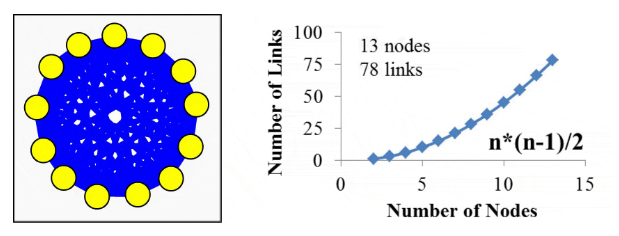
**C16 Internet of Things**

**C16.1 The Internet in a Nutshell**

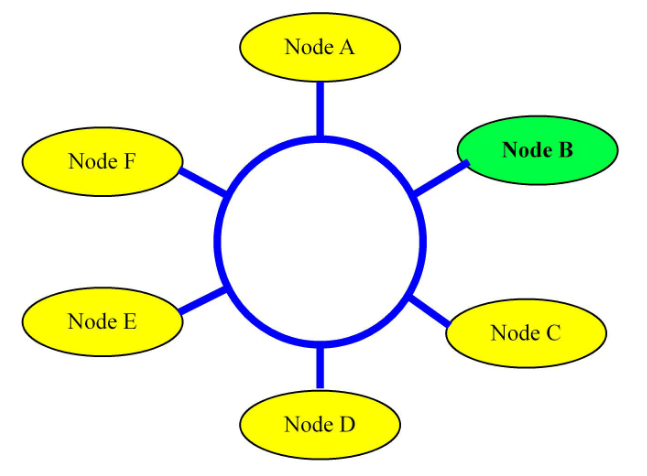
Cable Communication

Communication initially was with cables, however, as we increase the number of nodes that wish to communicate, we also increase the number of cables. The number of cables grows on the order of :



Token Ring

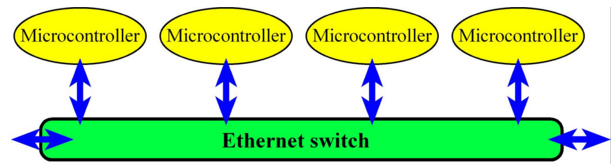
Next, came the token ring method. Where the node must hold a token to communicate:



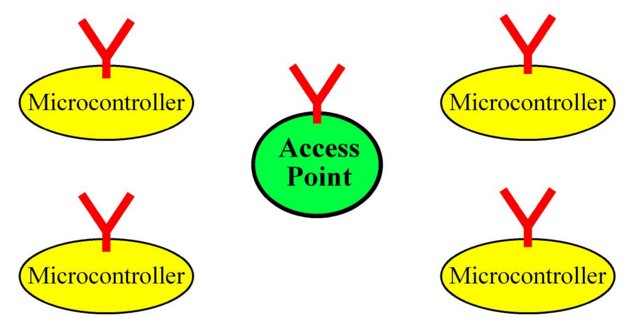
The issue here is that each node would have to wait for the other nodes as the token was passed around in a circular fashion.

CSMA Carrier Sense Multiple Access-Collision Detection

Next came CSMA. In this protocol, the microcontroller would sense to see if the network is idle when it wishes to communicate.



CSMA is used for wireless communication as well. Here, microwave transmission is used instead of a cable and communication is in 2.4, 3.6, 5 and 60 GHz frequency bands.

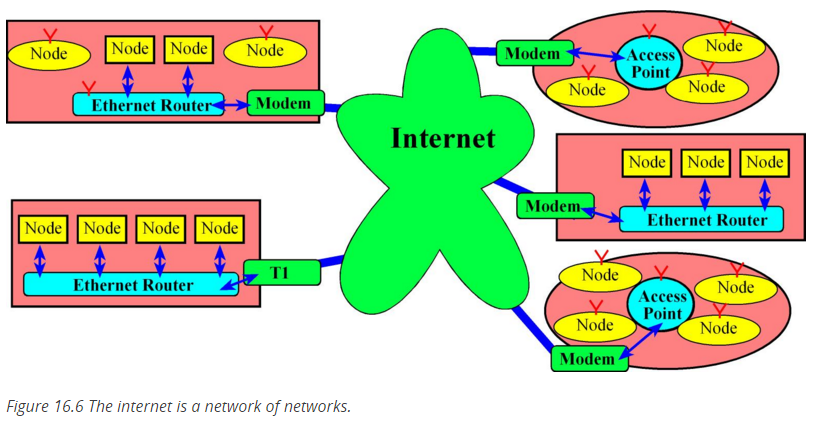


Internet

Now that we have figured out how to configure machines to communicate with each other in a small local area network (LAN), we need to figure how to get multiple LANs to communicate with each other. This interconnection is called the Internet. The collective issues concerning the interconnection of networks are addressed by what is called the Network layer.

First issue is one of identities. Machines must be given an identity – IP address. But having just numbers can be complex. Hence we have a lookup service called Domain Name Service (DNS) that maps symbolic names such as [www.google.com](http://www.google.com) to its corresponding IP address.

Second issue is routing – the movement of info from one machine to another in an optimal way.

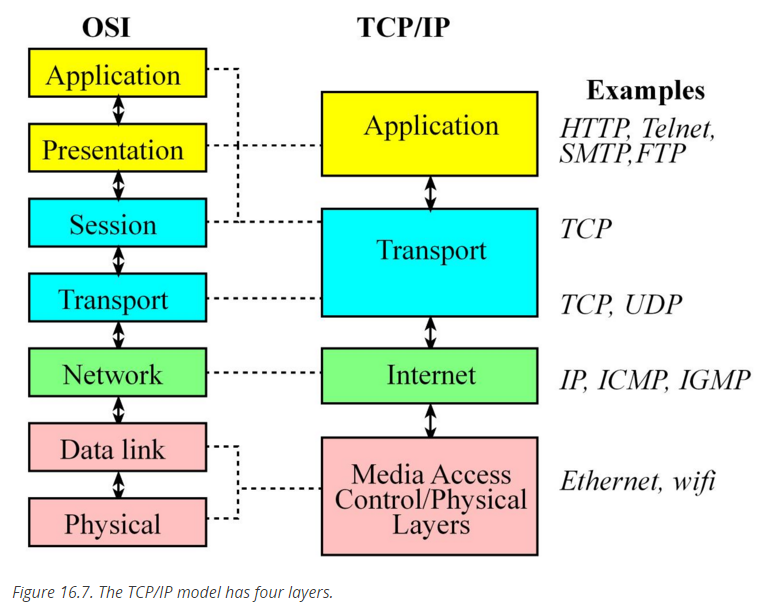


Each node or machine has a port which it can send and receive messages with. These ports fall in two categories, TCP ports and UDP ports. The Transport layer is responsible for the abstraction of ports.

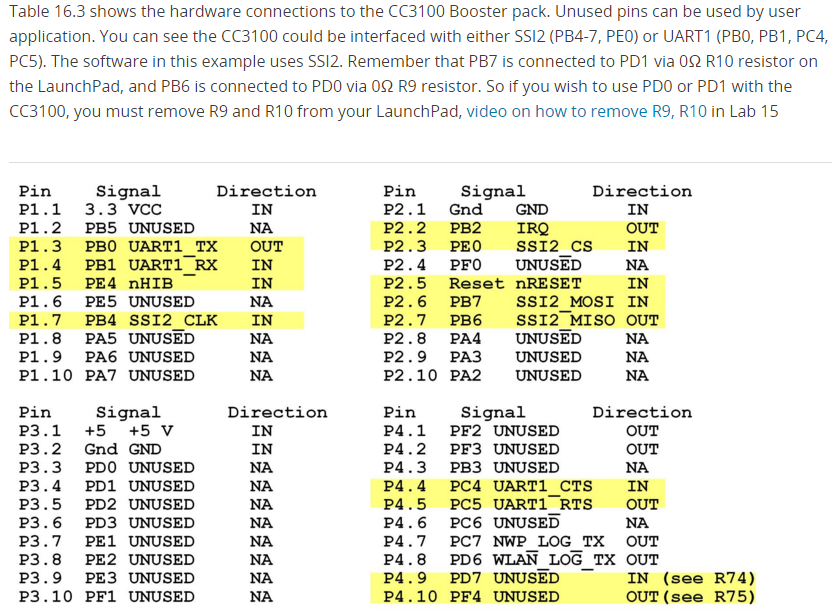
A socket captures the end point of communication which can be seen as containing, a port number, its type (TCP/UDP) and the IP address. Programming in the Internet is nothing but using the socket API. Socket API is very low level, hence one uses higher level Application layer protocols like HTTP, SMTP, and FTP (hypertext transfer, mail-transfer, file-transfer).

There is also a Media Access layer for multimedia applications like VoIP.

Thus, the layered model that the internet uses can be seen as having four layers: the Application layer, the Transport layer, the Network layer and the Media Access Layer.

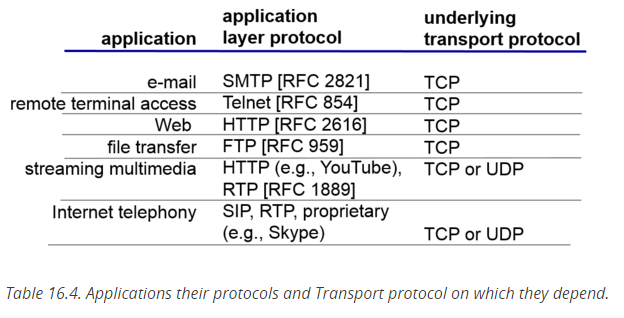


**C16.4 CC3100 Booster Pack**



**C16.5 Client Server Communication**

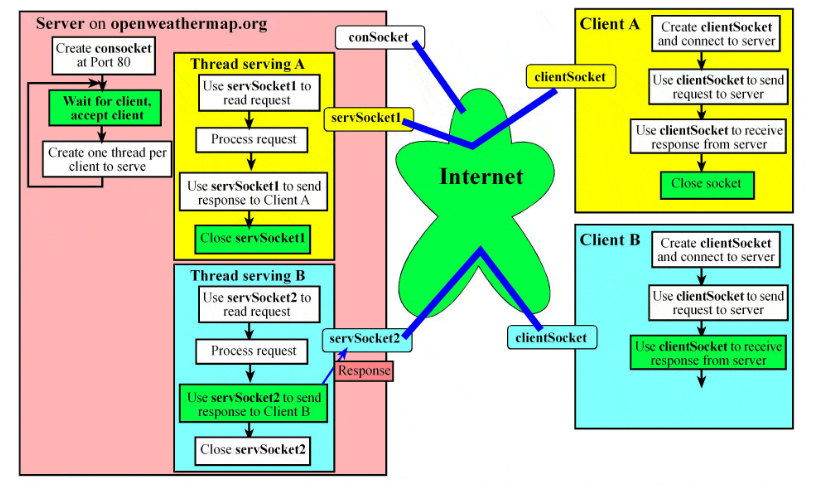
The client-server paradigm is the dominant communication pattern over the internet and has two forms, a TCP-based connection oriented form and a UDP-based connection-less form.



Streaming multimedia and internet telephony can use UDP because it does not care about reliability and having a few packets dropped. A Youtube video missing a few frames is okay.

A server running at a machine creates a socket bound to port xyz. The port number is either a standard port or one chosen specifically for the service. For example, if the server is a web server that is publicly accessible, then the port number is 80. This socket is called a connection socket (**conSocket**) where the servers waits for connection requests from clients. A client that wants to access the service provided by the server creates a client socket (**clientSocket**). The clients needs to know the hostname and the port at which the service is running to establish the connection using the client socket. Once the server sees the request, it accepts it on a separate socket (**servSocket**). It does not serve the client on the connection socket because, if it takes too long to serve the client then the server will be inaccessible to other clients seeking service. By accepting clients on a separate client, the server can continue to accept requests from new clients.

Once the server accepts a client, it creates a thread to service that client by passing the thread the socket at which the client was accepted.



**C16.6 Communication with a Weather Server**