TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

TEXAS INSTRUMENTS

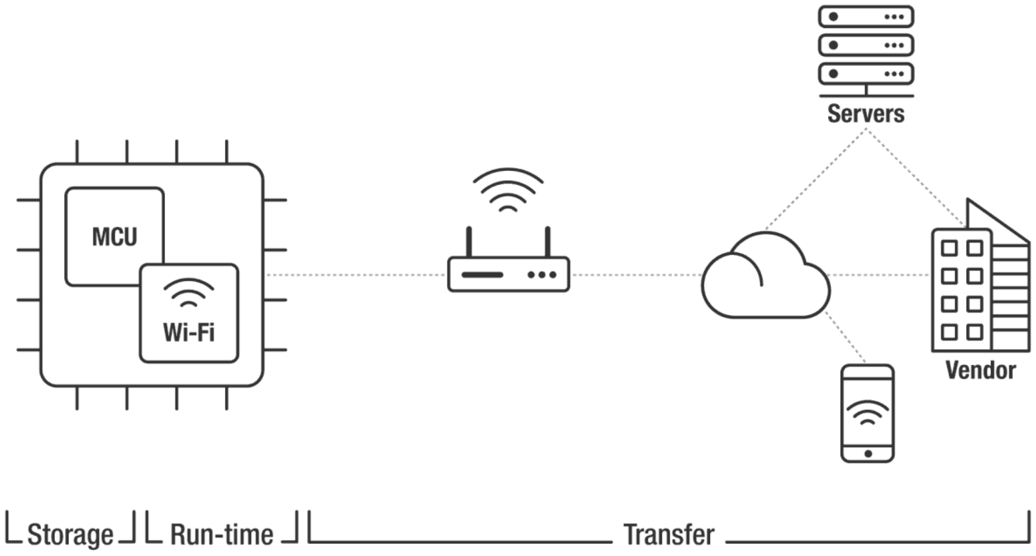
Module 20

Lecture: Internet of Things

Internet of Things

You will learn in this module

- Basic approach to the internet of things
- TCP/IP Transport Layer
- Domain Name Service
- Client-server Paradigm

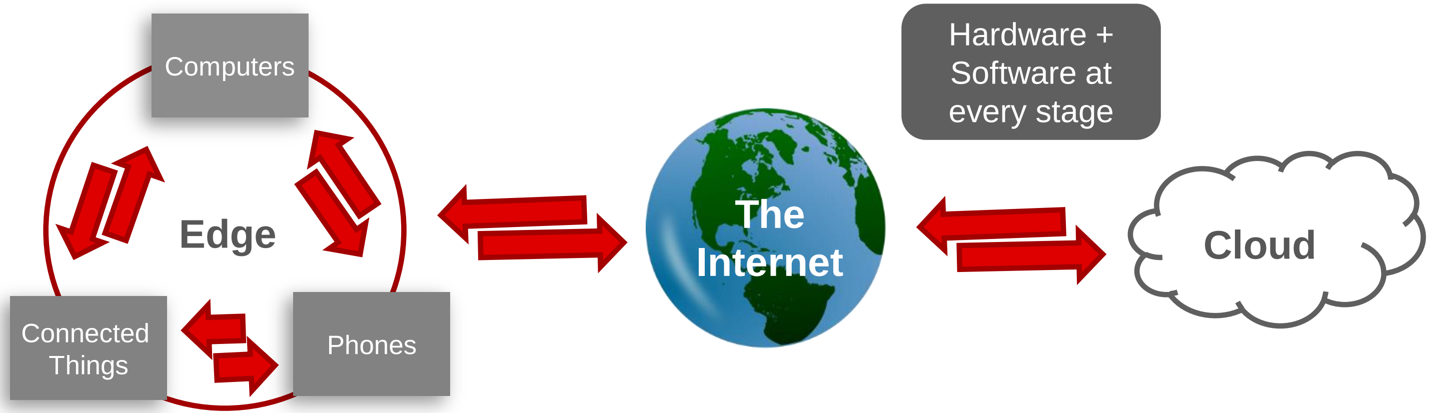


The Internet of Things; challenges

- Standardization
  - **SimpleLink™** implements a light-weight stack
- Interoperability
  - Technologies, vendors, companies
- Evolution
  - Incremental/continuous vs revolutionary
- Stability
- Abstraction
- Scalability
  - 50 million to 50 billion
- Security
  - Confidentiality, integrity, availability

The Internet of Things: A bird's-eye view

IoT Data passes from physical hardware layers to software layers back and forth, connecting the real and digital worlds



Think products
- Anything that can talk to the internet or connect to something else that can talk to the internet

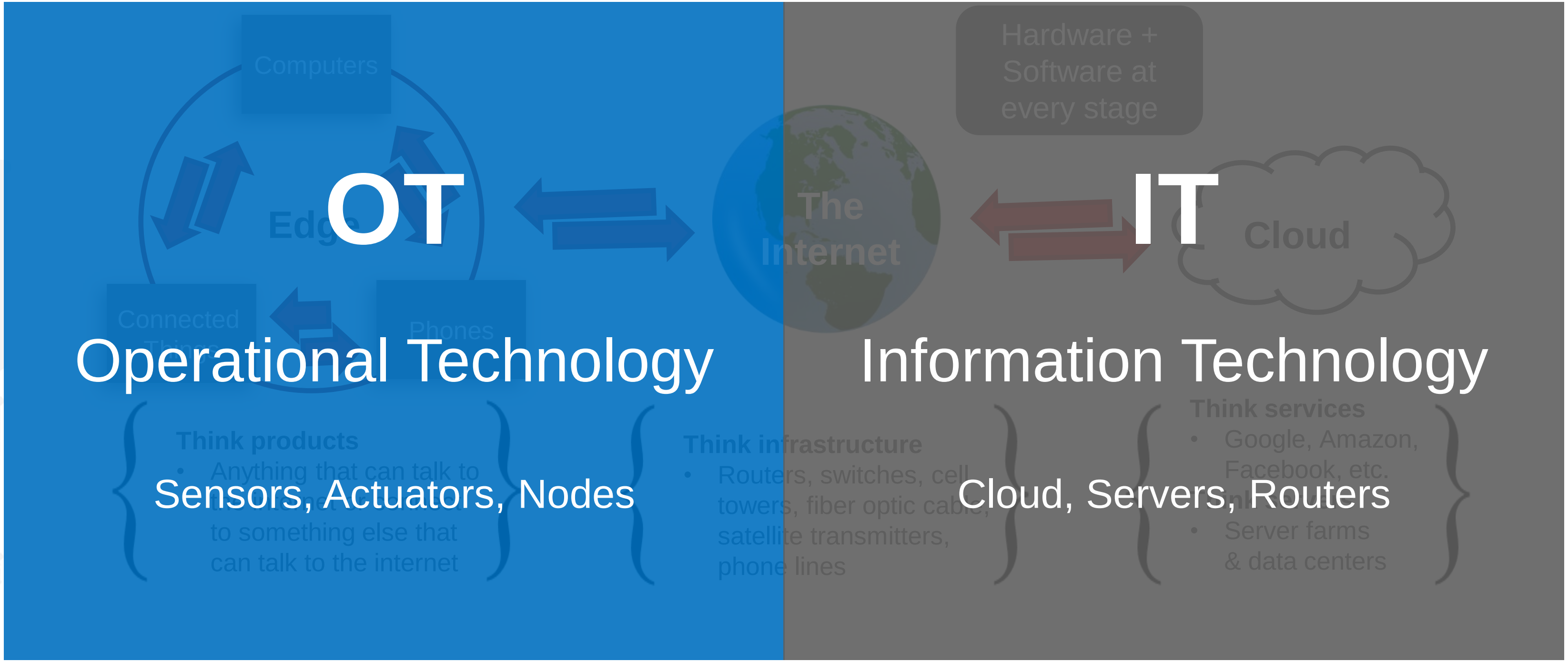
Think infrastructure
- Routers, switches, cell towers, fiber optic cable, satellite transmitters, phone lines

Think services
- Google, Amazon, Facebook, etc.

Think servers
- Server farms & data centers

The Internet of Things: A bird’s-eye view

IoT Data passes from physical hardware layers to software layers back and forth, connecting the real and digital worlds



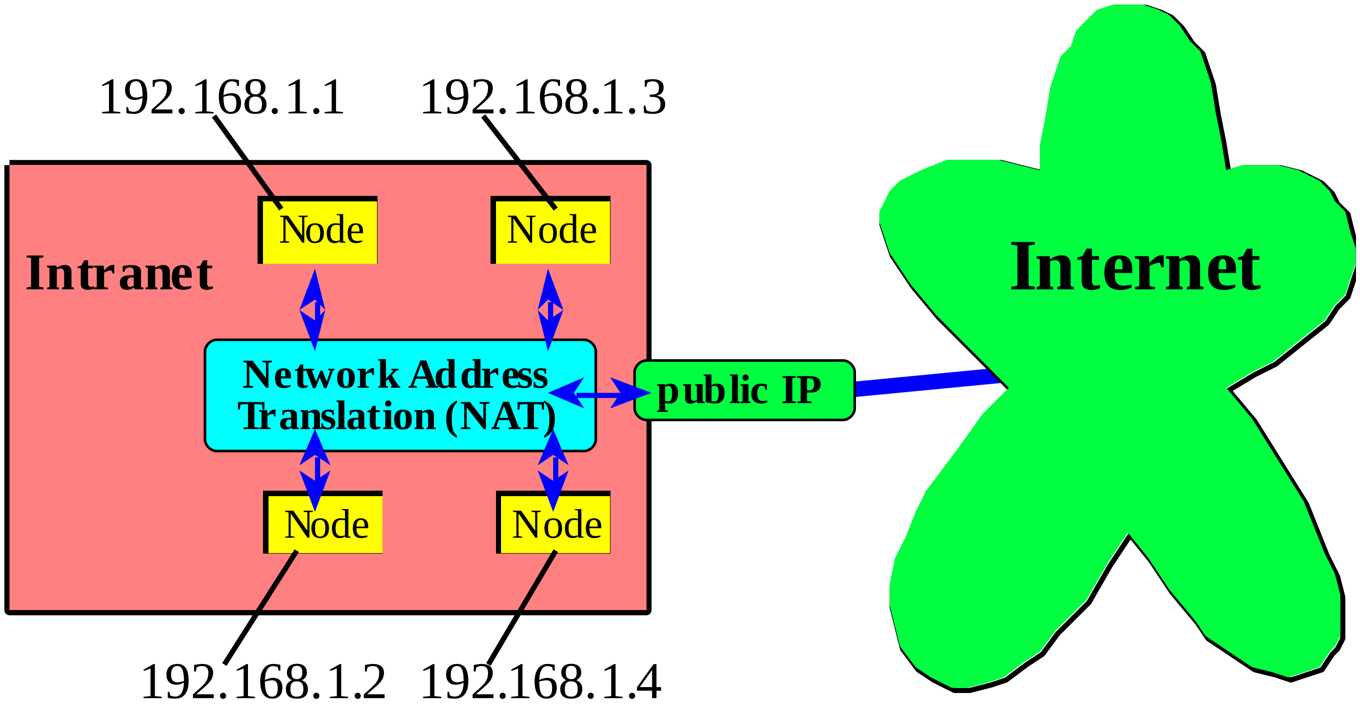
The Wi-Fi standard

Pros
- Ubiquitous infrastructure
- Direct connection to Internet
- Access a wide variety of APIs directly
- Only requires domain expertise in internet and firmware
- High data rate
- Security

Cons
- Poor for mobile and rural use cases
- Higher power consumption relative to some wireless standards
- Heavily reliant on network availability

Wi-Fi Primary Use Cases
- Smart Home
- Industrial/Commercial
- Fixed position connectivity
- Medical

Domain Name Service (DNS)



TCP/IP

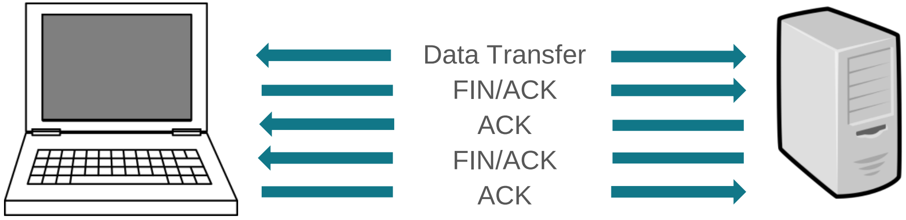
TCP is a more reliable way of Internet communication compared to UDP

- Transport Layer in the OSI model
- Use of Sockets

OSI Model

Application	Data
Presentation	Data
Session	Data
Transport	Segments
Network	Packets
Data Link	Frames
Physical	Bits

TCP Four Way Handshake



User A

Server B

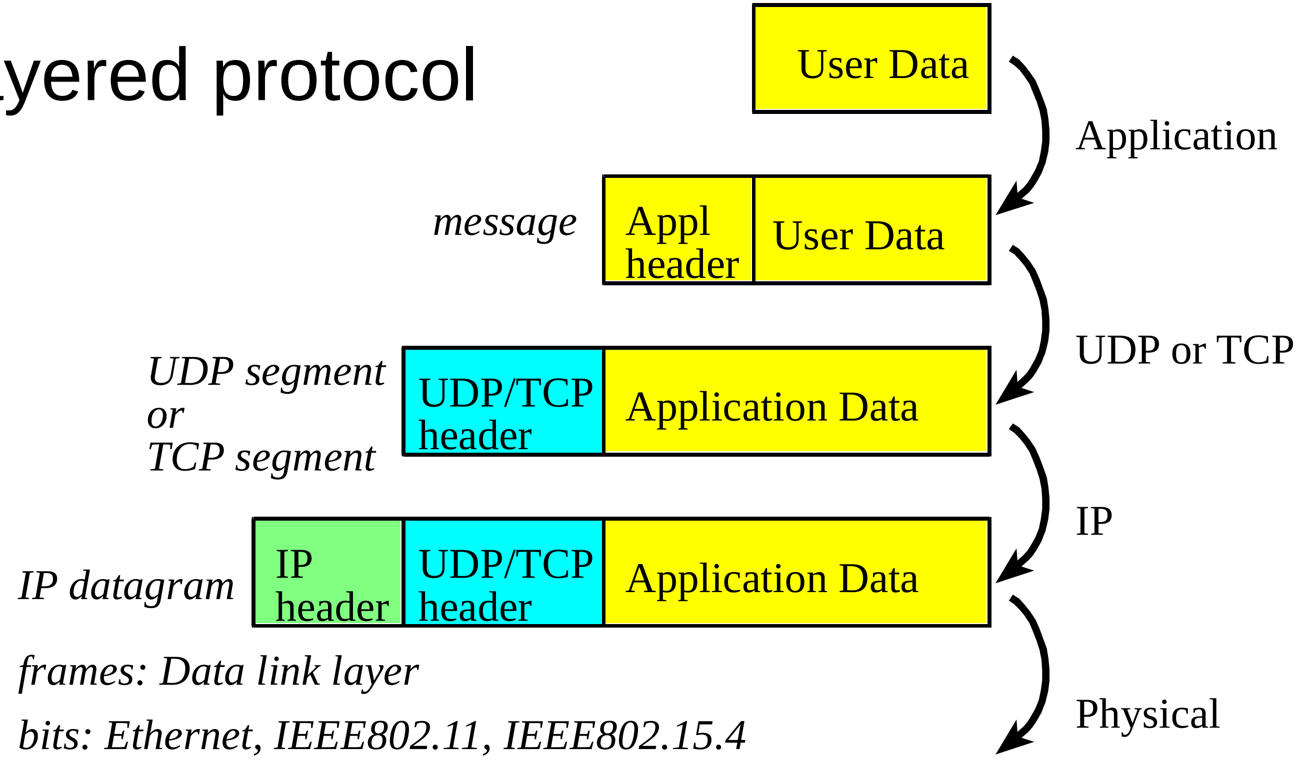
TCP Sockets

Client	Server
Socket()	Socket()
	Bind()
	Listen()
Connect()	Accept()
Send() Recv()	Send() Recv()
Close()	Close()

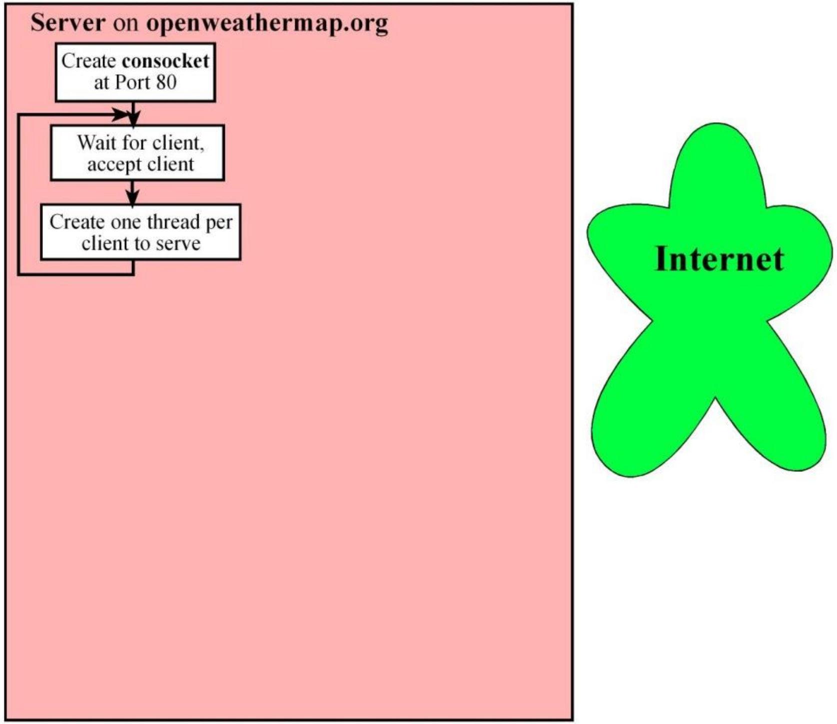
IP address + port number

Messages

Layered protocol

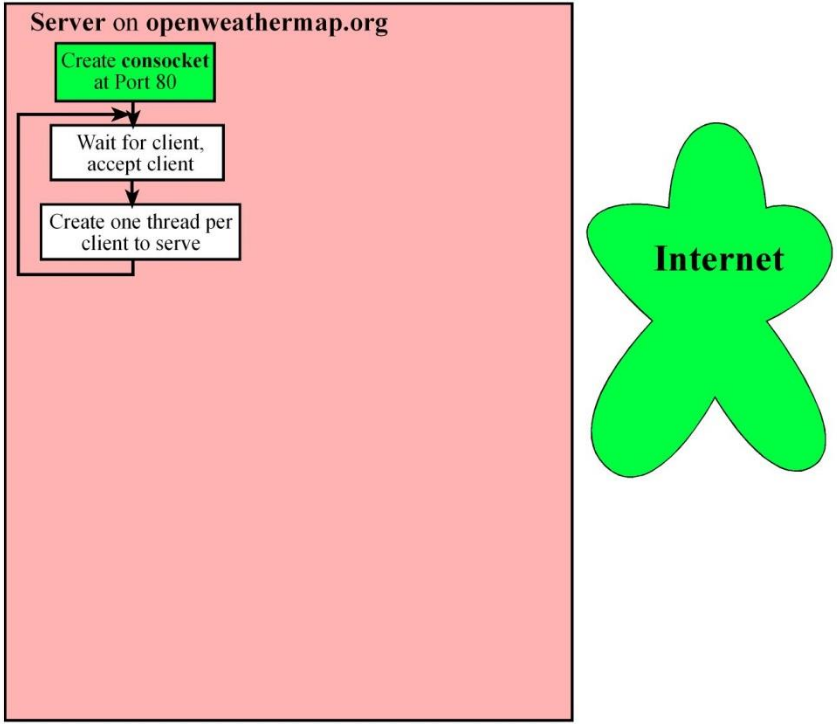


Client-server paradigm



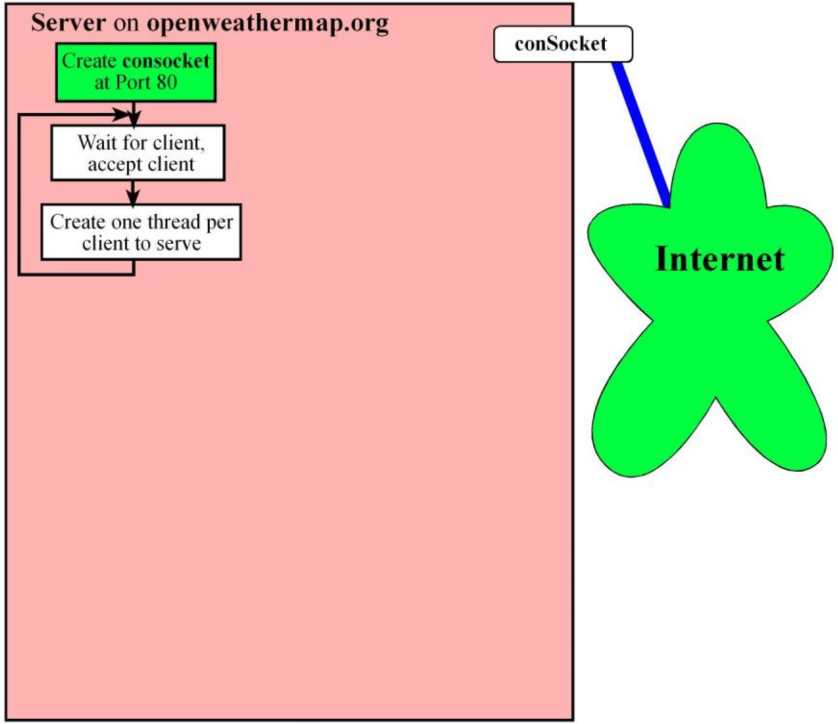
Let's walk through the steps where two clients are accessing the same server

Client-server paradigm



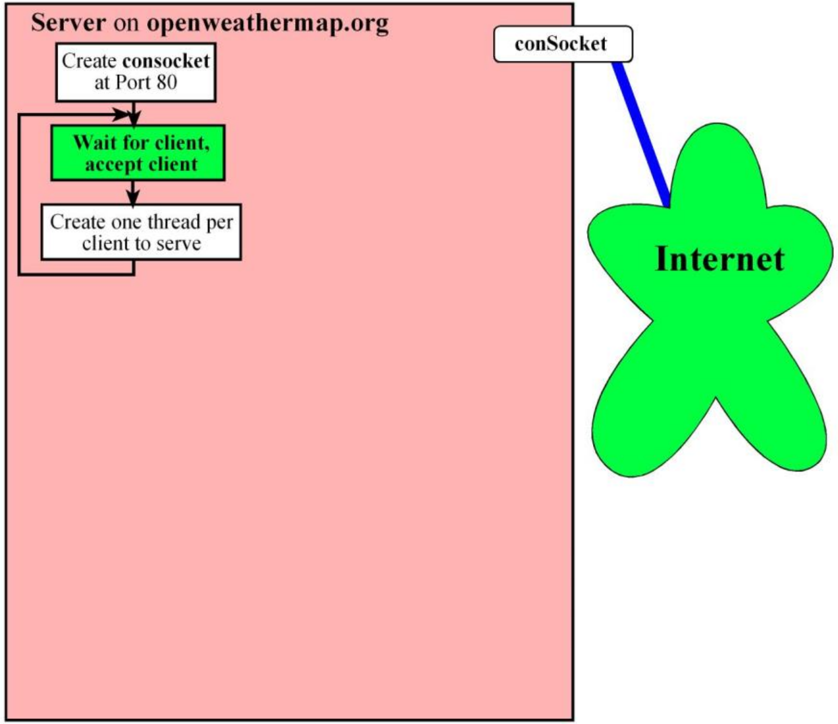
The server will create a connection socket, allowing requests from anywhere in the world

Client-server paradigm



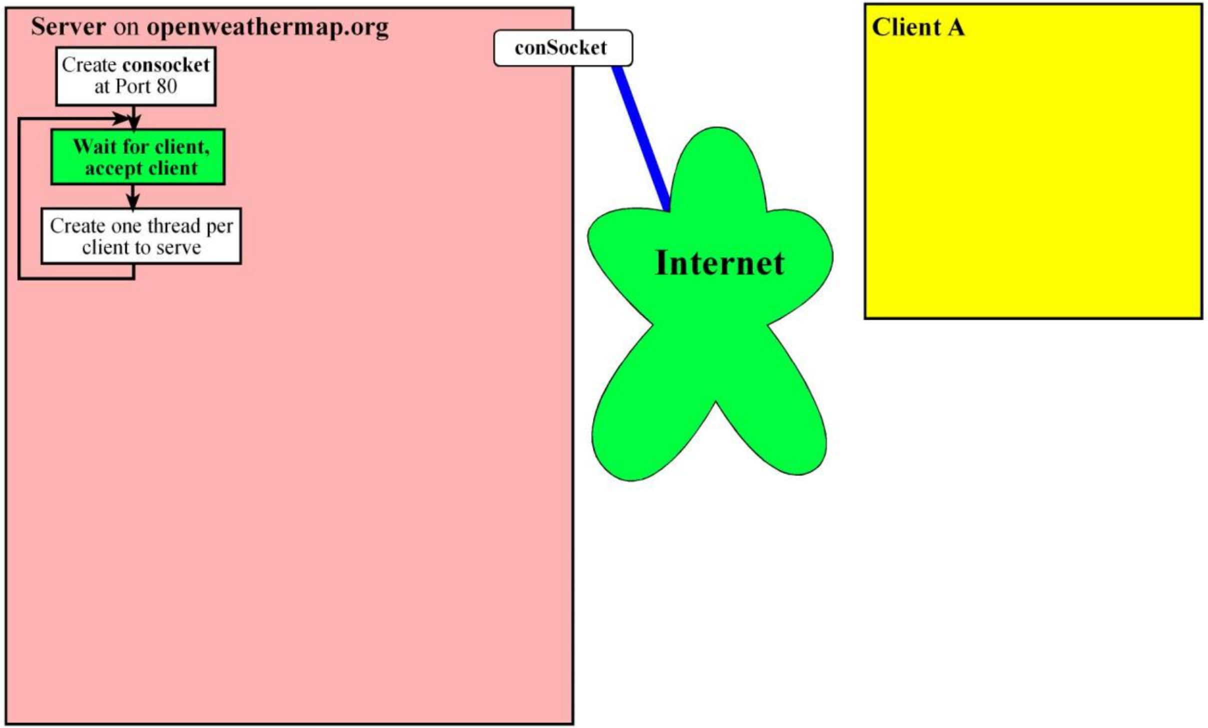
All one needs is the IP address and the port number to connect

Client-server paradigm



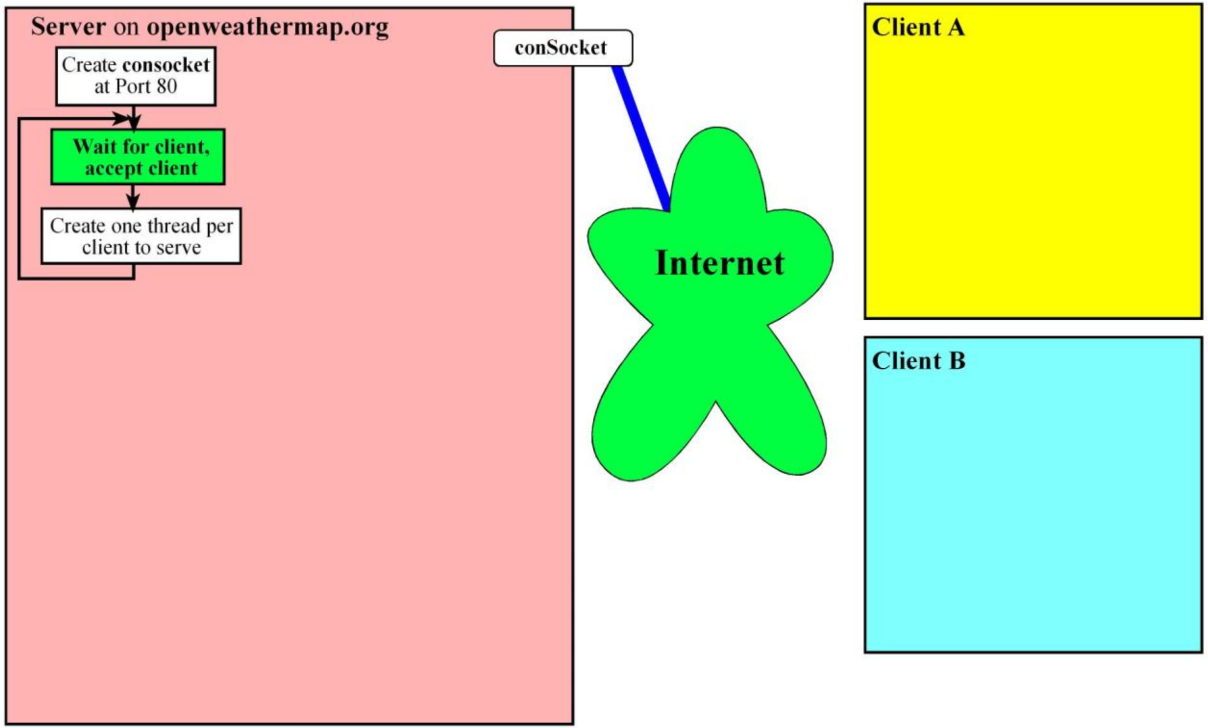
The server waits for clients to initiate (a server is the slave, and responds to client requests)

Client-server paradigm



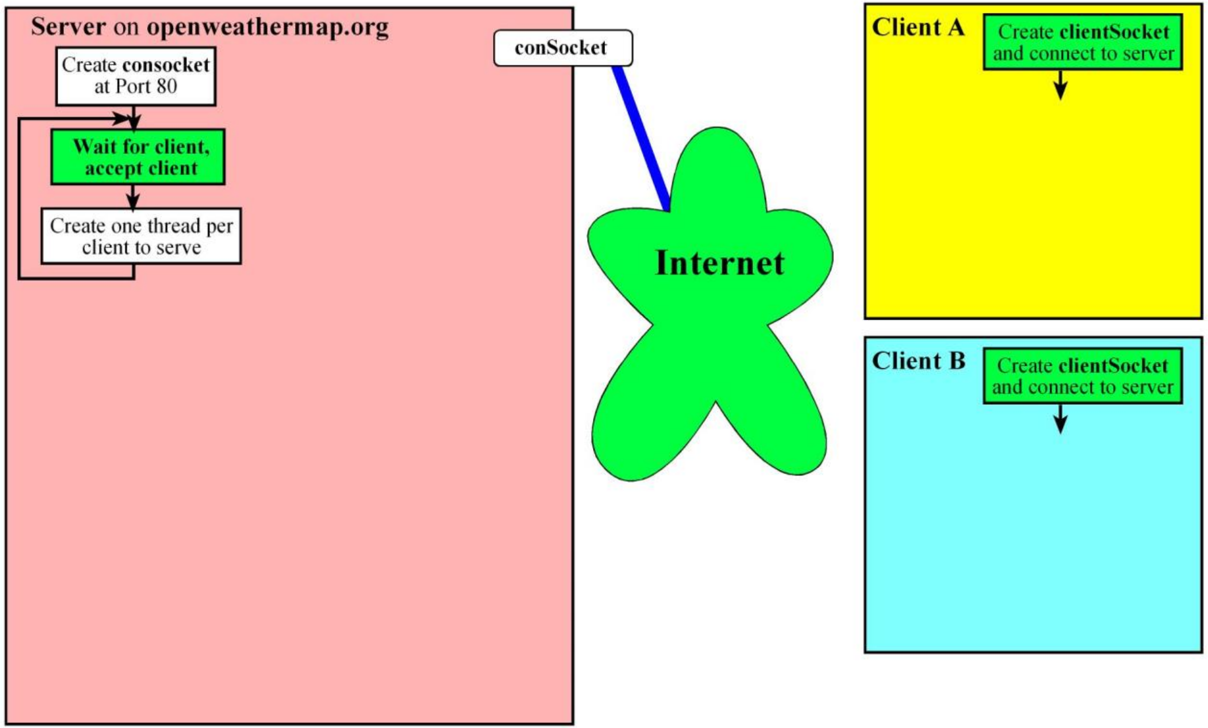
Let’s create a client (e.g., your RSLK robot)

Client-server paradigm



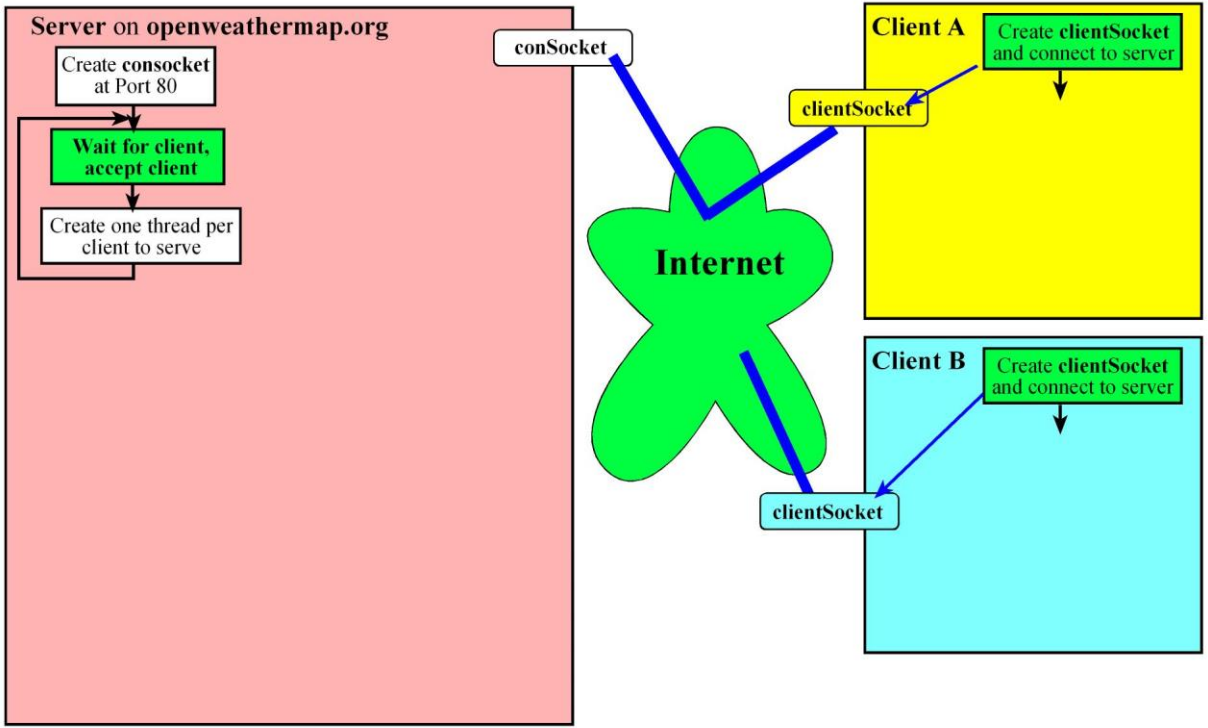
Let’s create a second client (there could be 0, 1 or 100,000 clients)

Client-server paradigm



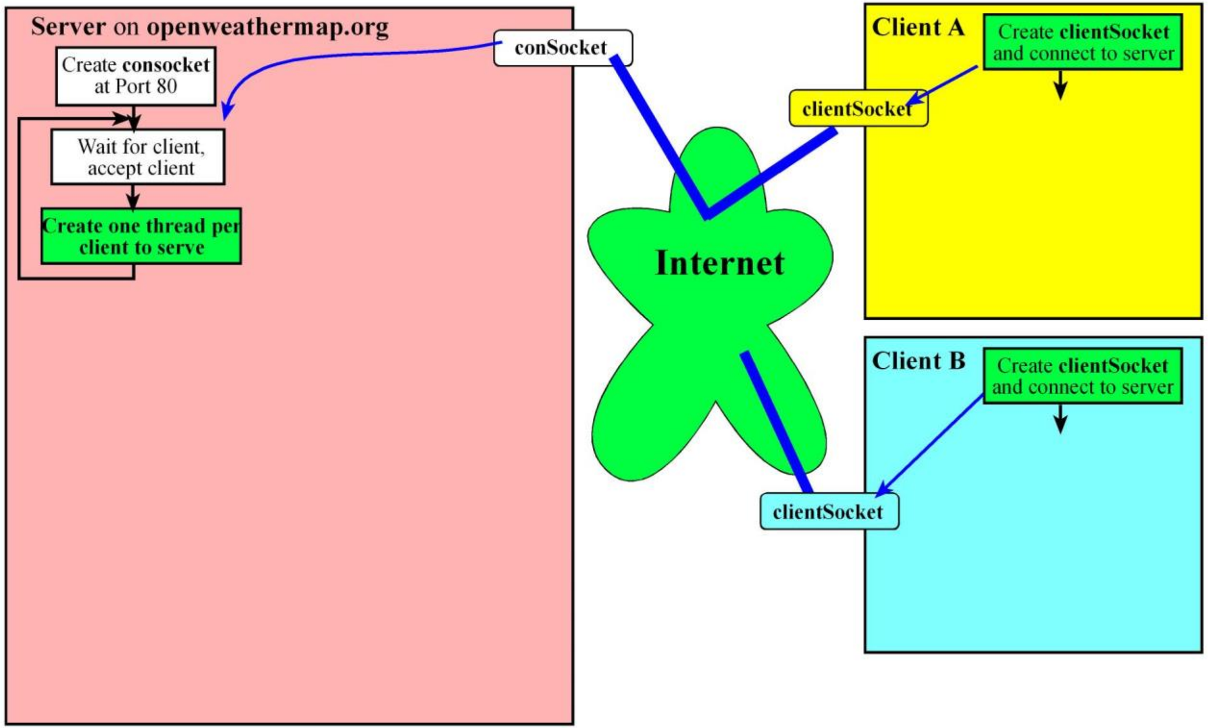
Sockets are software data structures through which communication occurs

Client-server paradigm



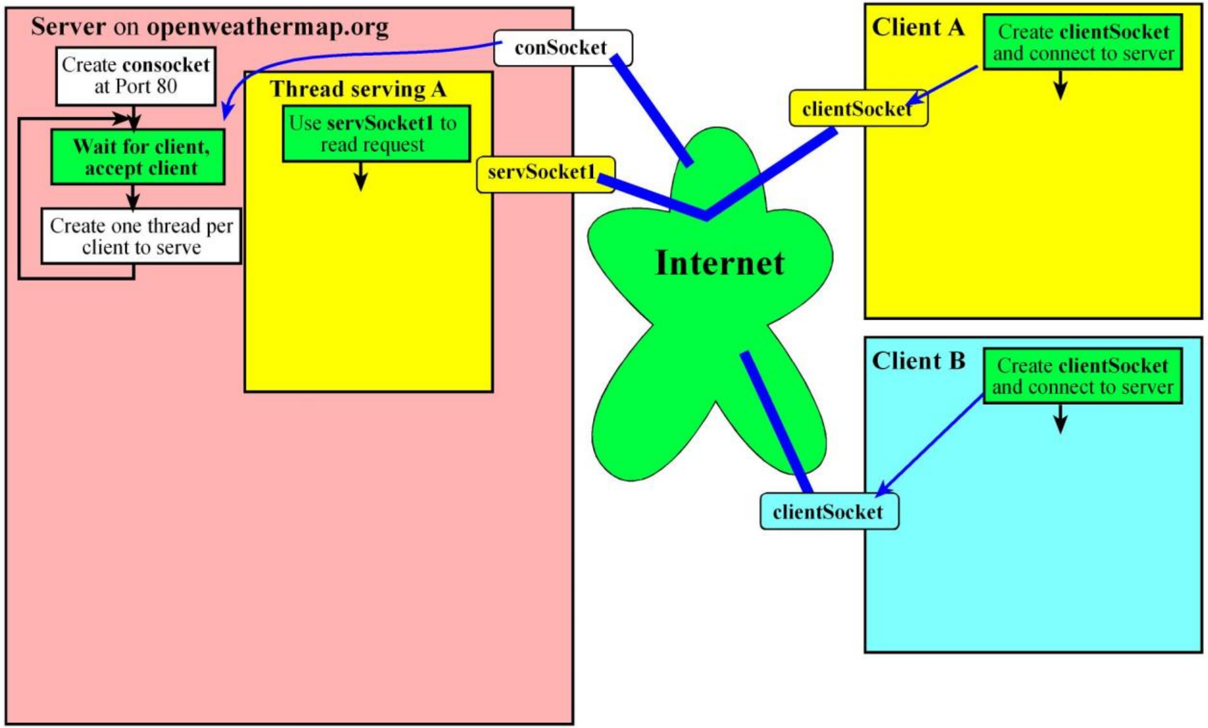
Through the clientSocket the client accesses the conSocket requesting a connection

Client-server paradigm



The server responds to the connection request by creating a unique thread to service that client

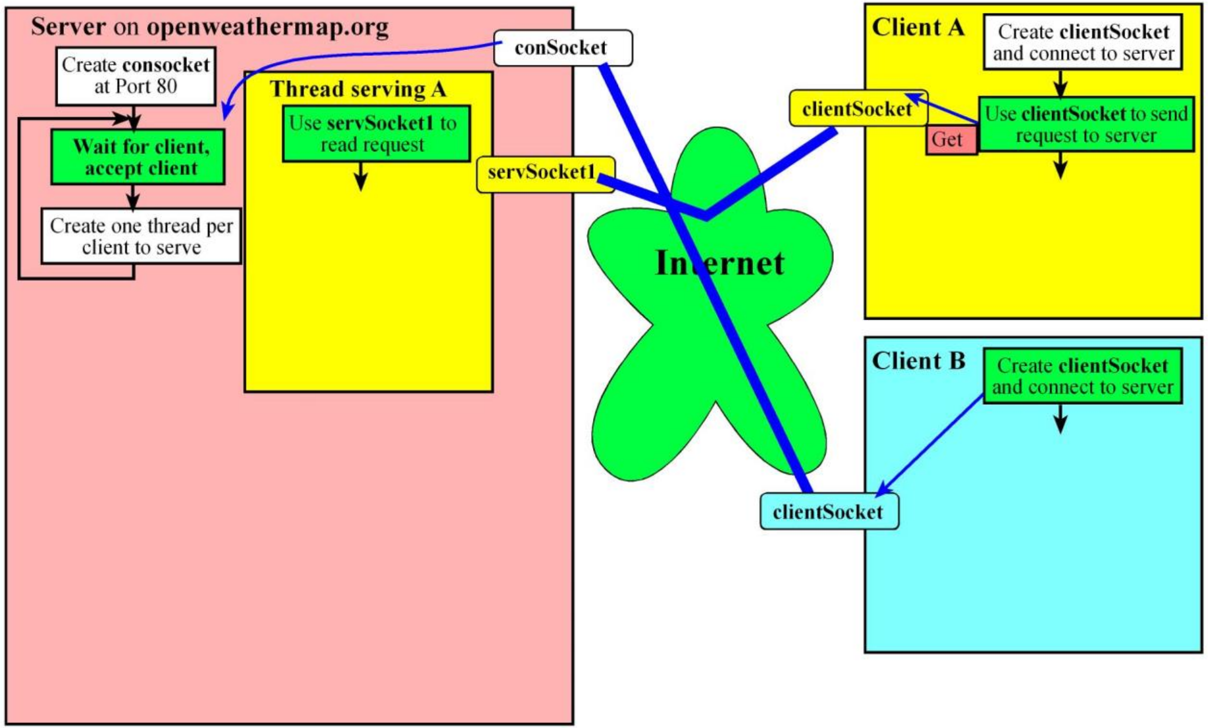
Client-server paradigm



The server will create a unique server socket for each client, and respond to client its existence

Client-server paradigm

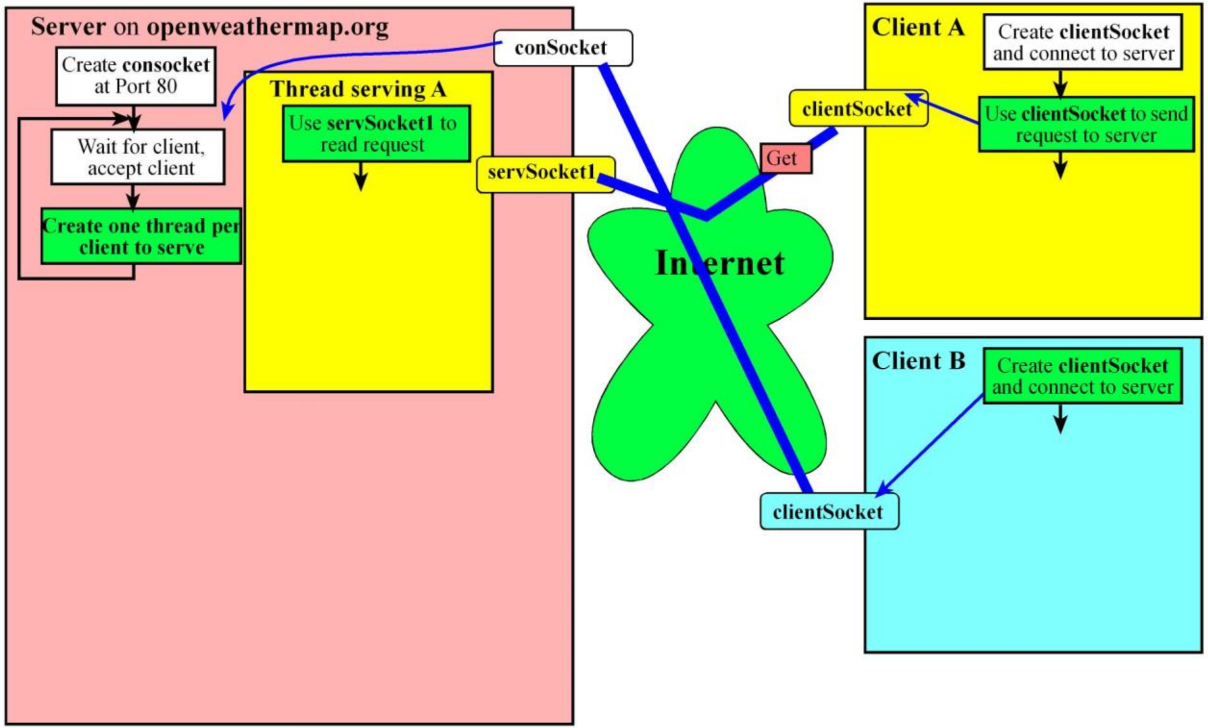
"What's it like in Austin?"



The clientSocket in the client is uniquely connected to the server socket in the server.

Client-server paradigm

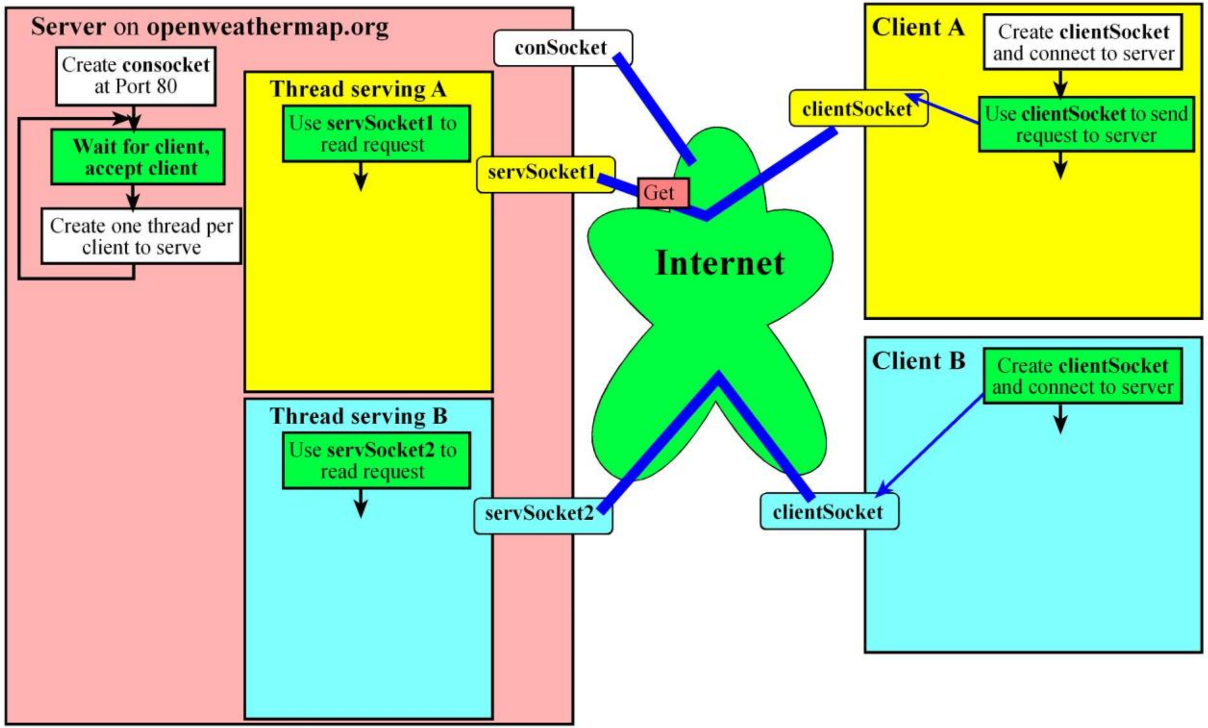
"What's it like in Austin?"



The TCP packet traverses the internet from client to server; a second client starts

Client-server paradigm

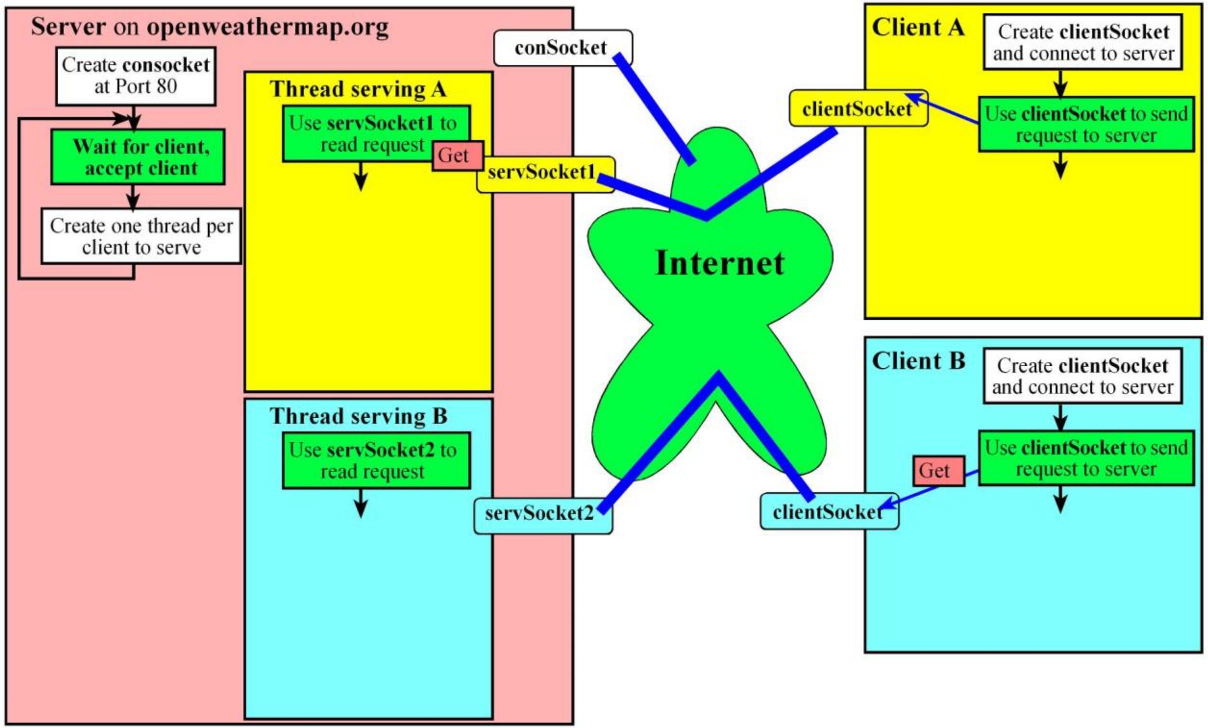
"What's it like in Austin?"



The server creates a second thread and server socket to service the second client

Client-server paradigm

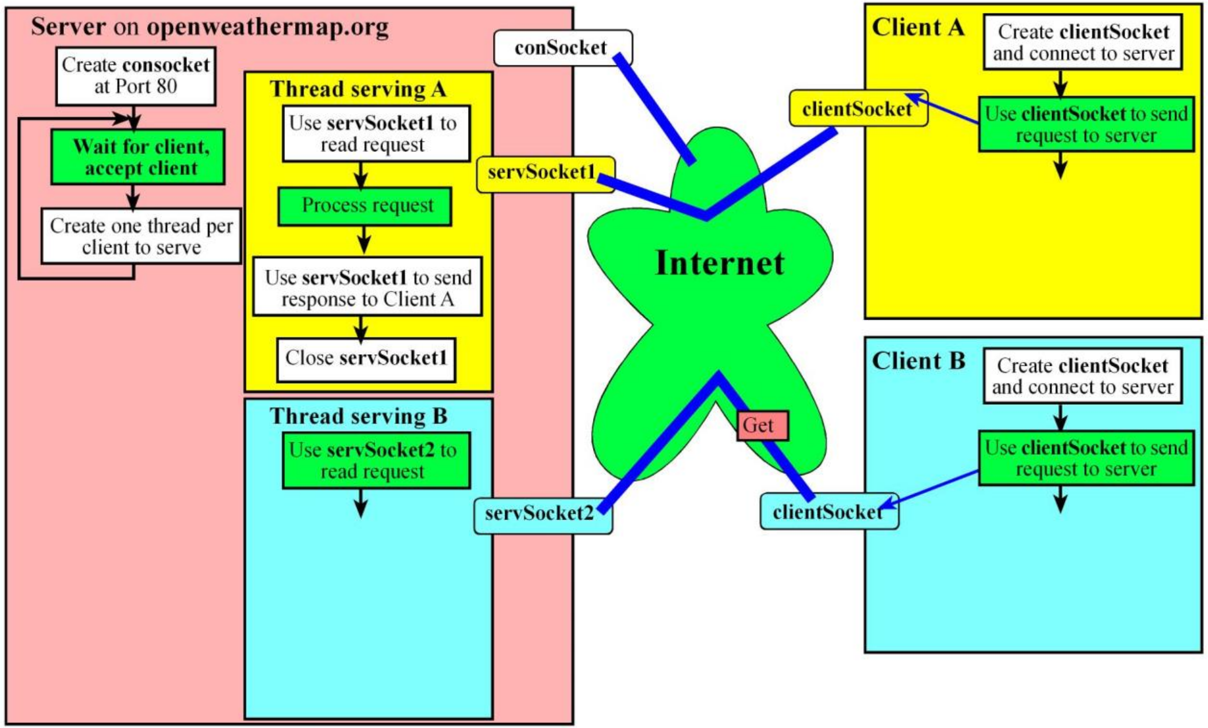
“What’s it like in Austin?”



“What’s it like in Dallas?”

The request from Client A is received (and a second request from Client B has started)

Client-server paradigm

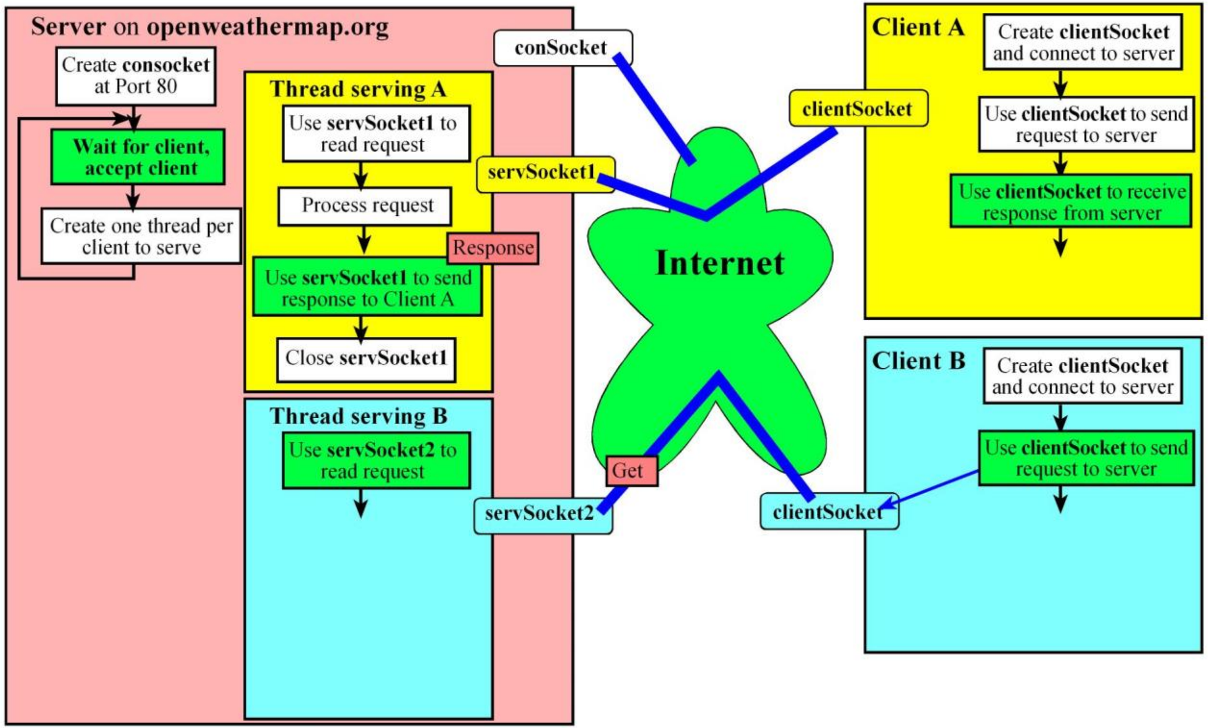


"What's it like in Dallas?"

Let's walk through the steps where two clients are accessing the same server

Client-server paradigm

"It is sunny and 20 C!"

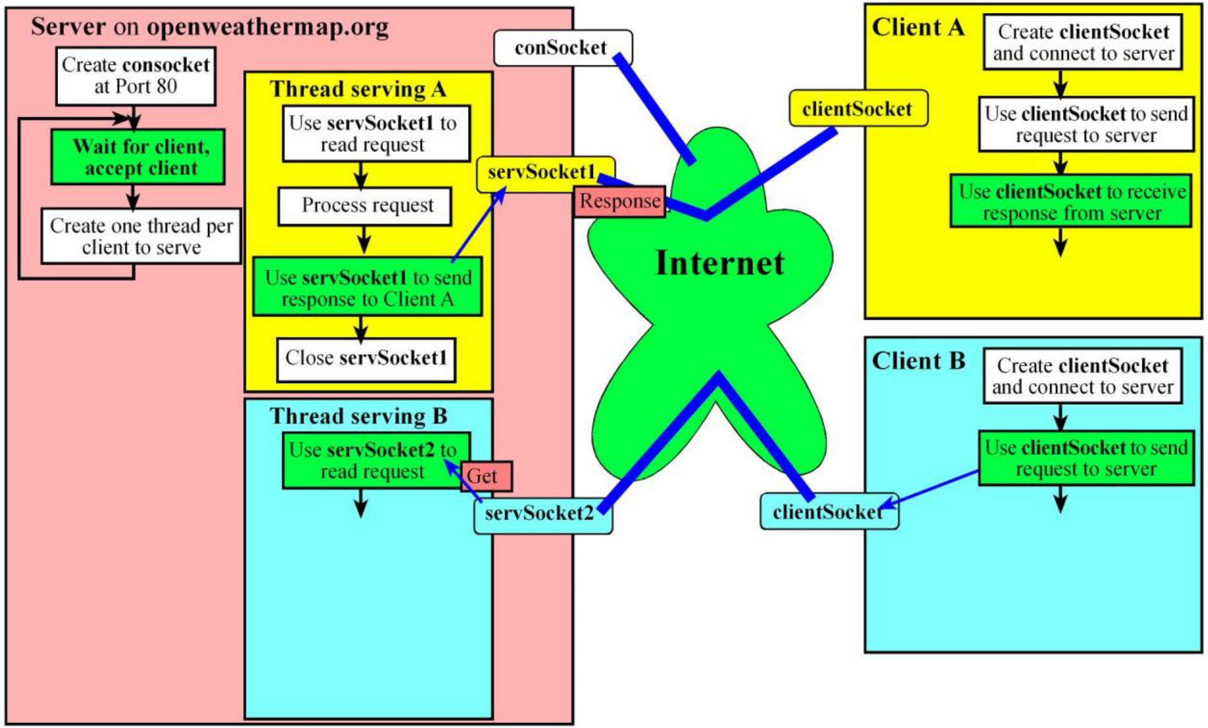


"What's it like in Dallas?"

Let's walk through the steps where two clients are accessing the same server

Client-server paradigm

"It is sunny and 20 C!"

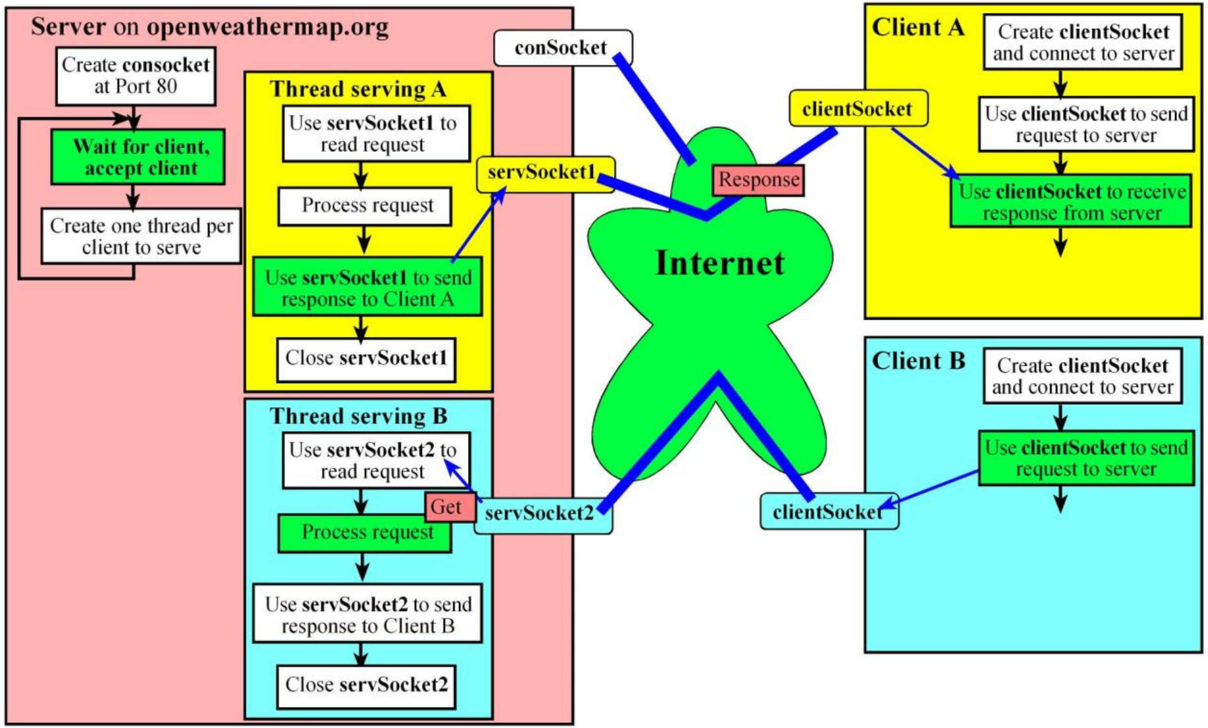


"What's it like in Dallas?"

Let's walk through the steps where two clients are accessing the same server

Client-server paradigm

“It is sunny and 20 C!”

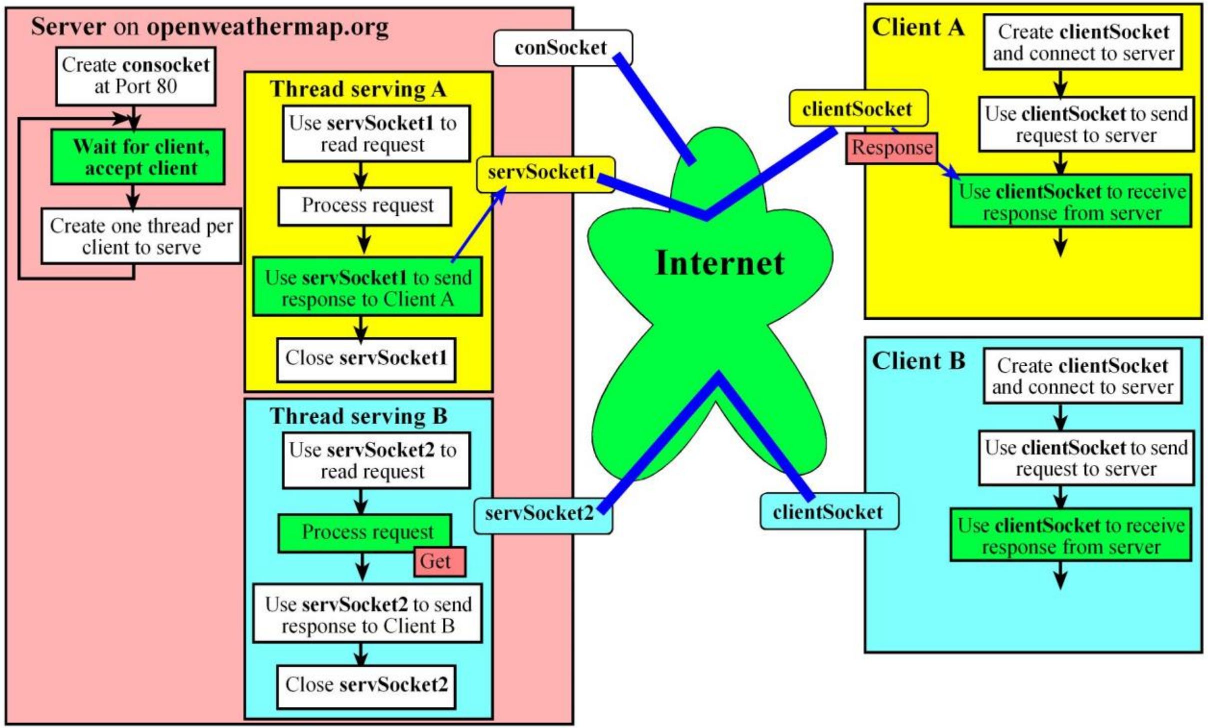


“What’s it like in Dallas?”

The server is processing the request from Client B

Client-server paradigm

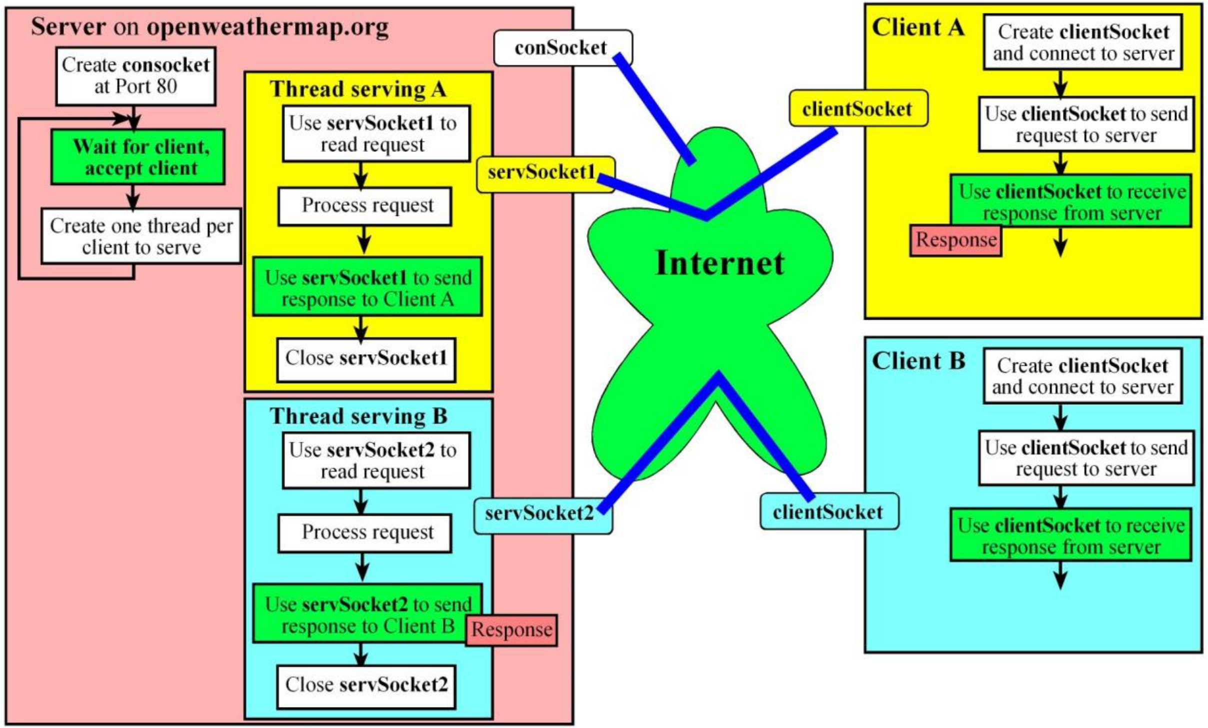
“It is sunny and 20 C!”



The response back to Client A is received

Client-server paradigm

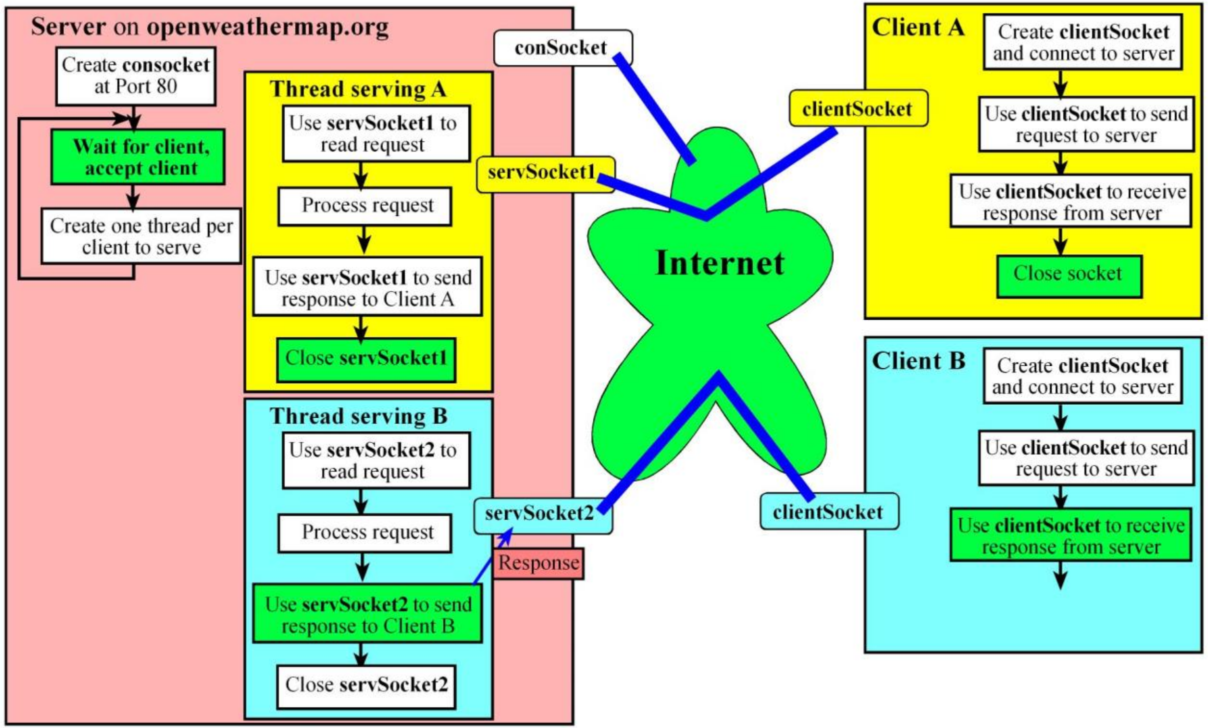
“It is sunny and 20 C!”



“It is cloudy and 15 C!”

The server is sending a response to client B, while client A is processing the response it received

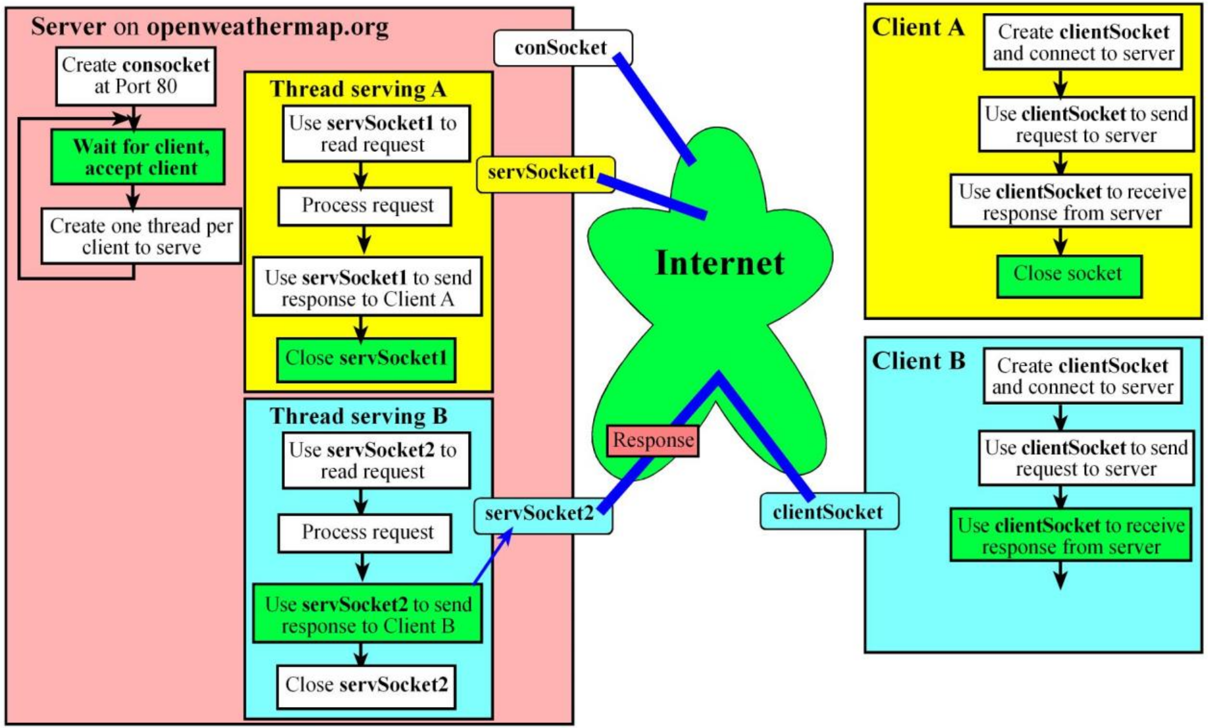
Client-server paradigm



“It is cloudy and 15 C!”

When the client is finished with the server it will close the socket

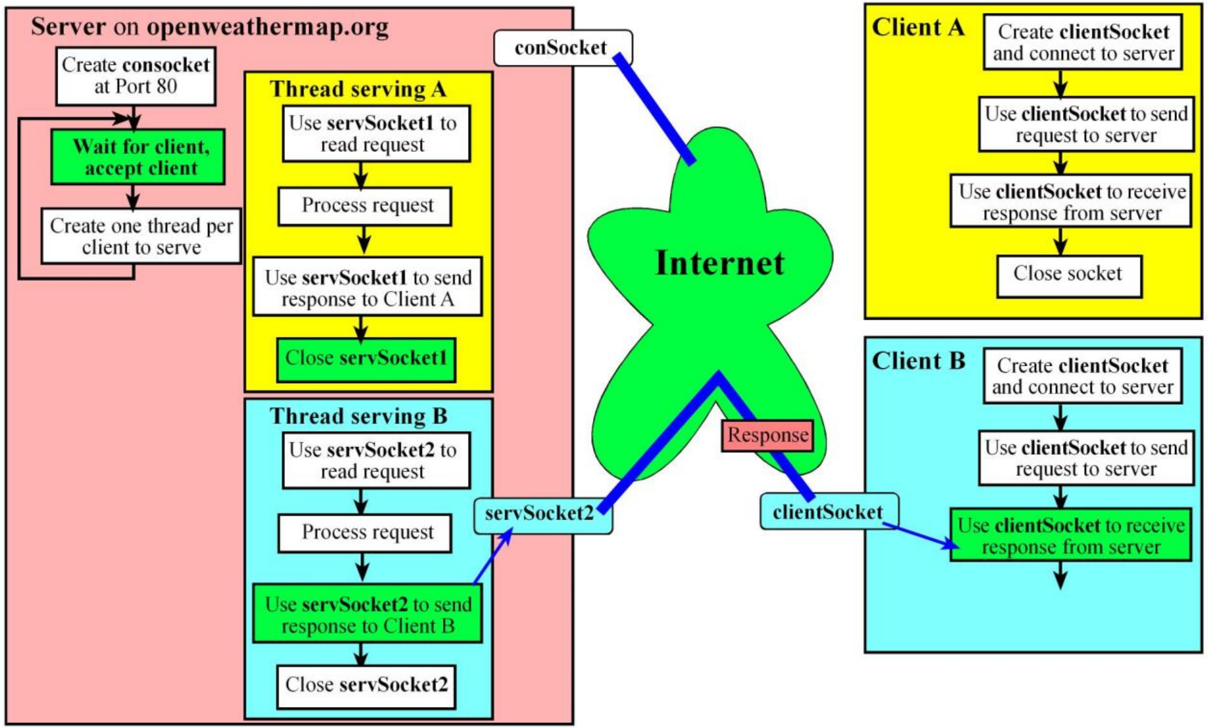
Client-server paradigm



"It is cloudy and 15 C!"

The server will recognize the unique clientsocket-serversocker pairing has been broken

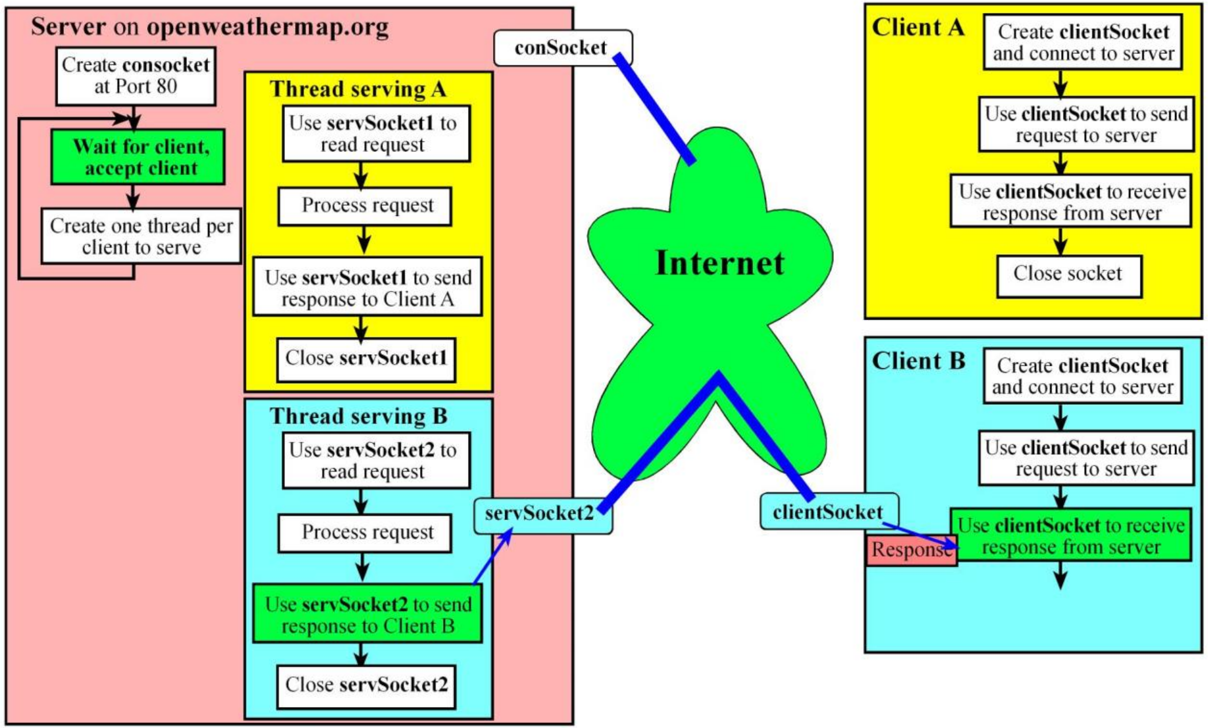
Client-server paradigm



"It is cloudy and 15 C!"

The server will close the server socket associated with the broken connection to client A

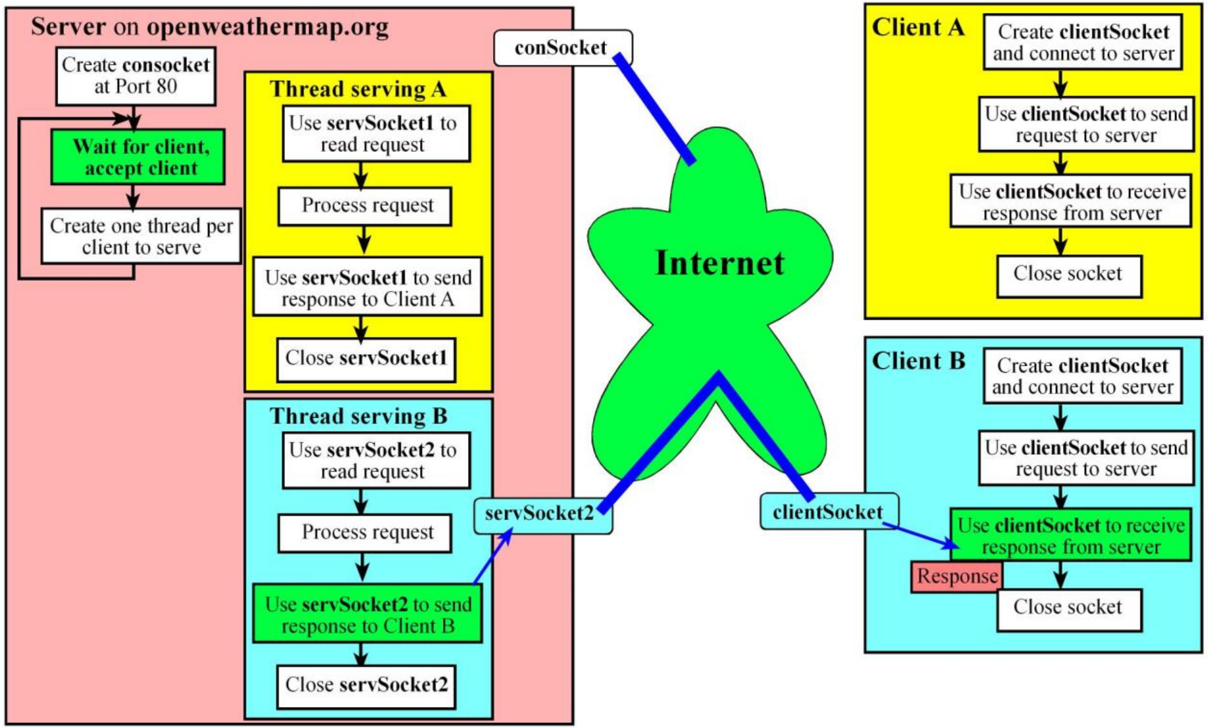
Client-server paradigm



"It is cloudy and 15 C!"

Client B follows the same steps as A

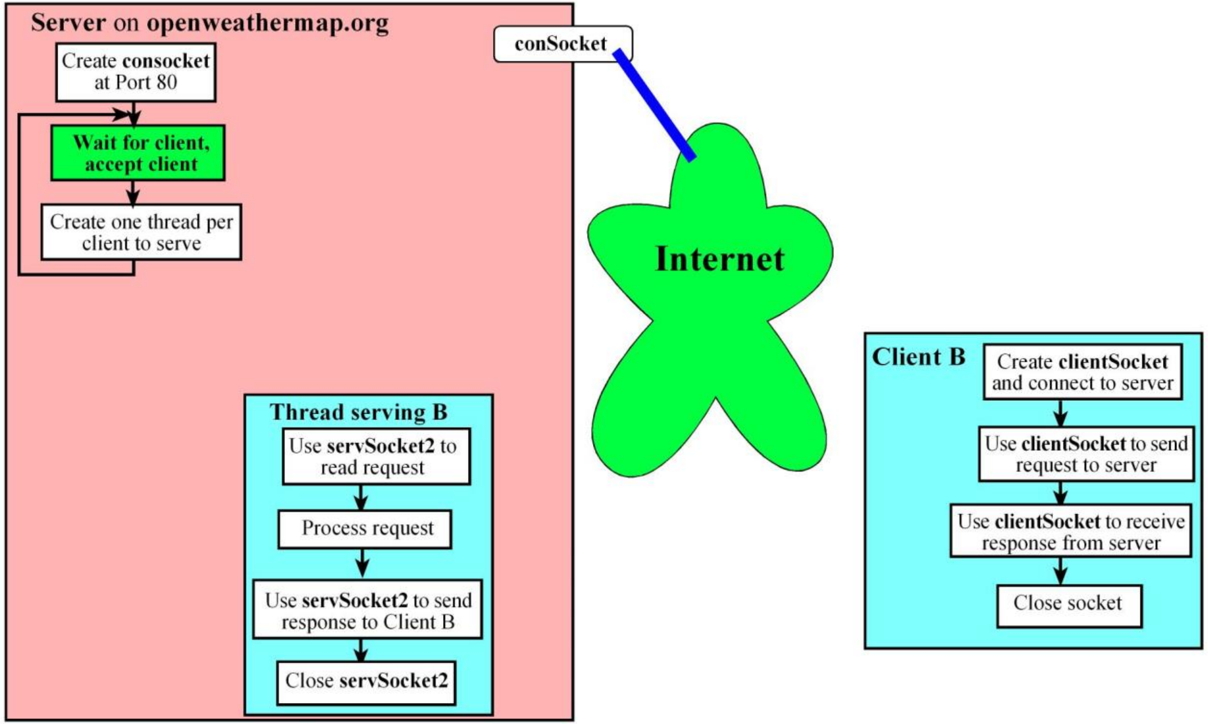
Client-server paradigm



"It is cloudy and 15 C!"

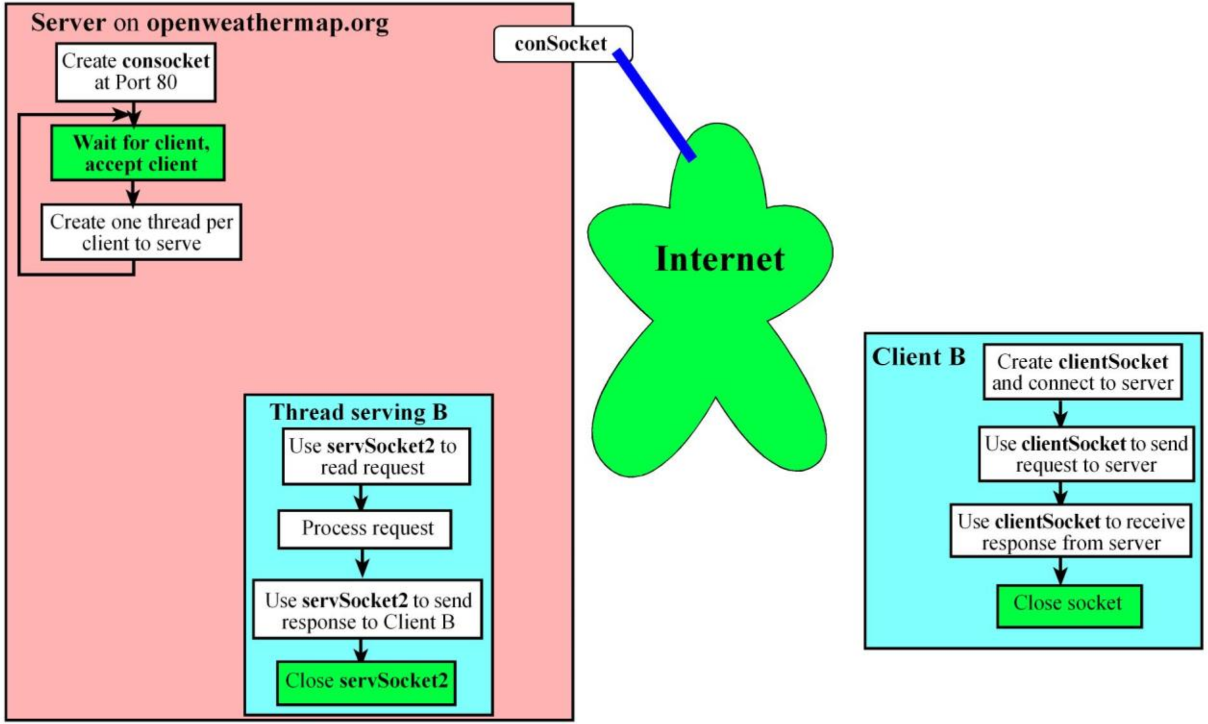
Client B processes the response it received

Client-server paradigm



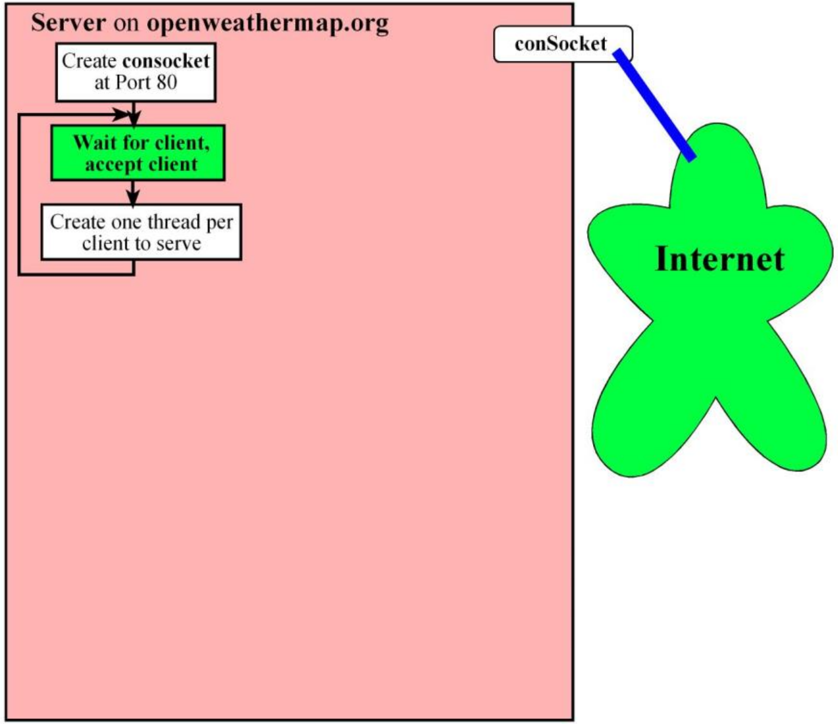
The server is done with client A, so it kills the thread associated with the connection with client A

Client-server paradigm



When the connection from Client B is closed, the server will close the associated server socket

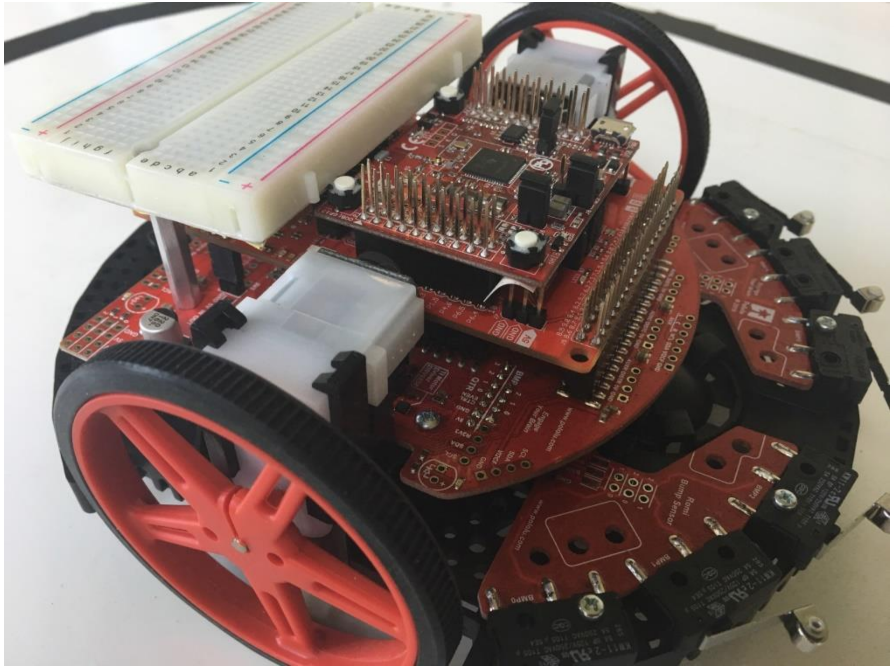
Client-server paradigm



The server is done with client B, so it kills the thread associated with the connection with client B

Summary

- Internet of Things
- TCP/IP
- Sockets
- Client-server paradigm



Module 20

Lecture: **SimpleLink™**

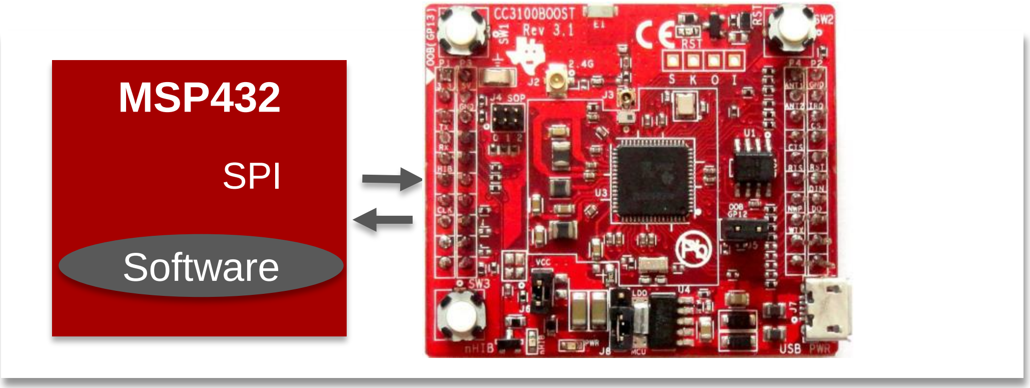
Wi-Fi

Lab 11

You will learn in this module

- Fundamentals of synchronous serial communication
- How to interface a Wi-Fi radio to TI’s LaunchPad Development board
- Make use of software driver (set of functions to create an abstract module)
- Connect to cloud services

SimpleLink™ Wi-Fi® CC3100 wireless network processor BoosterPack™ plug-in module



SimpleLink™

MCU Platform

One environment.
Unlimited potential.

TI SimpleLink™ SDK Software Tools

Software Tools

Code Composer Studio™ IDE	CCS Cloud IDE	Energia IDE
Industry-leading tool suite for advanced optimization	Quickest path to developing a real application	Brings Arduino (Wiring) APIs to TI SimpleLink MCUs

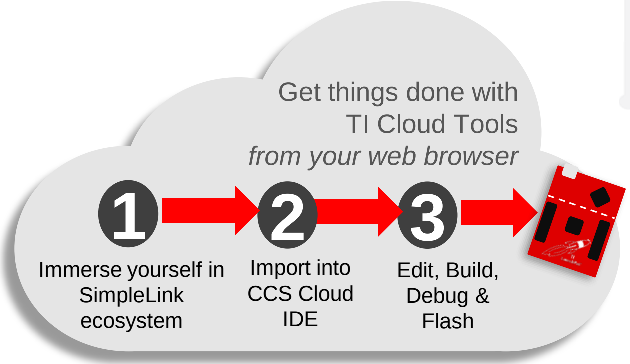
Tool Chain Support

IAR SYSTEMS — KEIL Tools by ARM — GCC

Configure → Build → Debug

- Configure: PinMux
- Build: GCC, UniFlash
- Debug: EnergyTrace

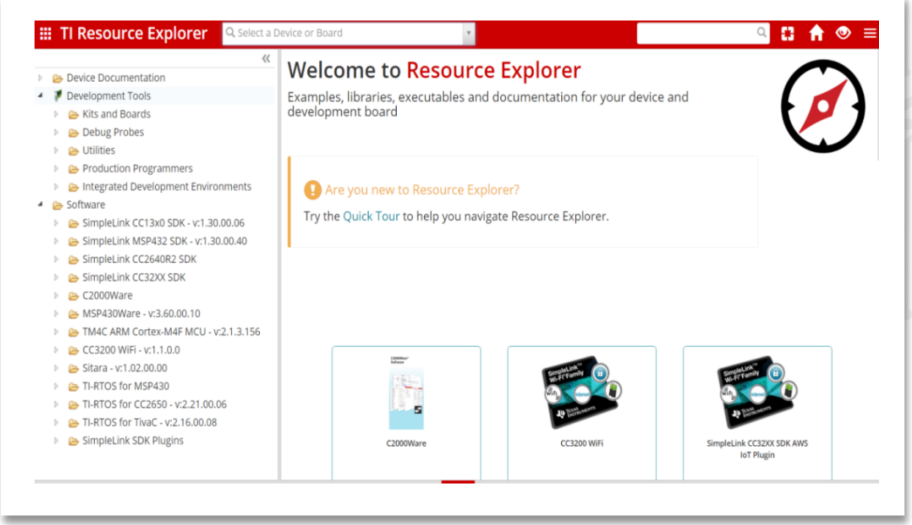
- Multiple IDE support: TI CCS, CCS Cloud, Energia
- Local & Cloud-based access
- Multiple toolchain options to match your development needs



Resource Explorer and SimpleLink™ Academy

Access Resource Explorer to import the latest code examples to CCS

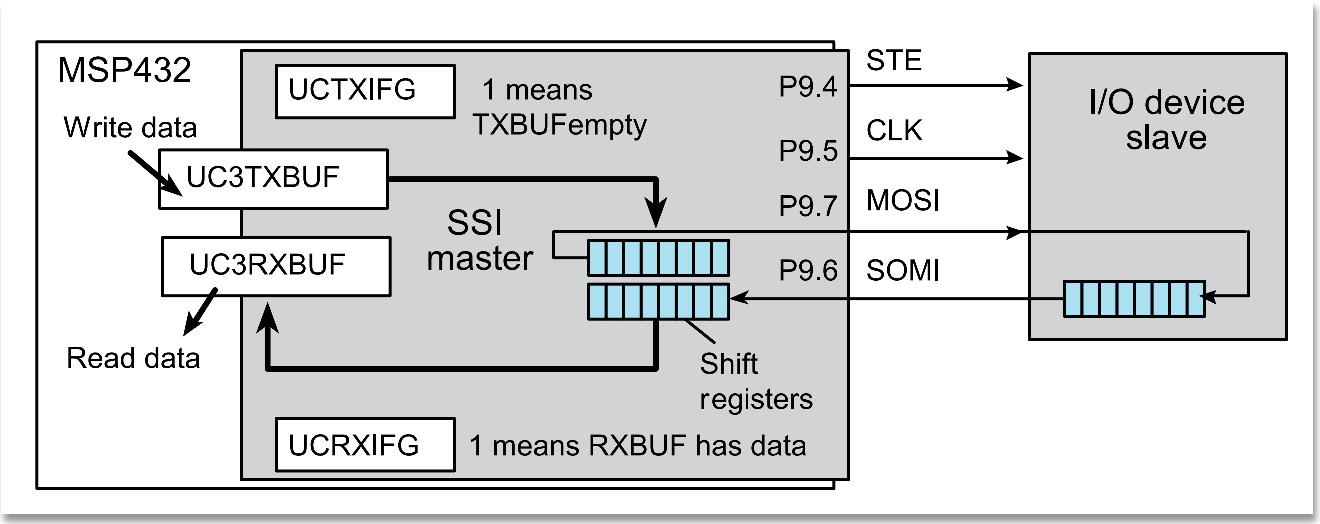
- Accessible from inside CCS (View → Resource Explorer) or from dev.ti.com
- Materials for all TI processors searchable by part number and EVM
- When searching for MSP432 inside Resource Explorer it also contains SimpleLink Academy training with labs that can be imported into CCS that cover topics like TI-RTOS and Connectivity
- Support for TI-RTOS, FreeRTOS, and non-RTOS based code examples



Review of Synchronous Serial Communication on the MSP432

- Synchronous means send clock and data
  - Send data on one edge of clock
  - Receive data on other edge
- Serial Peripheral Interface (SPI) Protocol

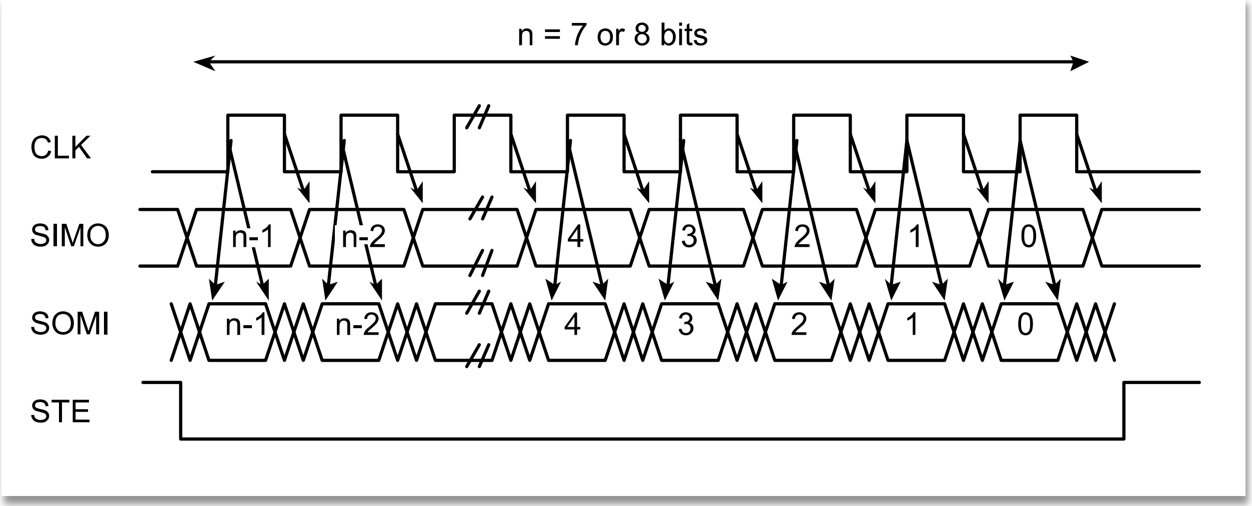
Lab 11



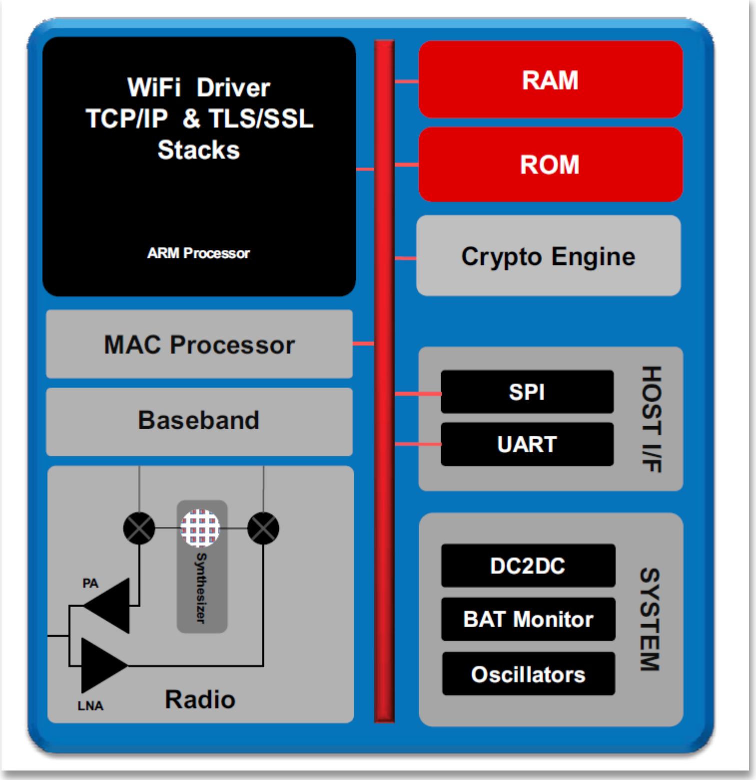
Serial Peripheral Interface (SPI) Timing

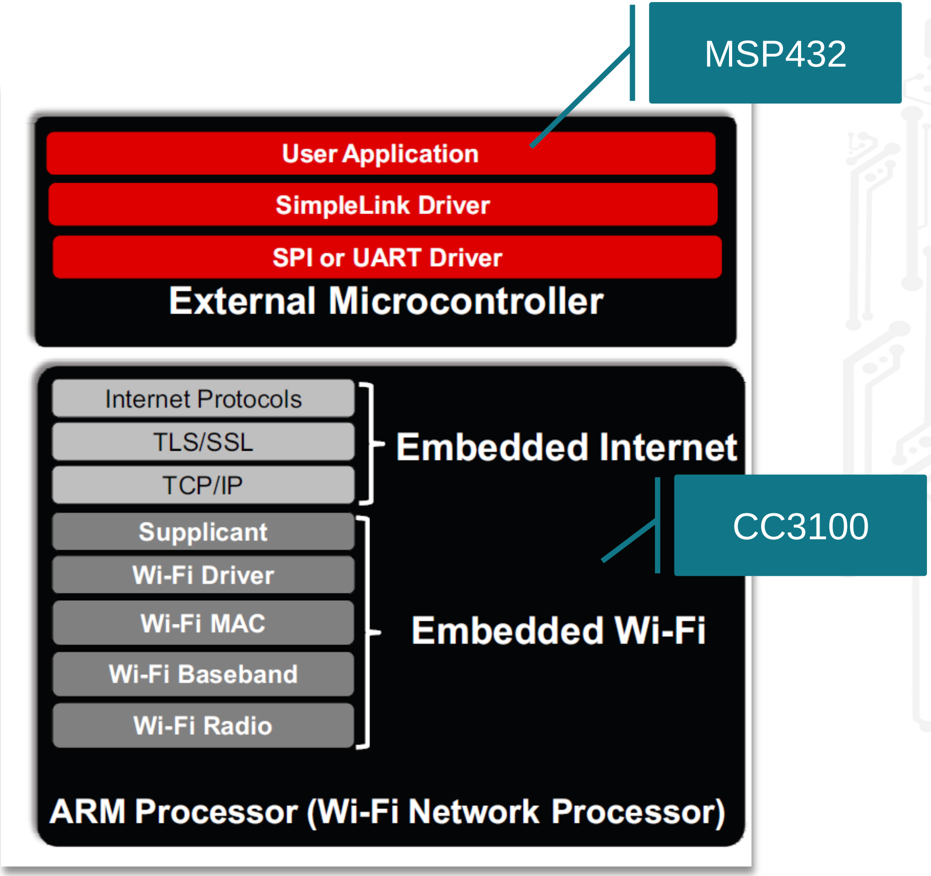
Signals

- Clock
- Data out
- Data in
- Enable



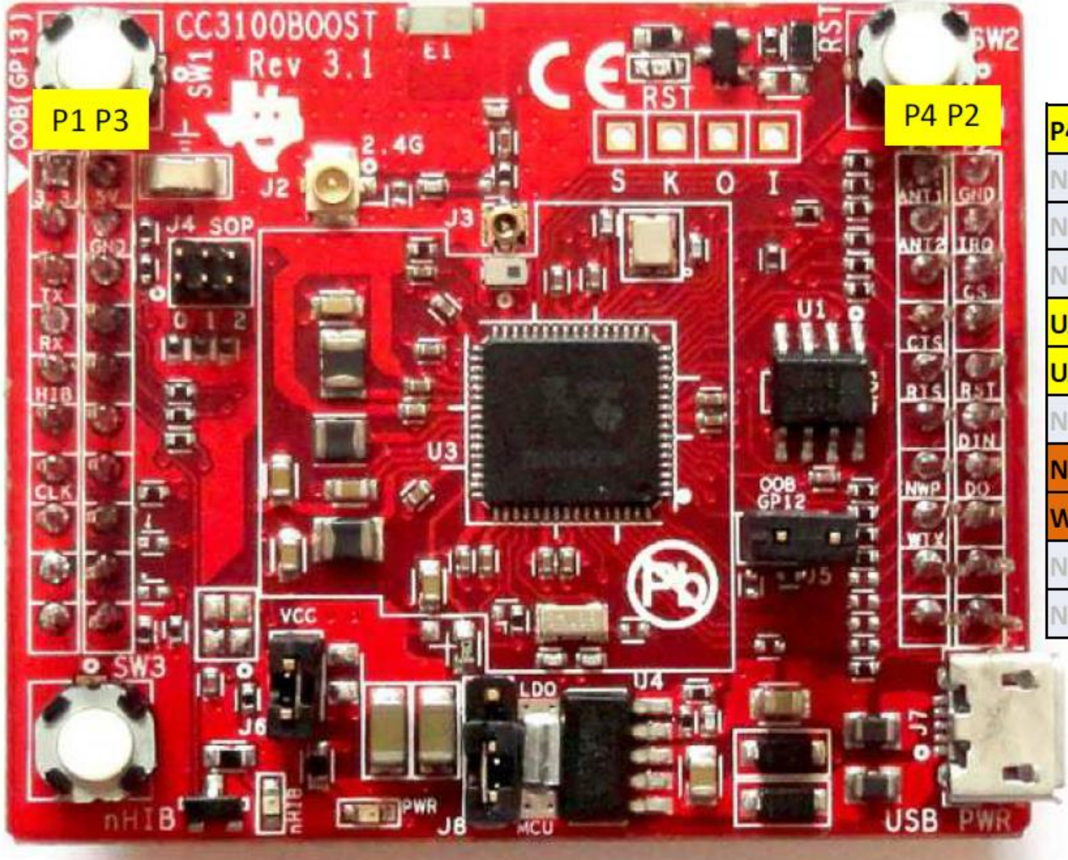
CC3100 Hardware





CC3100 Hardware

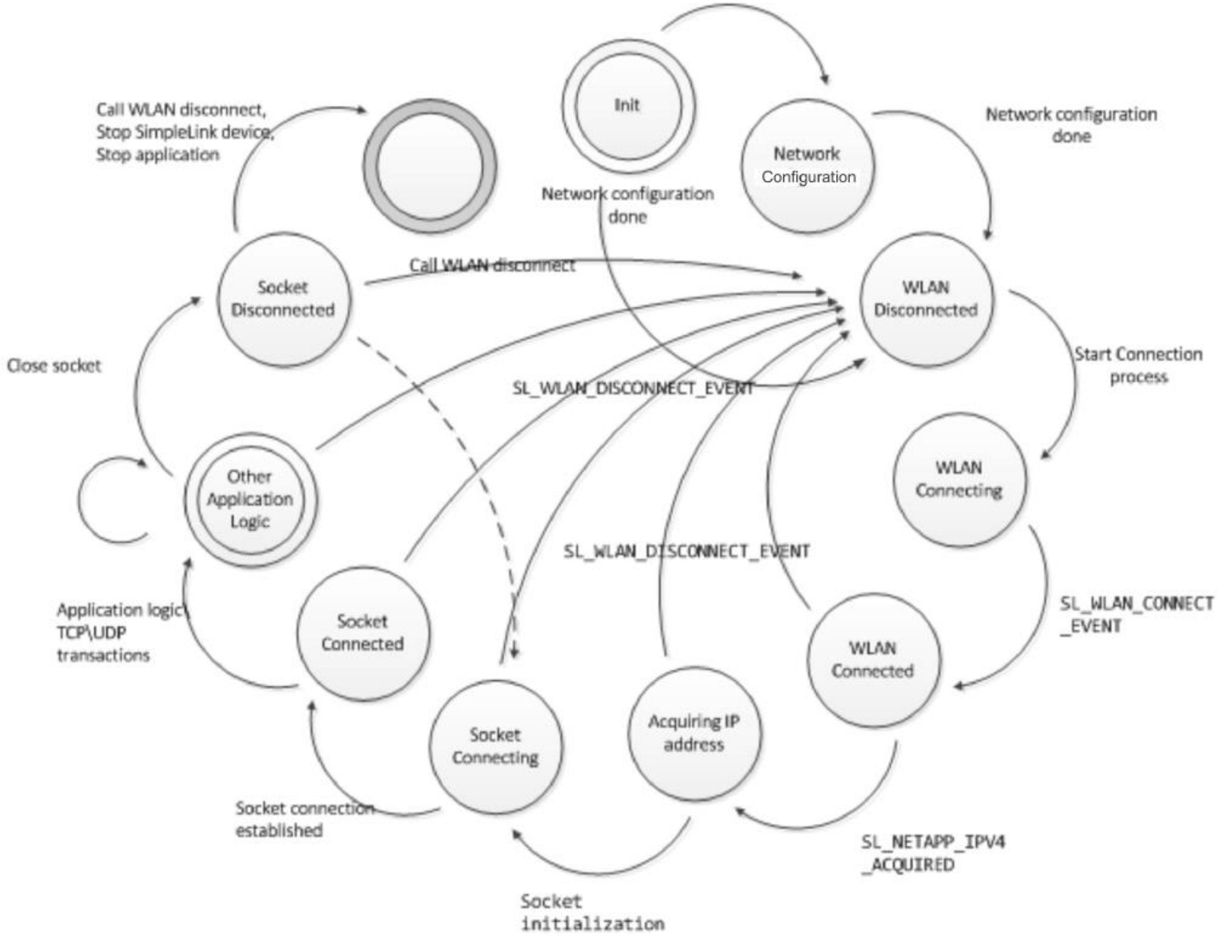
P1	P3
VCC(3.3V)	+5V
UN-USED	GND
UART1_TX	NC
UART1_RX	NC
nHIB	NC
UNUSED	NC
SPI_CLK	NC
UN-USED	NC
UN-USED	NC
UN-USED	NC



P4	P2
NC	GND
NC	IRQ
NC	SPI_CS
UART1_CTS	NC
UART1_RTS	nRESET
NC	SPI_MOSI
NWP_LOG_TX	SPI_MISO
WLAN_LOG_TX	NC
NC	NC
NC	NC

Uses SPI

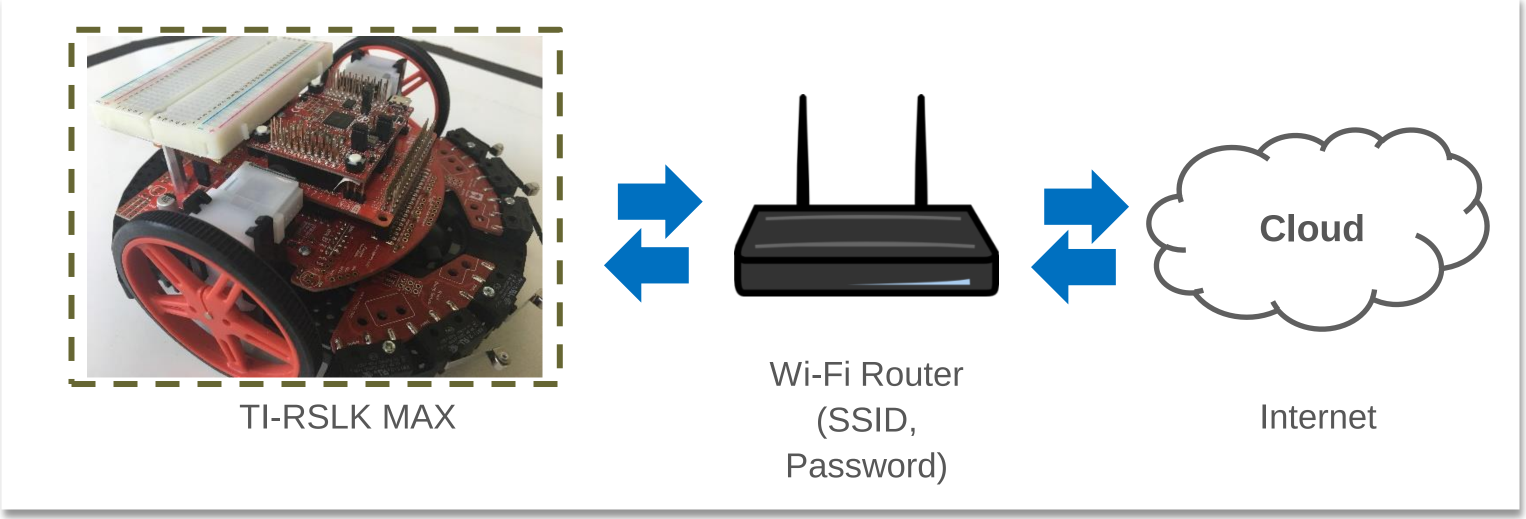
CC3100 Internet on a Chip



Summary

Wi-Fi provides

1. Communication from the robot can log debugging information on the cloud
2. Communication to the robot for remote control or to receive external data
3. Robot can autonomously query information from the web that may be relevant to its operation



ti.com/rslk