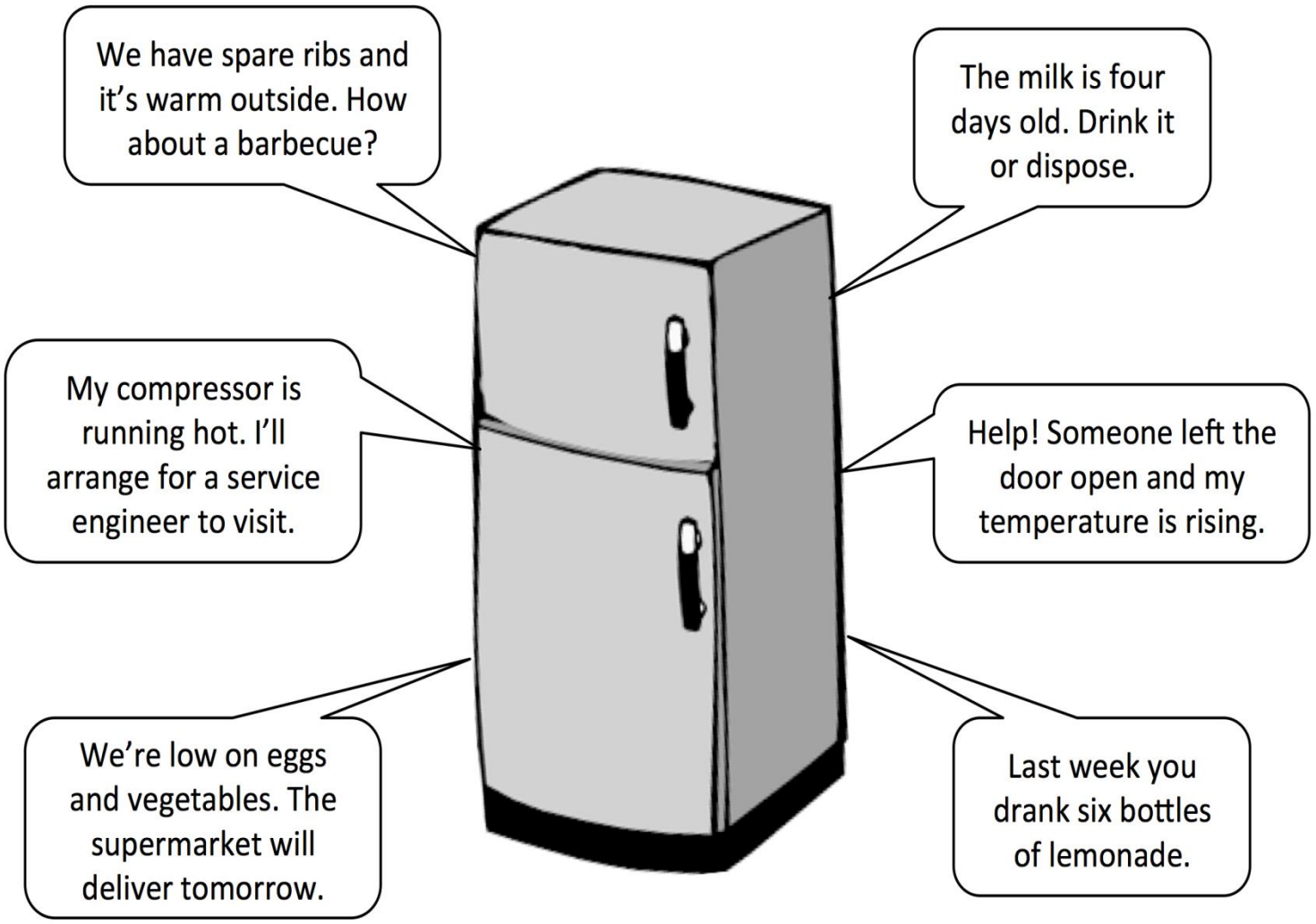
The concepts discused are the fundamental building blocks for a global *Internet of Things,* which is a phrase that has evolved to refer to a worldwide network of everyday objects and sensors that are connected to the Internet, allowing remote access to information that can be used to control and enhance everyday activities, whilst interacting with the mobile Internet devices which people are now carrying regularly, for example smartphones.

IoT data includes billions of sensors and databases that are made available for monitoring through the Internet, from travel and transport data, to environmental data, sports results and status data of mechanical devices.

The IoT concept includes reference to the term *cloud computing*, which utilises servers and data memory storage locations that are only accessed through the Internet. Companies such as Google, Microsoft, Apple and Dropbox all provide their own cloud-based services and one day it is anticipated that most of our personal and professional data (and programs) will be stored in the cloud.

More recently the IoT concept has expanded to include everyday objects attached to the Internet. Examples include the washing machine that can alert the repair man to an impending fault, the vending machine that can tell the Head Office it is empty, the manufacturer who can download a new version of firmware to an installed burglar alarm, or the home owner who can switch on the oven from the office or check that the garage door is closed.

An Internet of Things fridge



The Global Internet of Things

The potential anticipated power of IoT can be realised when Internet ‘things’ start to interact automatically and intelligently with other Internet ‘things’.

Some things, such as computers and smart phones can connect directly to the Internet through Ethernet or standard mobile communications protocols, whereas other devices will connect to the Internet through a local gateway or router.

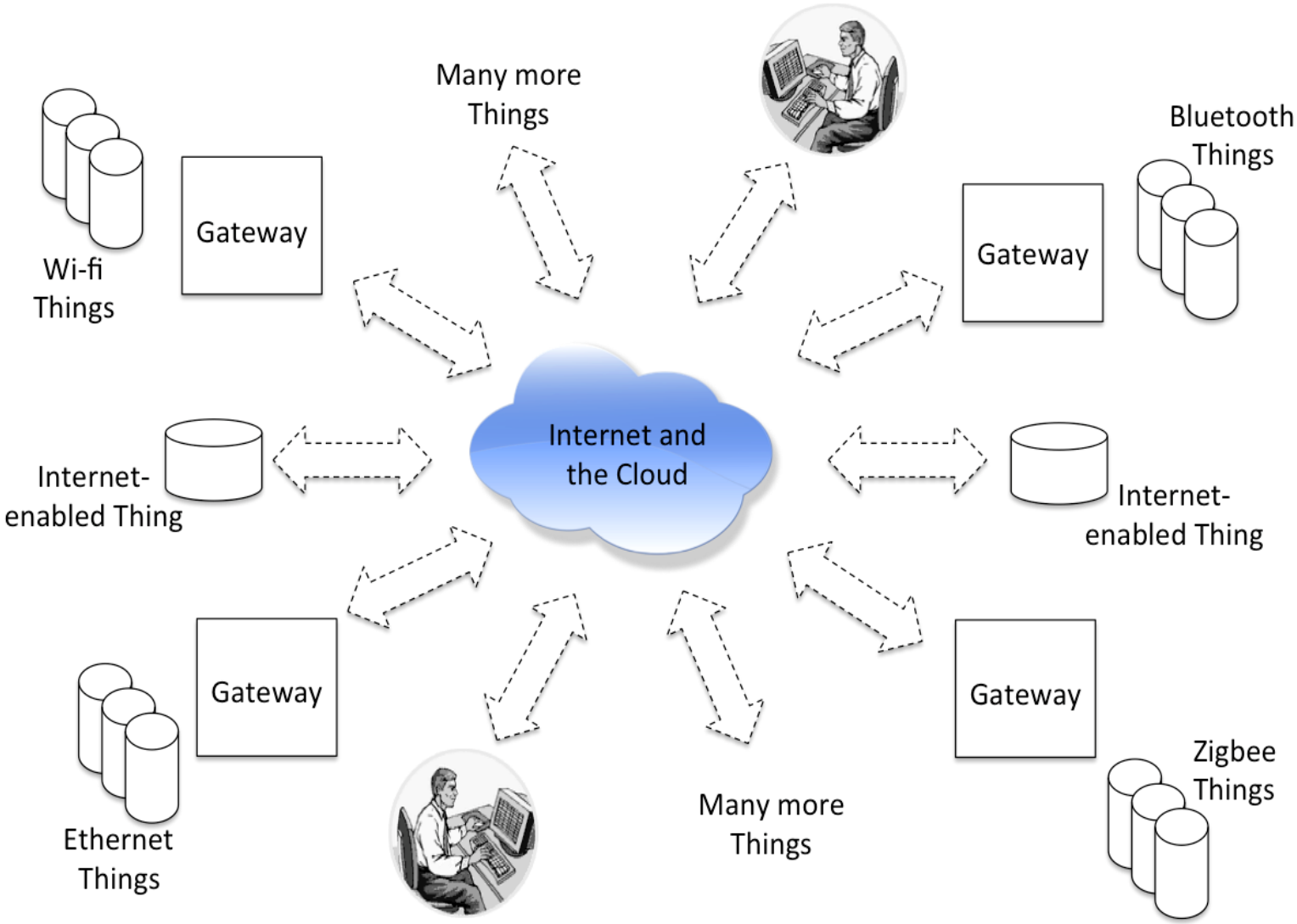
For example, electronic textiles and personal healthcare devices are becoming more widely available and it is not difficult to imagine a person who wears clothes (including wristbands and shoes) that can measure and record data about the person’s location, heart rate, body temperature, posture, perspiration levels, exercise routines and potentially even their emotional state.

This data can be measured by small, low-energy devices that are built into the clothes we wear and transmit information over Bluetooth, or another wireless communications protocol, to the user’s Internet-connected smartphone or watch.

The data is continuously sent to the cloud and can be accessed by the user and other people (or devices) that the data owner wishes to share their data with, such as family members, healthcare organisations, or home automation systems.

The IoT concept extends further to represent smart cities, which are made up of houses and communities using home automation and smart transport systems. The smart home has a number of IoT devices – not only the IoT fridge - but also Internet-connected heating systems, lighting and automatic windows that can be controlled through mobile devices, Internet-connected security systems, and smart renewable energy systems.

The Global Internet of Things



Opportunities and challenges for the Internet of Things

IoT Opportunity	IoT Challenge
Remote Control of Hardware Systems enables devices that would normally be controlled by physical switches and potentiometers to be controlled remotely through graphical user interfaces.	**Security** of IoT devices is a major concern. When devices are connected to the Internet it is feasibly possible for unauthorised persons to access the data and to interfere with control systems.
Real-Time Status Monitoring allows any IoT enabled device to be monitored remotely and historical charts of performance and functionality to be gathered.	**Reliability** is a challenge because continuous Internet connectivity is required, yet it can be sporadic and unreliable in some locations.
Intelligent Connected Functionality can be realised by the sharing of data between devices; meaning the status of one device can inform the action of another device.	**Robustness** of IoT products is a challenge because they are potentially exposed to more unexpected use, environmental conditions and accidents.
Remote Diagnostic Analysis allows users and manufacturers to constantly monitor the performance of their products after sale.	**Online Services** are required to enable IoT customers to register their products and access data associated with those products, bringing additional development costs.
Remote Firmware Updates allow functionality to be enhanced or modified remotely, enabling errors to be corrected or new features to be included.	**Graphical Interface Development** skills will be required for organisations to create bespoke user interfaces and mobile applications for accessing and controlling IoT devices.
User Data Gathering enables product developers to evaluate how customers use their products and use those statistics for future product development.	**Bespoke Designs** mean that potentially many different approaches to IoT design will be implemented over the coming years, making some devices compatible with each other and others unable to communicate.
Improved Consumer Experiences through graphical interfaces, plug-and-play products and improved customer support and education features.	**Standardisation and Legislation** is a significant challenge – when billions of devices are connected to the Internet, who will ensure they are all performing correctly and ethically?

mbed and the Internet of Things

In the previous network examples, we have always configured the mbed as a HTTP server that manages its own data traffic. The mbed needs to continuously poll (i.e. continuously check) for messages and a browser needs to regularly poll for changes in mbed status data in return.

This setup causes demand on the mbed, particularly if data is to be sent and received at the same time, if data is sent rapidly and continuously, and if data connections are initiated by more than one client at once.

As a result the RPC and HTTP server examples shown before are functional, but are prone to fail when overloaded.

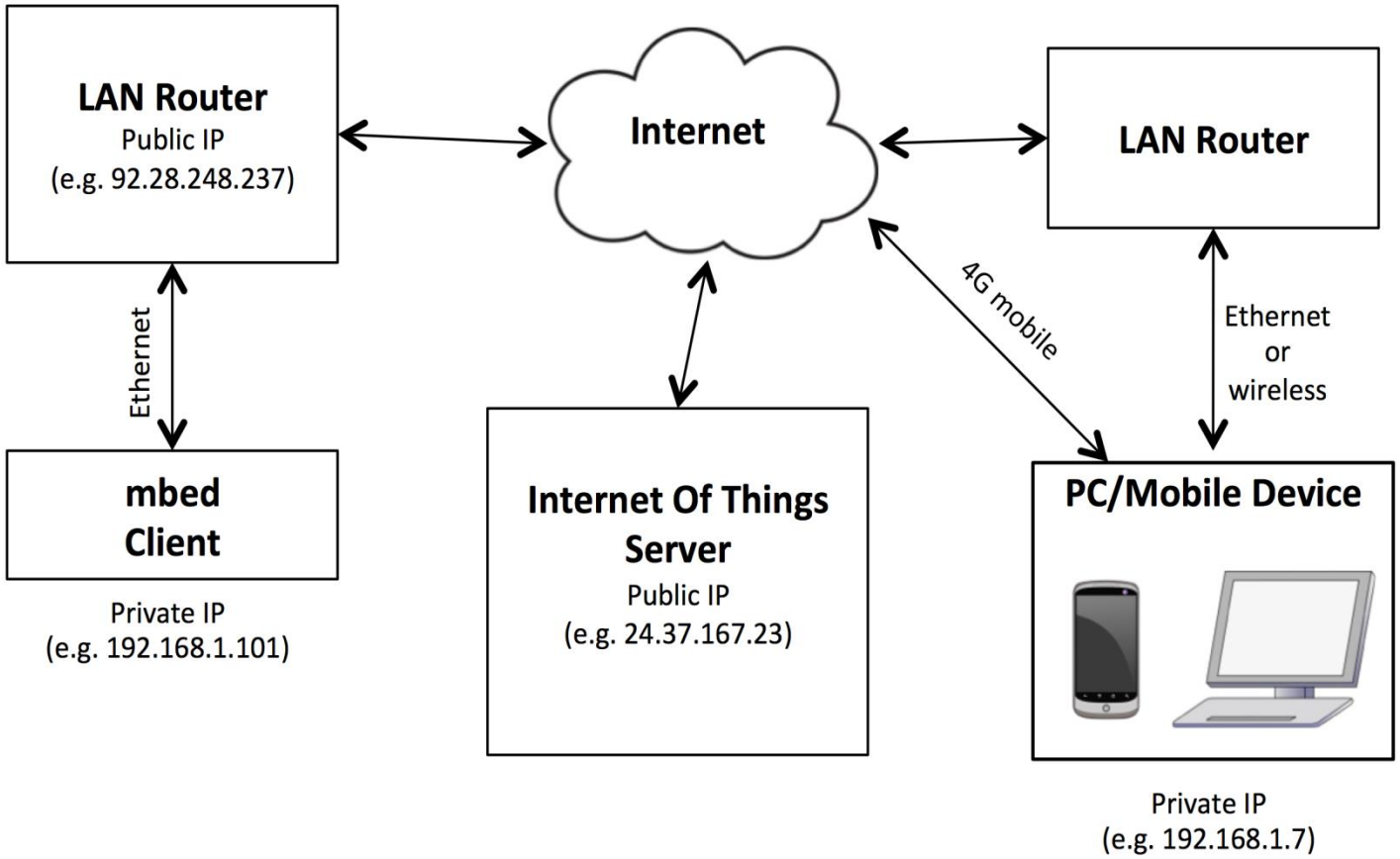
The IoT concept relies on continuous reliability and fast data transfer rates, so it is clear that a different solution is required for more advanced IoT applications with the mbed

One method is to use a dedicated and robust IoT server that is connected to the Internet and handles all messaging traffic between clients. This is a particularly advantageous method for using the mbed as an IoT device, as all mbed devices can be configured as clients and avoid the complex processing responsibilities of the server.

The server in this setup can be built with substantial processing power and housed in a remote industrial location, where it can be as large and powerful as is necessary to manage all the clients that may wish to communicate with it.

ARM have implemented this application of mbed IoT by using the open source HTML5 *WebSockets* protocol, which allows bidirectional communications between a server and client devices or browsers.

mbed and the Internet of Things



mbed and the Internet of Things

The WebSocket approach for implementing mbed IoT projects is still susceptible to security breaches. It is important that would-be criminals cannot hijack IoT communications, access confidential information and data, or take control of physical devices that are IoT enabled.

One solution is to ensure that all data communication packets over the IoT network are encrypted with advanced algorithms that mean only the intended recipients can access and utilise the contained information.

This needs to be handled by both the server and the client for encryption and decryption, meaning that every IoT device has to be programmed with encryption software within. Given that most embedded systems developers are not encryption experts, this causes a significant headache and is a barrier for innovation around IoT.

Moving forwards, ARM have chosen the mbed to be their flagship IoT development platform for the future. They intend to use the mbed RTOS as a low-level software platform that can manage all encryption and decryption tasks without the programmer needing to become an expert.

The mbed RTOS therefore adds this to its list of multithreading responsibilities. In the coming years they are planning to launch new suites of mbed IoT development frameworks, tools and services.

The commercial mbed website (www.mbed.com) explains how they foresee the future of the mbed for IoT applications, ensuring that the platform will have a significant role to play in the ‘world of connected everything’.

Chapter quiz questions

1. Describe the ‘Manchester’ digital communication format for Ethernet signals.
2. Sketch the following Ethernet data streams, as they would appear on an analog oscilloscope, labelling all points of interest:
    - 0000
    - 0101
    - 1110
3. What are the minimum and maximum Ethernet data packet sizes, in bytes?
4. What does the terms ‘Gateway Address’ and ‘Network Mask’ refer to?
5. What is the differences between a public IP address and a private IP address?
6. What does the term RPC refer to? Give a brief explanation of the use of RPC in embedded systems.
7. What does the concept of the Internet of Things (IoT) refer to? Give two examples of IoT devices.
8. Name three opportunities and three challenges associated with Internet of Things systems.
9. Describe three IoT systems that might be found in a ‘smart home’ and explain the benefits that each system brings to the household.
10. Draw a wide area network diagram of an mbed IoT application that serves three remote mbed devices and can be accessed by a mobile device or Internet connected PC.

Chapter review

- Ethernet is a high-speed serial protocol which facilitates networked systems and communications between computers, either within a local network or through the World Wide Web.
- LANs and WANS use Ethernet and wireless communications to connect Internet servers and clients through routers, allowing web pages, data and control messages to be accessed between devices.
- The mbed can communicate through Ethernet to access data from files stored on a data server computer.
- The mbed can be configured to act as an Ethernet file server itself, allowing data stored on the mbed to be accessed through a network.
- The mbed Remote Procedure Call (RPC) interface and libraries allow mbed variables and outputs to be manipulated from an external client through the Internet.
- WebSockets allow robust data communications between a server and mbed clients, meaning that mbed devices can programmed to be controlled remotely through the Internet by accessing web pages and mobile applications.
- The Internet of Things (IoT) refers to the concept of everyday objects and devices being connected to the Internet, allowing them to be controlled and analysed from anywhere in the world.
- IoT concepts bring many advantages with respect to massively connected systems, local or global transfer of data, and intelligent systems on a grand scale. IoT does however bring challenges with respect to real-time data, security and reliability.