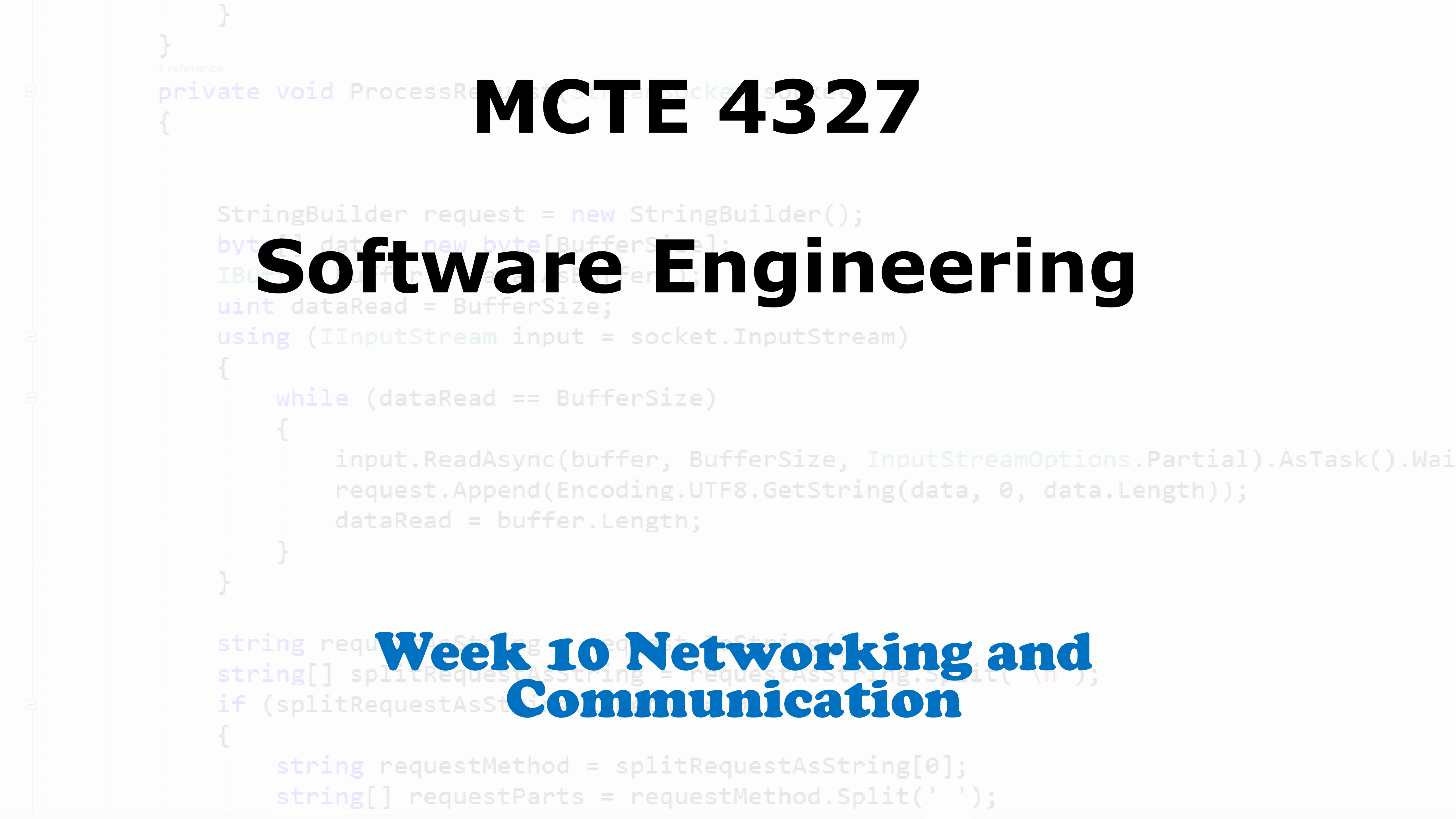
MCTE 4327

Software Engineering

Week 10 Networking and Communication



```
        }
    }
    1 reference
    private void ProcessRe
    {

        StringBuilder request = new StringBuilder();
        uint dataRead = BufferSize;
        using (IInputStream input = socket.InputStream)
        {
            while (dataRead == BufferSize)
            {
                input.ReadAsync(buffer, BufferSize, InputStreamOptions.Partial).AsTask().Wai
                request.Append(Encoding.UTF8.GetString(data, 0, data.Length));
                dataRead = buffer.Length;
            }
        }

        {
            string requestMethod = splitRequestAsString[0];
            string[] requestParts = requestMethod.Split(' ');
```

Outline

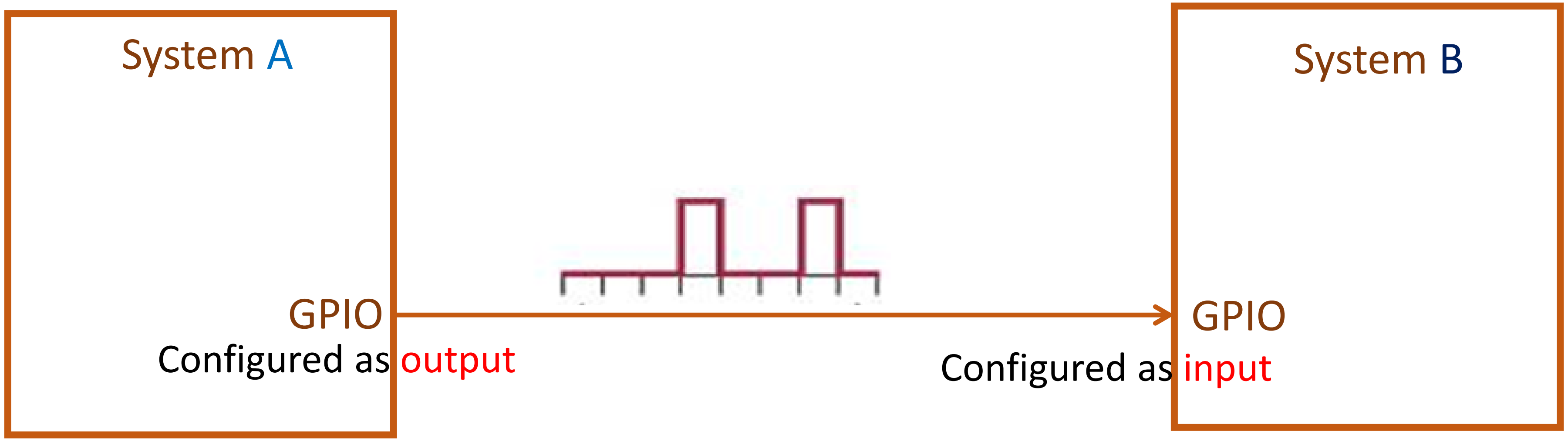
- Protocol
- HTTP
- Address and Ports

Motivation

Suppose System A has to send **a single byte** of information to System B.

Questions

- If the GPIO of the receiver is low, is the sender sending 000000 or not sending anything?
- How fast is the sender sending?
- Is the sender sending the MSB first or the LSB first?



System A

GPIO

Configured as output

System B

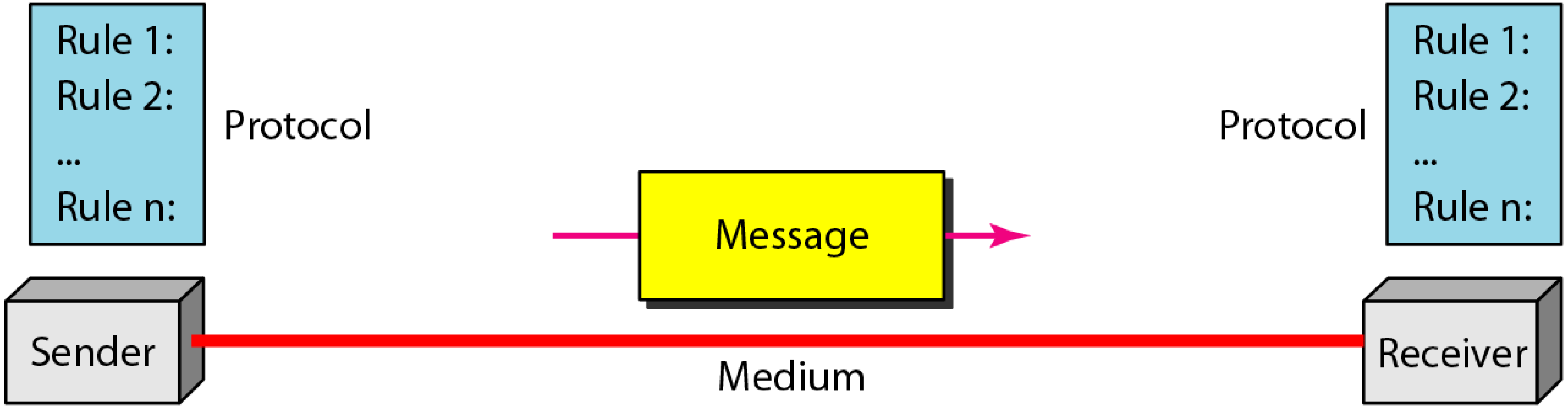
GPIO

Configured as input

Protocols

- The communicating devices need to agree on rules to send and receive data.
- This set of rules is called a protocol.
- Without a protocol, two devices may just be connected but not be communicating.

Examples: HTTP, FTP, MQTT, SMTP, POP, TCP, UDP, IMAP, SOAP



Example :

SMTP

S = Server
C = Client

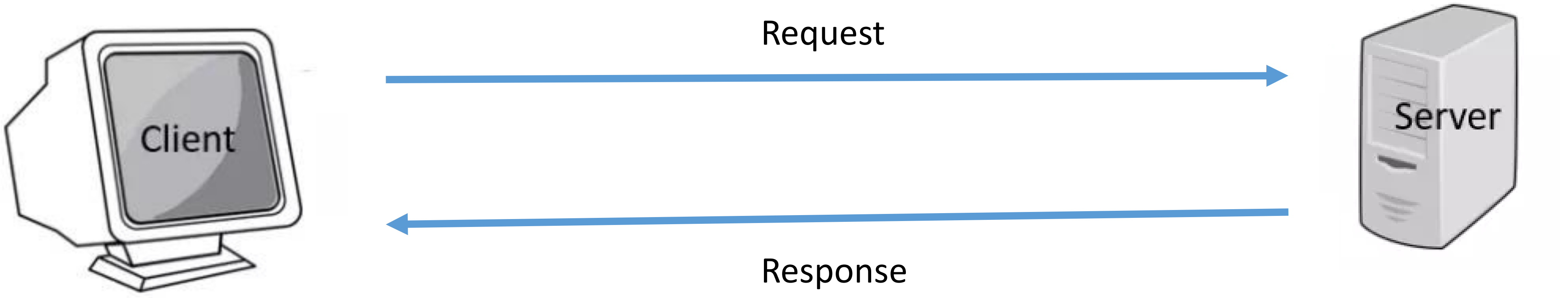
```
S: 220 smtp.example.com ESMTP Postfix
C: HELO relay.example.com
S: 250 smtp.example.com, I am glad to meet you
C: MAIL FROM:<bob@example.com>
S: 250 Ok
C: RCPT TO:<alice@example.com>
S: 250 Ok
C: RCPT TO:<theboss@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: From: "Bob Example" <bob@example.com>
C: To: Alice Example <alice@example.com>
C: Cc: theboss@example.com
C: Date: Tue, 15 Jan 2008 16:02:43 -0500
C: Subject: Test message
C:
C: Hello Alice.
C: This is a test message with 5 header fields and 4 lines in the message body.
C: Your friend,
C: Bob
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye
{The server closes the connection}
```

HTTP

- The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.
- HTTP is the foundation of data communication for the World Wide Web.

- The client submits an HTTP *request* message to the server. The server, which provides *resources* such as webpages and other content, or performs other functions on behalf of the client returns a *response* message to the client.

- The response message contains HTML code for the browser to display.



Sample HTTP request

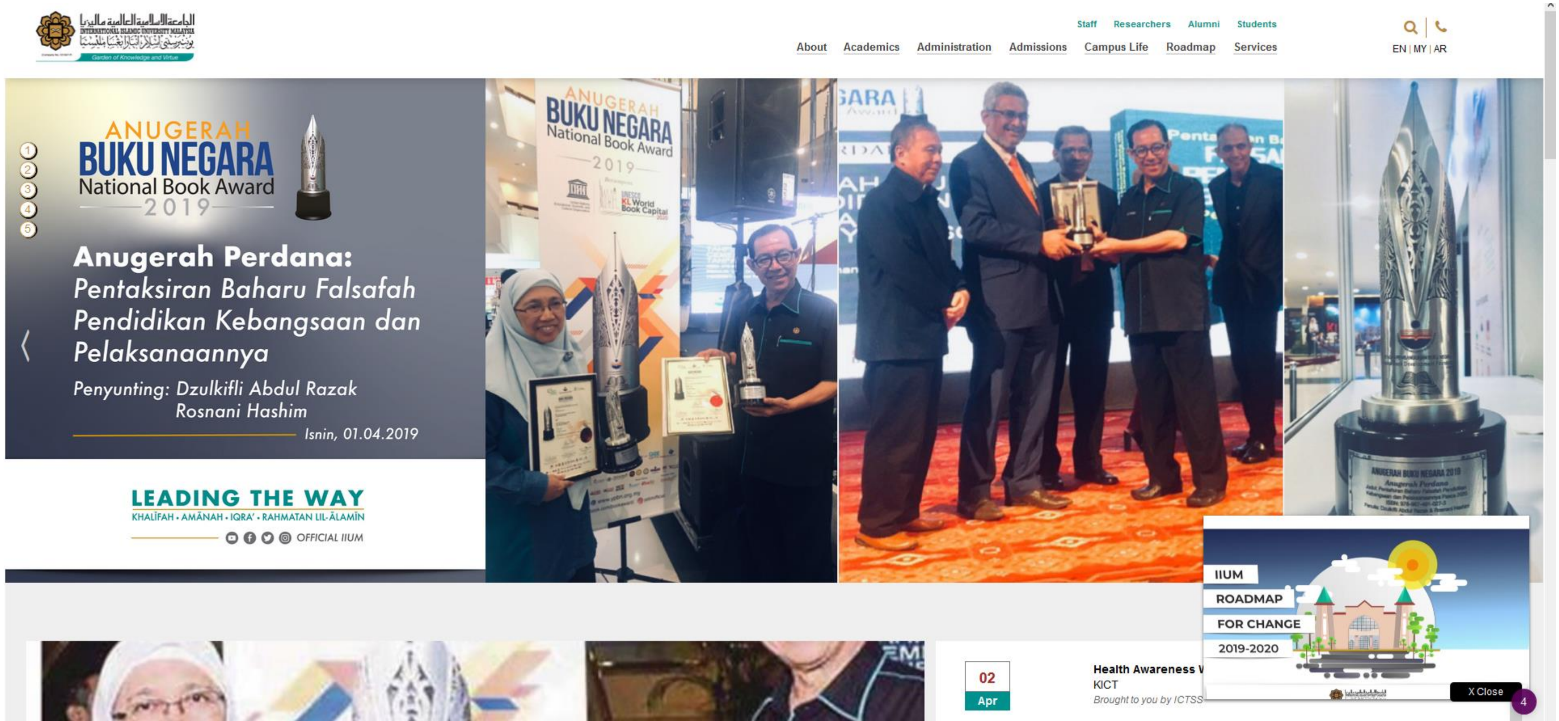
```
GET / HTTP/2.0                         //get the home page of the specified site using http 2.0 protocol
Host: www.iium.edu.my                  //the website address
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:66.0) Gecko/20100101 Firefox/66.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

Sample HTTP response

```
HTTP/2.0 200 OK                        //200 is code for OK
date: Mon, 08 Apr 2019 10:33:02 GMT  //Server date
expires: -1
cache-control: private, max-age=0
content-type: text/html; charset=UTF-8
strict-transport-security: max-age=31536000
content-encoding: br
server: gws
content-length: 57435              //the length of the HTML message in byte
                                   //Blank line
<!DOCTYPE html>                  //the HTML webpage starts
<html lang="en" dir="ltr">
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
  <title>International Islamic University Malaysia</title>
  <meta name="viewport" content="width=device-width, initial-scale=1,……….
```

HTML

- Webpages are described in HTML. Upon receiving HTTP response, browser interprets the HTML and displays the page.



IP Address

For communication to work, a computer or device requires an address. The Internet uses two addressing systems:

IPv4

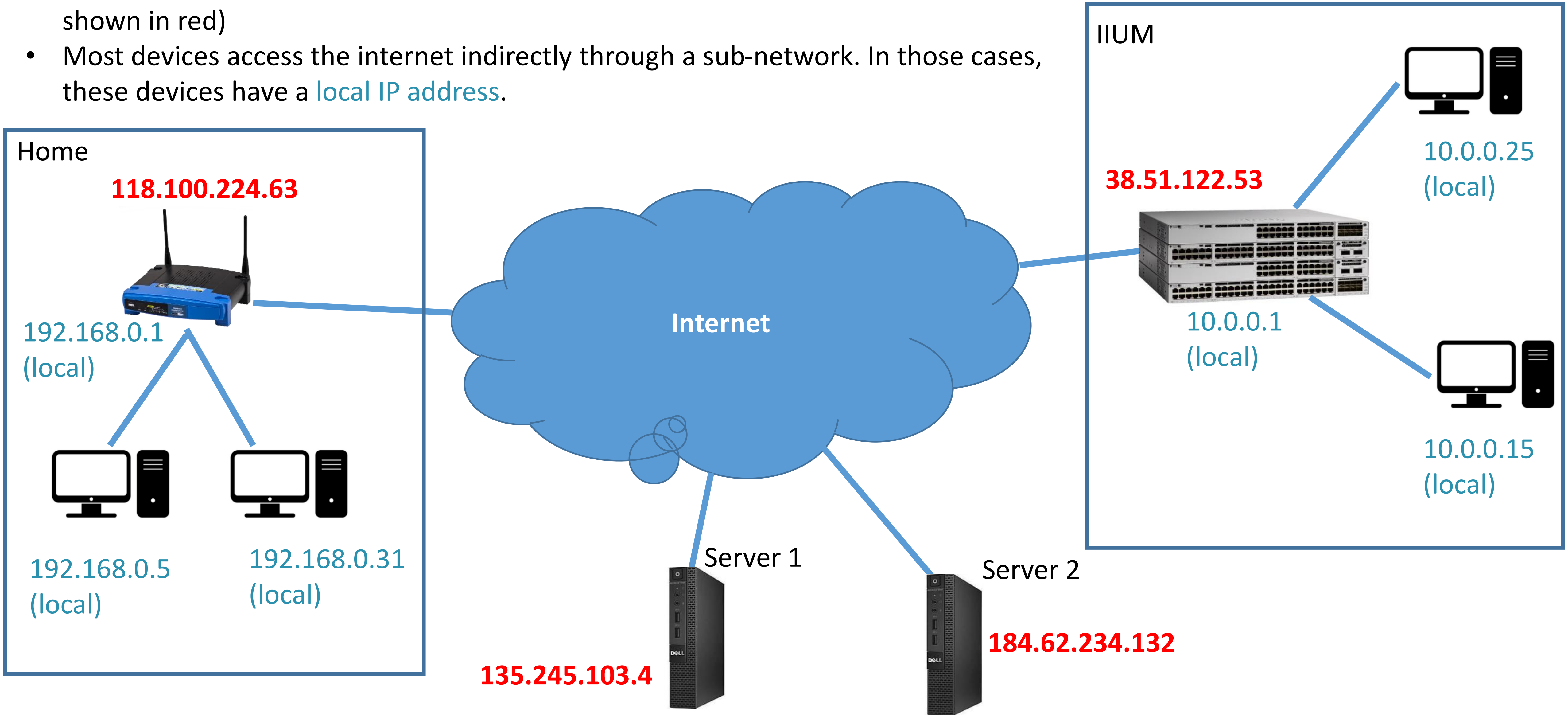
Currently the dominant addressing system. IPv4 addresses are 32 bits wide. When string-formatted, IPv4 addresses are written as four dot-separated decimals (e.g., 101.102.103.104). An address can be unique in the world or unique within a particular subnet (such as on a corporate network).

IPv6

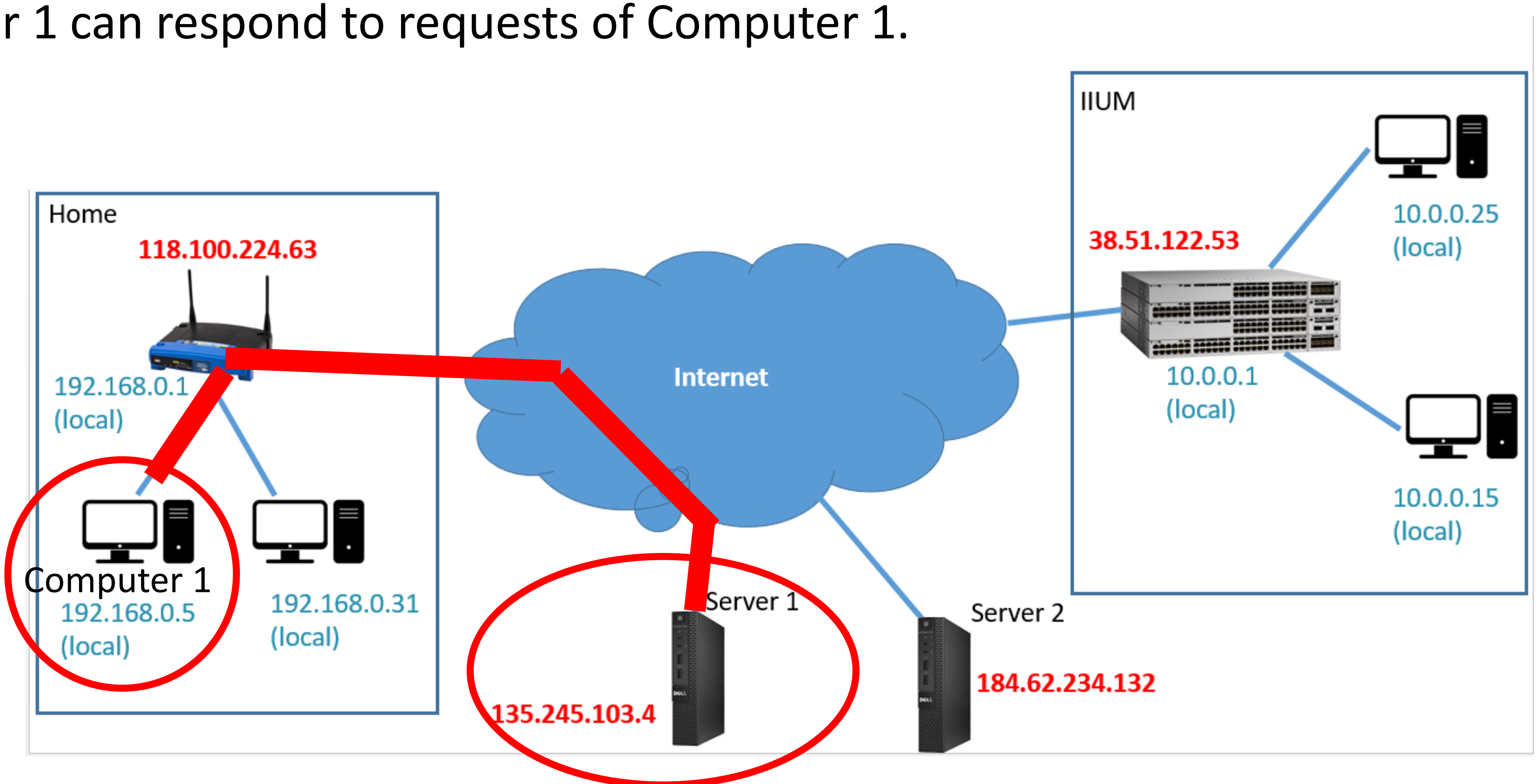
The newer 128-bit addressing system. Addresses are string-formatted in hexadecimal with a colon separator (e.g.,[3EA0:FFFF:198A:E4A3:4FF2:54fA:41BC:8D31]).

IP address

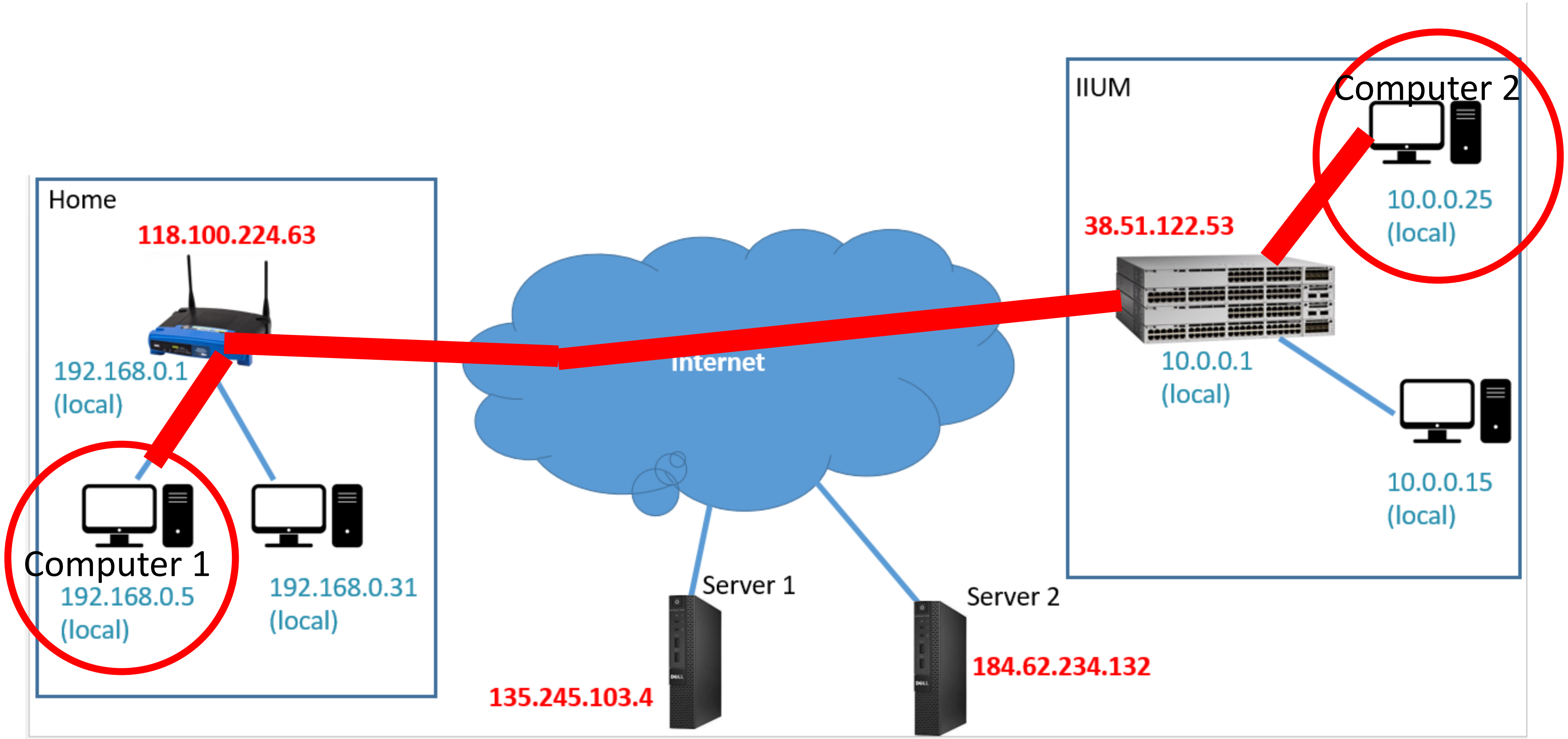
- Every device directly connected to the Internet should have a unique **IP address** (as shown in red)
- Most devices access the internet indirectly through a sub-network. In those cases, these devices have a local IP address.



- In the example below, Computer 1 can reach Server 1 through the home router.
- But Server 1 cannot reach Computer 1, except if the router is specifically configured to do so through "port forwarding".
- But Server 1 can respond to requests of Computer 1.

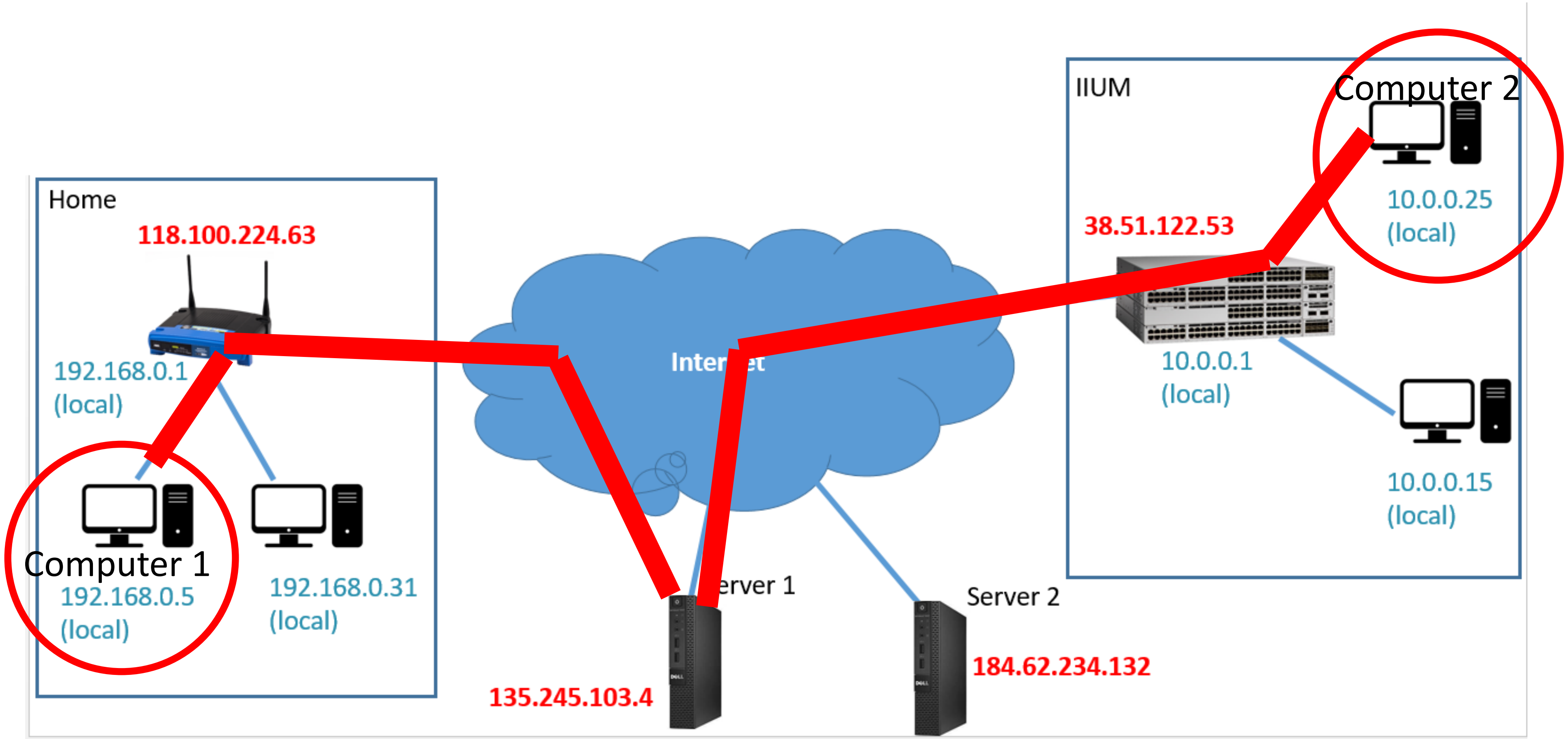


- Similarly, computer 1 and computer 2 cannot independently talk to each other.
- In fact, they may be even we aware of the presence of each other.



- However, computer 1 and computer 2 can talk to each other through Server 1 if server 1 relays their messages.

- E.g. chatting server (like Whatsapp)



IPAddress class

The IPAddress class in the *System.Net* namespace represents an address in either protocol. It has a constructor accepting a byte array, and a static Parse method accepting a correctly formatted string:

```
IPAddress a1 = new IPAddress(new byte[] { 101, 102, 103, 104 });
IPAddress a2 = IPAddress.Parse("101.102.103.104");
Console.WriteLine(a1.Equals(a2)); // True
Console.WriteLine(a1.AddressFamily); // InterNetwork


IPAddress a3 = IPAddress.Parse("[3EA0:FFFF:198A:E4A3:4FF2:54fA:41BC:8D31]");
Console.WriteLine(a3.AddressFamily); // InterNetworkV6
```

Port number

- The TCP and UDP protocols break out each IP address into 65,535 ports, allowing a computer on a single address to run multiple applications, each on its own port.

- Many applications have standard port assignments; for instance, HTTP uses port 80; SMTP uses port 25.

If we combine an IP address and a port number, we have a complete address called end point.

```
IPAddress a = IPAddress.Parse("101.102.103.104");
IPEndPoint ep = new IPEndPoint(a, 222); // Port 222

Console.WriteLine(ep.ToString()); // 101.102.103.104:222
```

DNS

- Since IP addresses are difficult to remember, human-friendly domain names are used.
- Domain name servers (DNS) serves like a as the phone book for the Internet by translating human-friendly computer hostnames into IP addresses
- For example, the domain name www.example.com translates to the addresses *93.184.216.34.*

URL

- A Uniform Resource Locator (URL), colloquially termed a web address is a reference to a web resource that specifies its location on a computer network and a mechanism for retrieving it.

- A typical URL could have the form http://www.example.com/index.html

URI

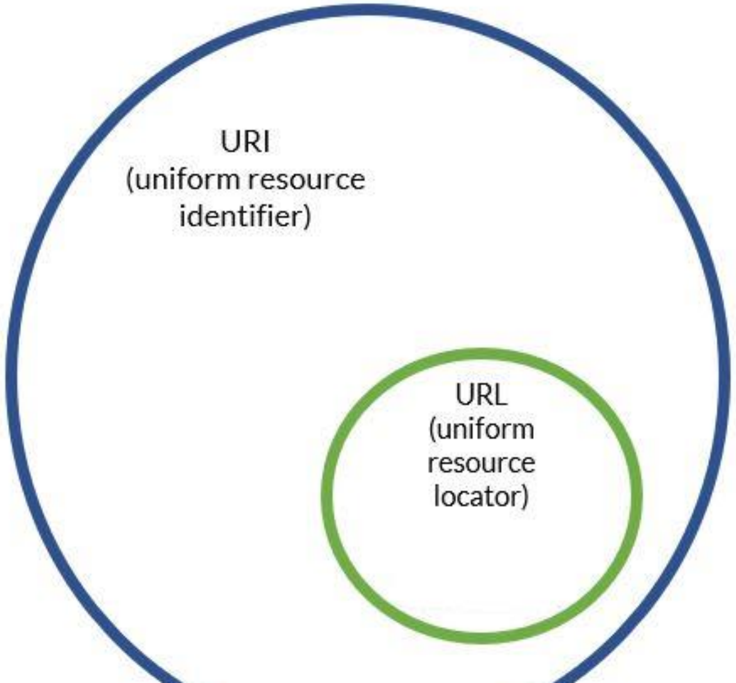
- URI is a generalization of URL to cover resources on a local computer, mail addresses and other protocols.
- The exact formatting is defined by the Internet Engineering Task Force.
- Examples

  http://www.ietf.org

  ftp://myisp/doc.txt

  mailto:joe@bloggs.com

  file:///home/user/file.txt



```csharp
Uri info = new Uri("http://www.domain.com:80/info/");
Uri page = new Uri("http://www.domain.com/info/page.html");

Console.WriteLine(info.Host); // www.domain.com
Console.WriteLine(info.Port); // 80
Console.WriteLine(page.Port); // 80 (Uri knows the default HTTP port)
Console.WriteLine(info.IsBaseOf(page)); // True

Uri relative = info.MakeRelativeUri(page);
Console.WriteLine(relative.IsAbsoluteUri); // False
Console.WriteLine(relative.ToString()); // page.html
```

TCP/UDP

- TCP and UDP constitute the transport layer protocols on top of which most Internet—and local area network—services are built.

- **TCP** is connection-oriented and includes reliability mechanisms. HTTP, FTP, and SMTP use TCP.
- **UDP** is connectionless, has a lower overhead, and supports broadcasting. DNS and *BitTorrent use UDP.*

TcpListener and TcpClient class

The TcpListener class stored under the *System.Net.Sockets* namespace provides simple methods that listen for and accept incoming connection requests in blocking synchronous mode.

The TcpClient class stored under the *System.Net.Sockets* namespace provides simple methods that initiates connection to a TcpListener.

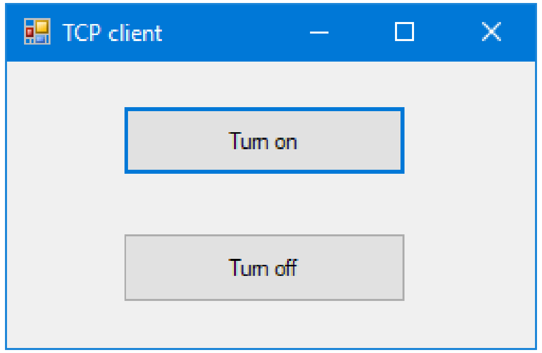
Example

Develop software for client and server where the client can remotely turn ON or OFF the LED attached to the server.

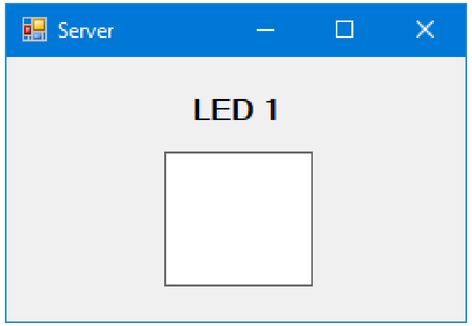
Use the following custom protocol

- Client should send 1 to the server to turn on the LED.
- Client should send 0 to the server to turn off the LED.
- Server should reply 101 if the requested task has been carried out successfully.

Client



Server


Server
LED 1