MICROCHIP

Regional Training Centers

Designing a WiFi Soluton on WiFi G Demo Board

COM-WiFi

MICROCHIP
Regional Training Centers

Agenda

1. **WiFi® G Demo Board Demonstration**
   - Install MPLAB IDE/X IDE and XC32 1.x compiler
   - Pickit Serial Analysis and WiFi G Demo Board
   - Demo WiFi SoftAP
2. **Embedded Wi-Fi®**
   - What's that?
   - Microchip 802.11 Solution
3. **WiFi® G Demo Board—Hand on**
   - Modify Code to connect Wireless AP
4. **Wireless Network Deployment & Wireless AP Feature**
   - Wireless AP Hand on
5. **Overview of 802.11 Networks**
   - IEEE 802.11 Protocol
   - 802.11 Media Access and Frame Control
   - 802.11 Security
6. **Microchip TCP/IP Stack**
   - TCPIP WiFi Console App
   - WiFi Remote Controller

MICROCHIP

Regional Training Centers

1.Wi-Fi® G Demo Board Demonstration

MICROCHIP

Regional Training Centers

Working with the MRF

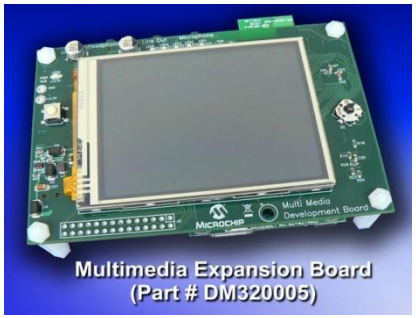
Wi-Fi® Development Tools

- Select a development platform
  - PICDEM.net™ 2 Board, PIC18 Explorer
  - Explorer 16, Multimedia Development Board
  - PIC32 Starter Kit with I/O Expansion Board

DV164037

DM163024

DM320001



DM320005

- Download Microchip TCPIP Framework
- Plug in MRF Wi-Fi® PICtail™ Board

AC164149

- Start with an example
- Create your application

MICROCHIP
Regional Training Centers

Supported Platforms

- 8-bit
  - PICDEM.net™ 2 Board+ AC164149
  - PIC18 Explorer + PICtail™ Board
- 16-bit
  - Explorer 16 + AC164149
- 32-bit
  - PIC32 Starter Kit + I/O Expansion Board + AC164149
  - Explorer 16 + PIC32 PIM + AC164149
  - PIC32 Starter Kit + Multimedia Expansion Board

MICROCHIP
Regional Training Centers

Quick Start Tools

	Explorer Based Development Board				Stand Alone Evaluation Kit			
Series	MRF		RN		MRF		RN	
Platform	PICtail™/PICtail Plus Board		PICtail™/PICtail Plus Board		Wi-Fi® Comm Demo		Eval Kit	
Module	MRF24WB	MRF24WG	RN131	RN171	MRF24WB	MRF24WG	RN131	RN171
Image								
Part #	AC164136-04	AC164149	RN-131-PICTAIL	RN-171-PICTAIL	DV102411	DV102412	RN-131-EK	RN-171-EK
Availability	NOW	NOW	NOW	NOW	NOW	NOW	NOW	NOW

TCP/IP stacks and Application Demos online at: www.microchip.com/mla

Pickit Serial Analysis and WiFi G Demo Board

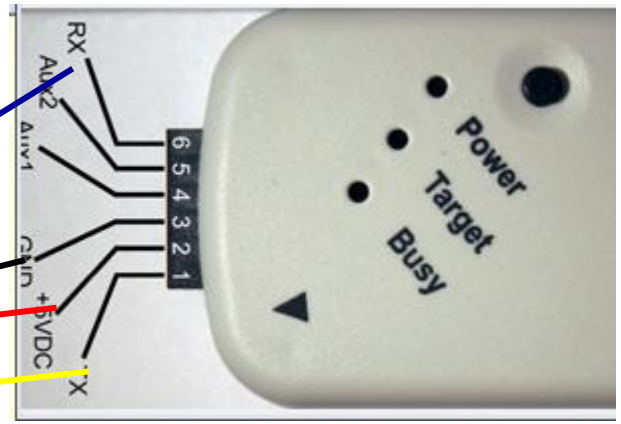
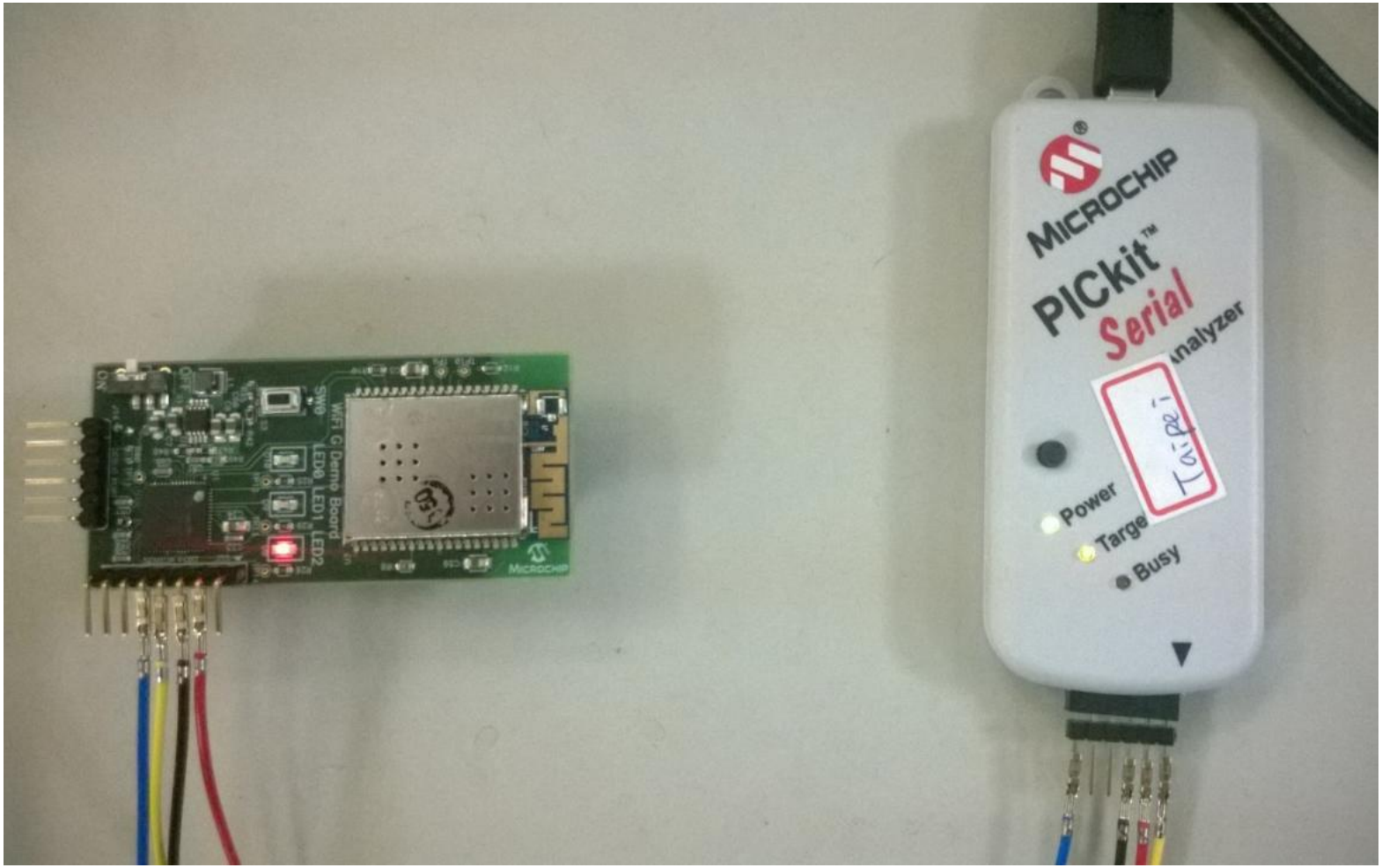


TABLE 2-1: J15 EXPANSION PORT PIN CONFIGURATIONS

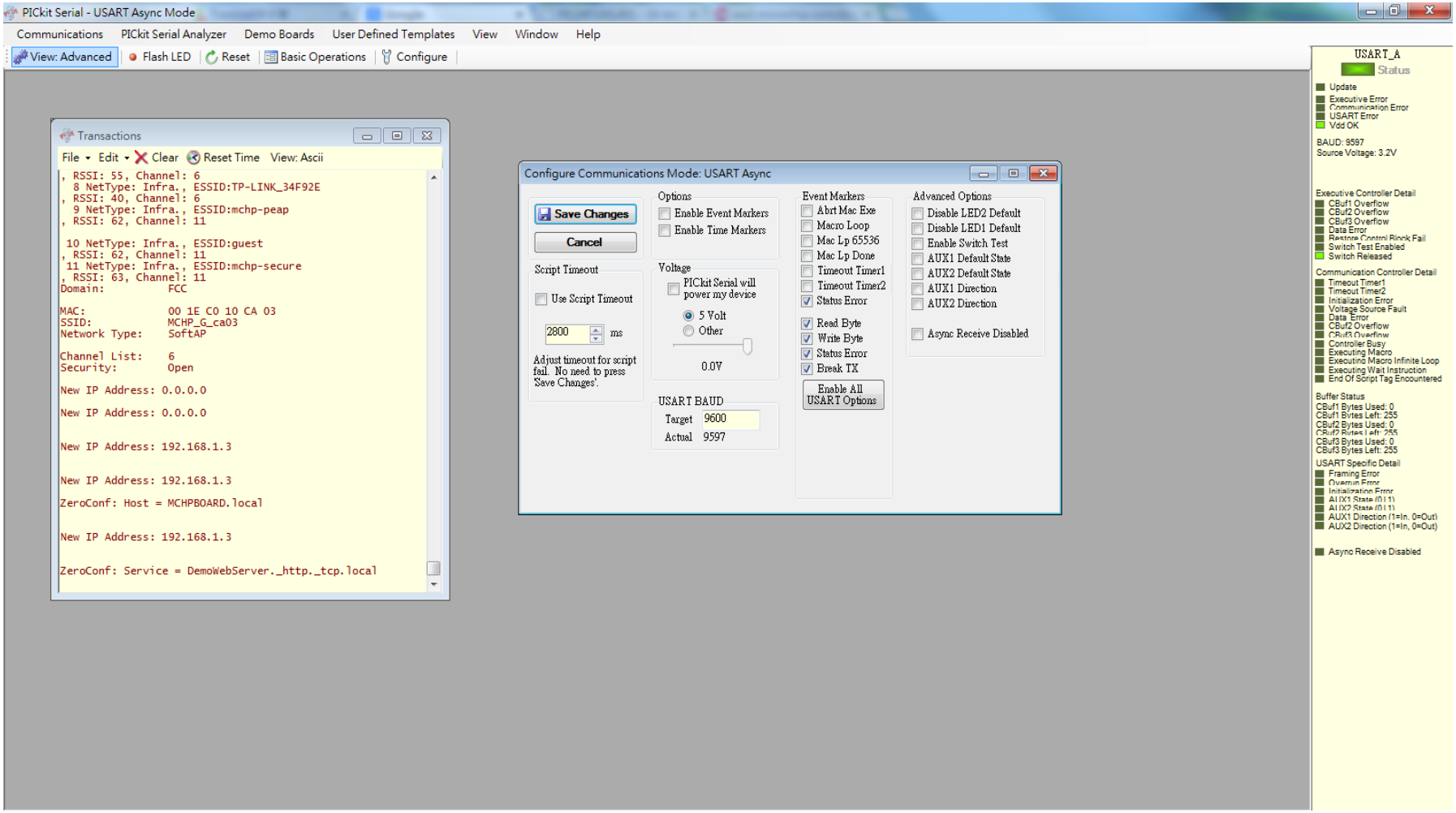
Pin 1	Pin 2	Pin 3	Pin 4	Pin 5	Pin 6	Pin 7	Pin 8	Interface
SCK	+3.3V	GND	SDI	SDO	/SS	-*	-*	SPI
-	+3.3V	GND	SDA	SCL	-	-*	-*	I^2C™
RTS	+3.3V	GND	RX	TX	CTS	-*	-*	USART
TX2	+3.3V	GND	RX1	TX1	RX2	-*	-*	DUAL USART

MICROCHIP
Regional Training Centers

Pickit Serial Analysis and WiFi G Demo Board



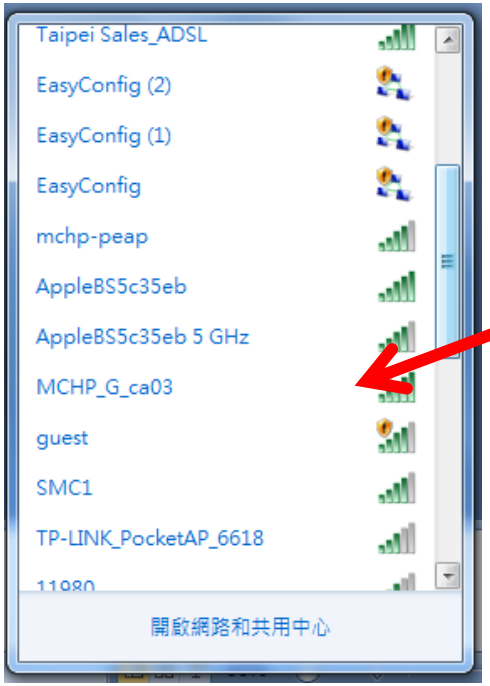
Pickit Serial Analysis and WiFi G Demo Board



Hand on

- **Lab 1: Compiler WiFiGDemo _SoftAP Project and Using IPHONE or Android PHONE to connect WiFi G Demo board**

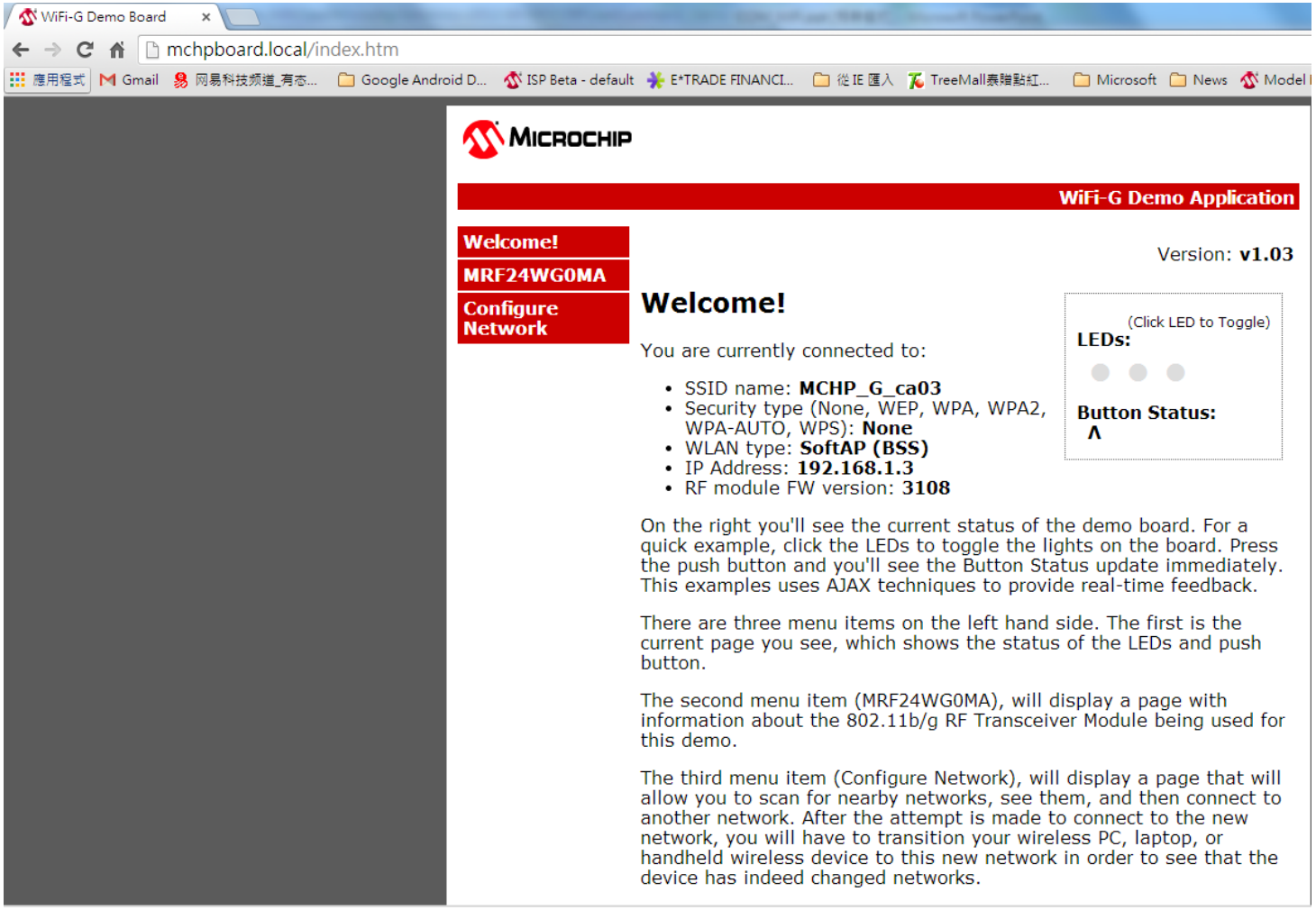
Selecting a Network Name (SSID)



1.Connect to SSID: MCHP_G_xxxx
xxxx = Lost 4 digits of MAC address

2.Browse to: http://mchpboard.local or http://192.168.1.3

Hand on



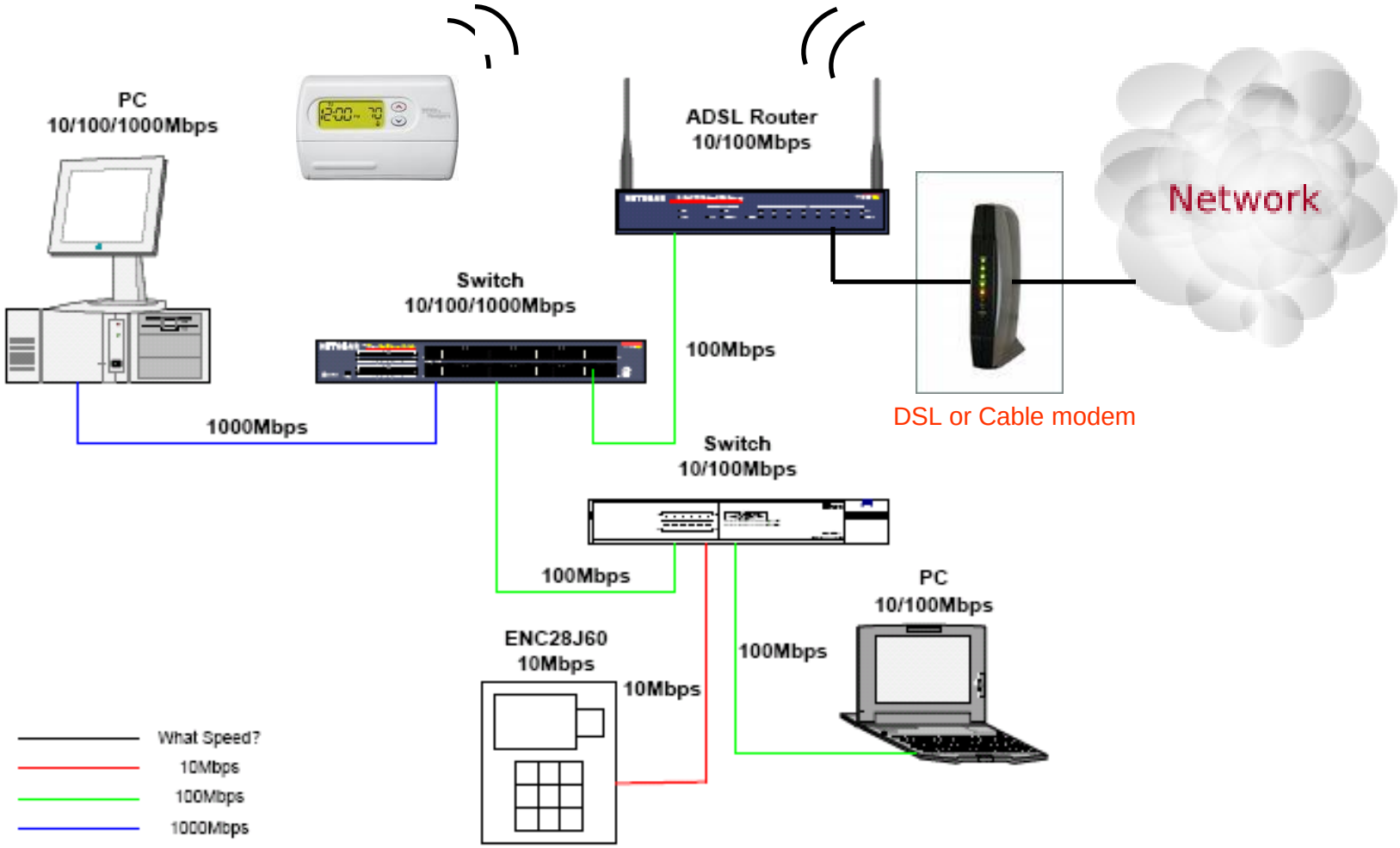
MICROCHIP

Regional Training Centers

2. Embedded WiFi

MICROCHIP
Regional Training Centers

What is Wi-Fi®?



MICROCHIP
Regional Training Centers

What is Wi-Fi®?

- Ethernet is the most widely deployed datacom network in the world
- Wi-Fi is wireless Ethernet
  - Adds mobile internet connectivity
  - Removes the wire, but retains the LAN, WAN, WWW connection

What is the Internet?

- A "nuts and bolts" view -

PC

server

wireless laptop

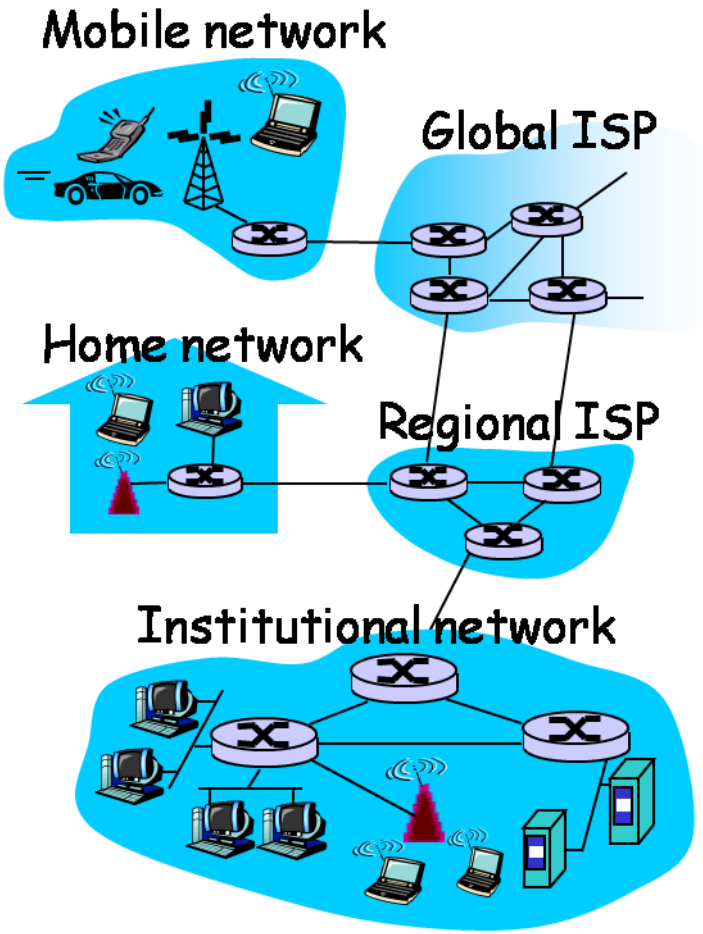
cellular handheld

access points

wired links

router

- *Millions of connected computing devices*:
  - Hosts (or "stations")
    - Some run application services ("Servers")
    - Some consume application data ("Clients")
  - All running TCP/IP protocol suite
- *Communication links:*
  - Fiber, copper, radio, satellite
  - Data Transmission rate = link bandwidth
- *Routers:* Forward packets (chunks of data)



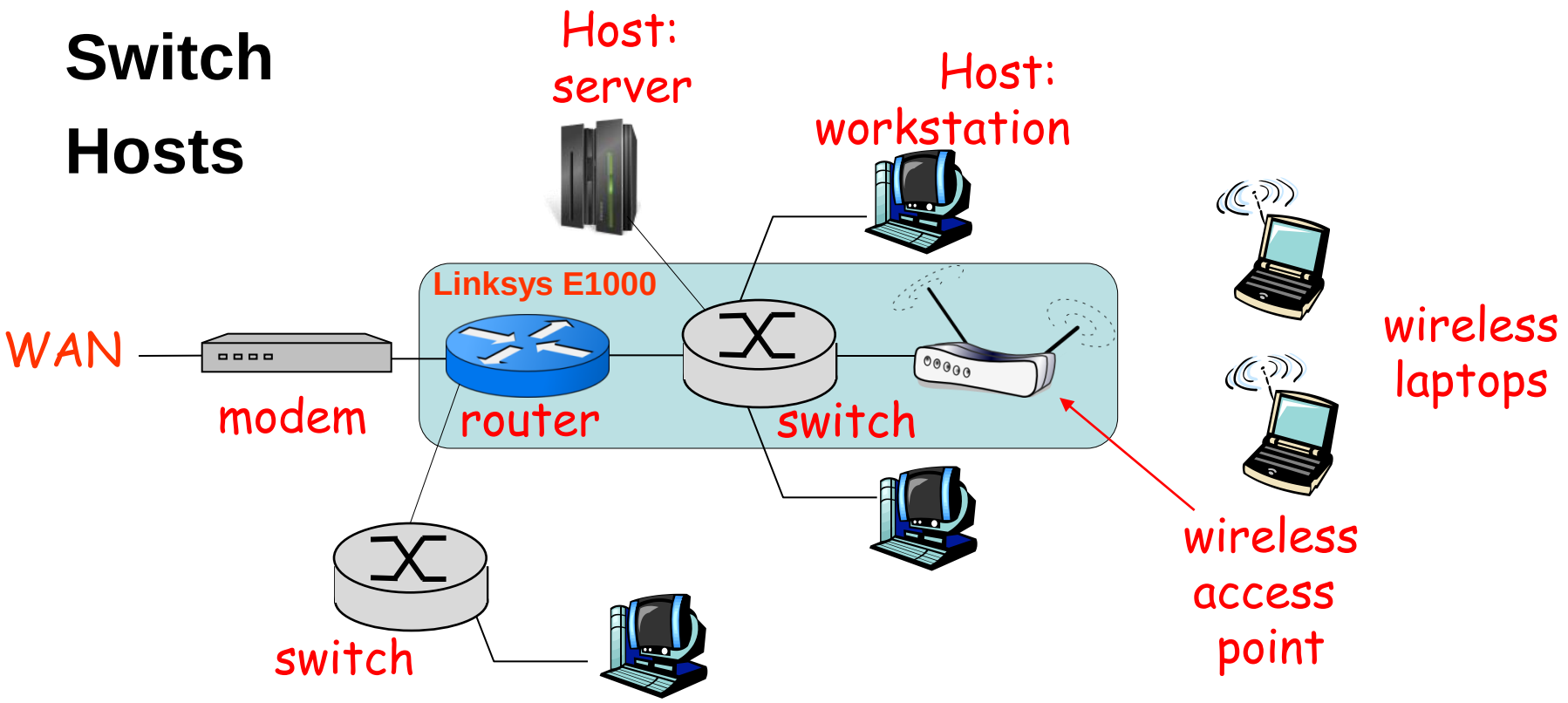
Typical Access Network

Modem to connect to the Wide Area Network (WAN) – other physical end to Ethernet

Router

Switch

Hosts



MICROCHIP
Regional Training Centers

Embedded Wi-Fi®

- Embedded products require
  - 8, 16, 32-bit, processor support
  - Small memory footprints
    - Run from on-chip memory
  - Low power
    - Battery operation
  - Fast time to market

Microchip 802.11 Solution

- **Infrastructure (BSS)**

- **Ad Hoc (IBSS)**

Embedded Wi-Fi®

Wi-Fi Protected Setup™

- Enables users to:
  - Automatically configure new wireless networks
  - Add new devices
  - Enable security

http://www.wi-fi.org/wifi-protected-setup

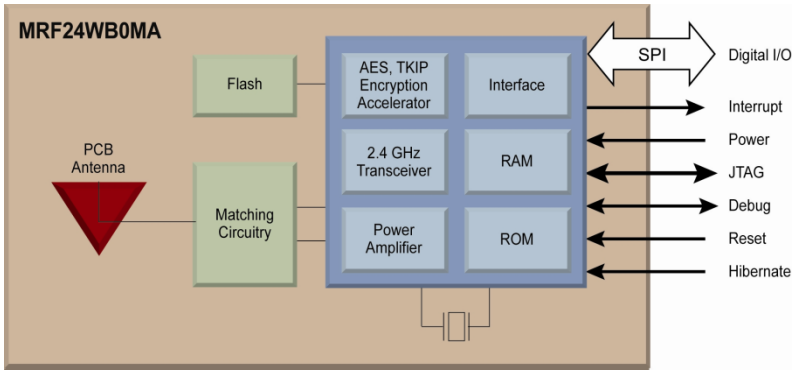
Wi-Fi Direct™

- Enables users to:
  - Connect to one another without joining a traditional network

http://www.wi-fi.org/Wi-Fi_Direct.php

MRF24WB0MA/MB 802.11b Wi-Fi®

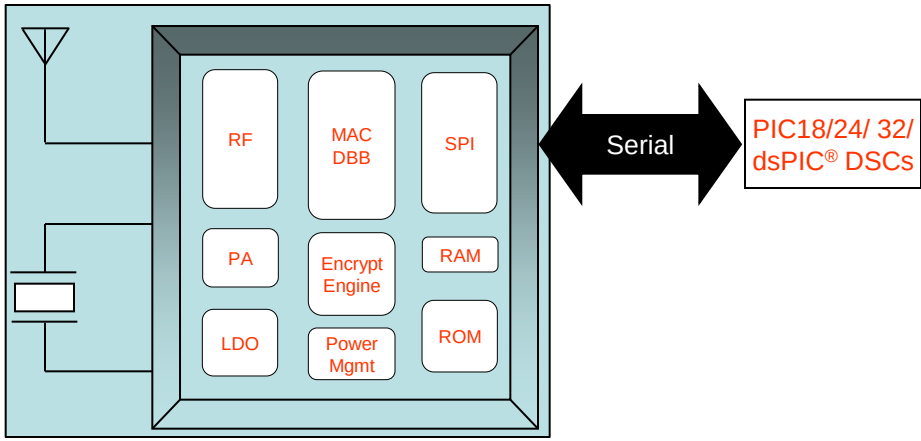
- **MRF24WB0MA and MRF24WB0MB Wi-Fi Transceiver Module Features**
  - Low data-rate Wi-Fi
  - 1&2 Mbps
  - Used when can control AP that is used
  - Ultra low power consumption
    - 0.1uA hibernate
    - 250uA 802.11PS
    - 185mA Max. Tx
  - FCC, and international regulatory certification
  - Wi-Fi Certified
  - Works with PIC® microcontrollers
  - SPI interface to PIC microcontrollers
  - Supports WEP, WPA and WPA2 security protocols



- **Free TCP/IP Stack**
  - Uses current Microchip TCP/IP stack
  - Free download from
    www.microchip.com/mla

MRF24WG0MA/MB 802.11b/g Wi-Fi®

- **MRF24WB0MA and MRF24WB0MB Wi-Fi Transceiver Module Features**
  - 5mbps streaming Wi-Fi, 54mbps burst
  - 1,2,5.5, 6,9,11,12,18,24,36,48,54 Mbps
  - Connects to standard wireless access points
  - FCC and international regulatory certification
  - Works with PIC® microcontrollers
  - SPI interface to PIC microcontrollers
  - SoftAP, Wi-Fi Direct
  - Infrastructure and Adhoc
  - 802.11PS
  - Wi-Fi Direct Client
  - Supports WEP, WPA, WPA2, WPA-EAP security protocols



- **Free TCP/IP Stack**
  - Uses current Microchip TCP/IP stack
  - Free download from www.microchip.com/mla

Product Portfolio

MODULES	MRF24WB0MB	MRF24WB0MA	RN-171	RN-131 C/G	MRF24WG0MB	MRF24WG0MA
802.11 Radio	b	b	b/g	b/g	b/g	b/g
Power	+10dBm	+10dBm	+12dBm	+18dBm	+18dBm	+18dBm
Antenna	uFL	Built-in	PIN	Chip/uFL	uFL	Built-in
Stack	PIC® MCU	PIC MCU	Onboard	Onboard	PIC MCU	PIC MCU
MCU Support	PIC18+	PIC18+	PIC12+	PIC12+	PIC18+	PIC18+
Throughput	1mbps	1mbps	0.7-2mbps	0.7-2mbps	5mbps	5mbps



RN Wi-Fi® Modules

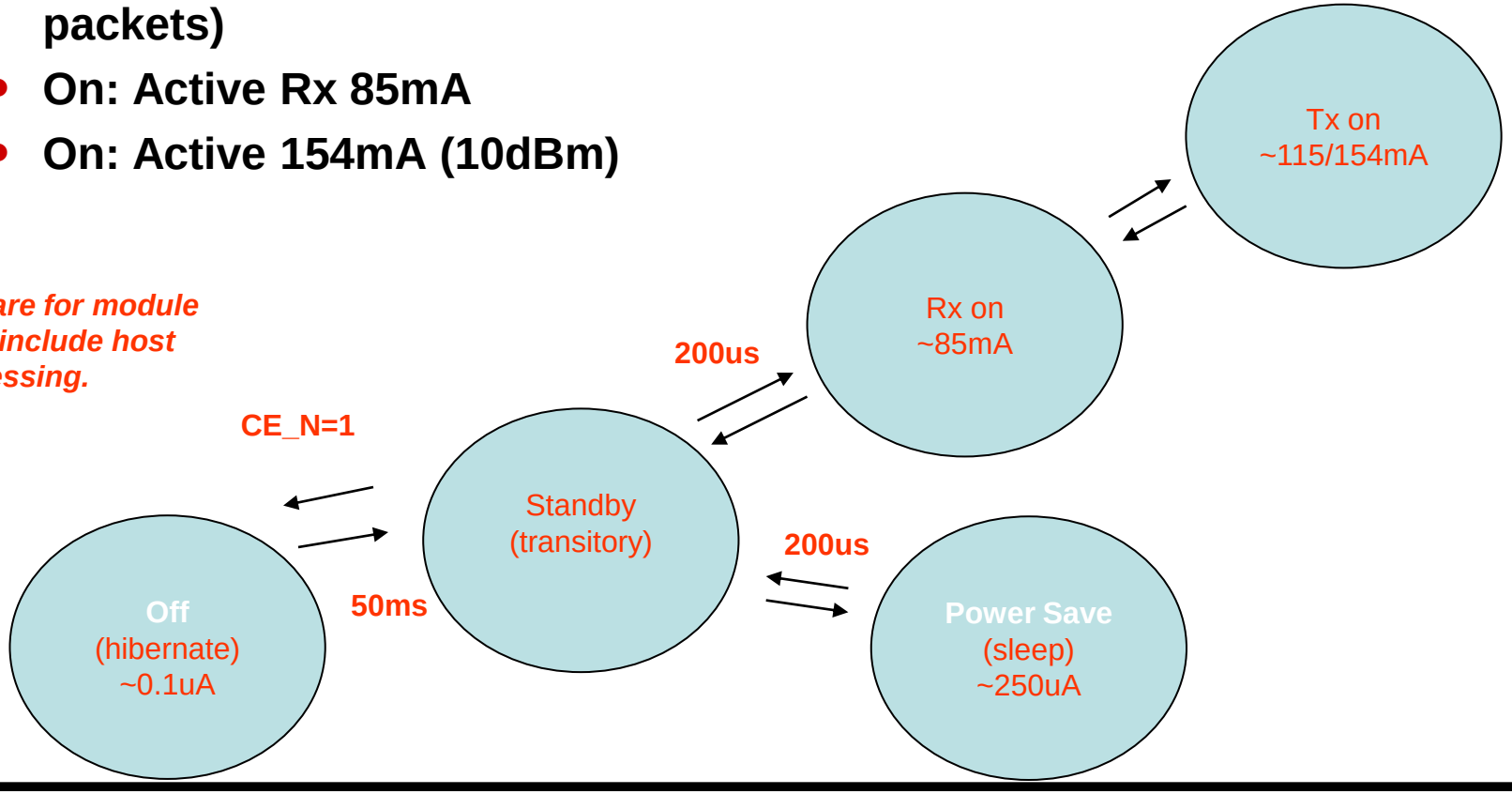
	RN-171	RN-131
Radio	True 802.11 b / g	True 802.11 b / g
Antenna	PCB trace, wire, chip, U.FL connector for external	On board chip and U.FL connector for external
Power	4uA sleep, 30mA RX, 130mA TX	4uA sleep, 40mA RX, 200mA TX
Range	10 meter to 180 meter (software configurable)	200 meter chip 300 meter external 4” dipole
HW Interface	TTL UART, SPI slave*	TTL UART, SPI slave*
Development Kit	RN-171-EK	RN-131-EK

http://www.microchip.com/wi-fi

Understanding Power Modes on MRF24WB0Mx

- Several power modes for low-power operation
  - Off: 0.1uA (including SPI flash on module)
  - Power Save: 250uA (Stand-by & between packets)
  - On: Active Rx 85mA
  - On: Active 154mA (10dBm)

Note: Times are for module and do not include host processing.



Product Fit

- Types of products that are ideal
  - Small data quantity to transfer
  - Non-continuous transfer
  - Data is not high bandwidth streaming







MICROCHIP
Regional Training Centers

End-User Station Configuration

- **May need to:**
  - **select or enter SSID**
  - **enter Security Key**

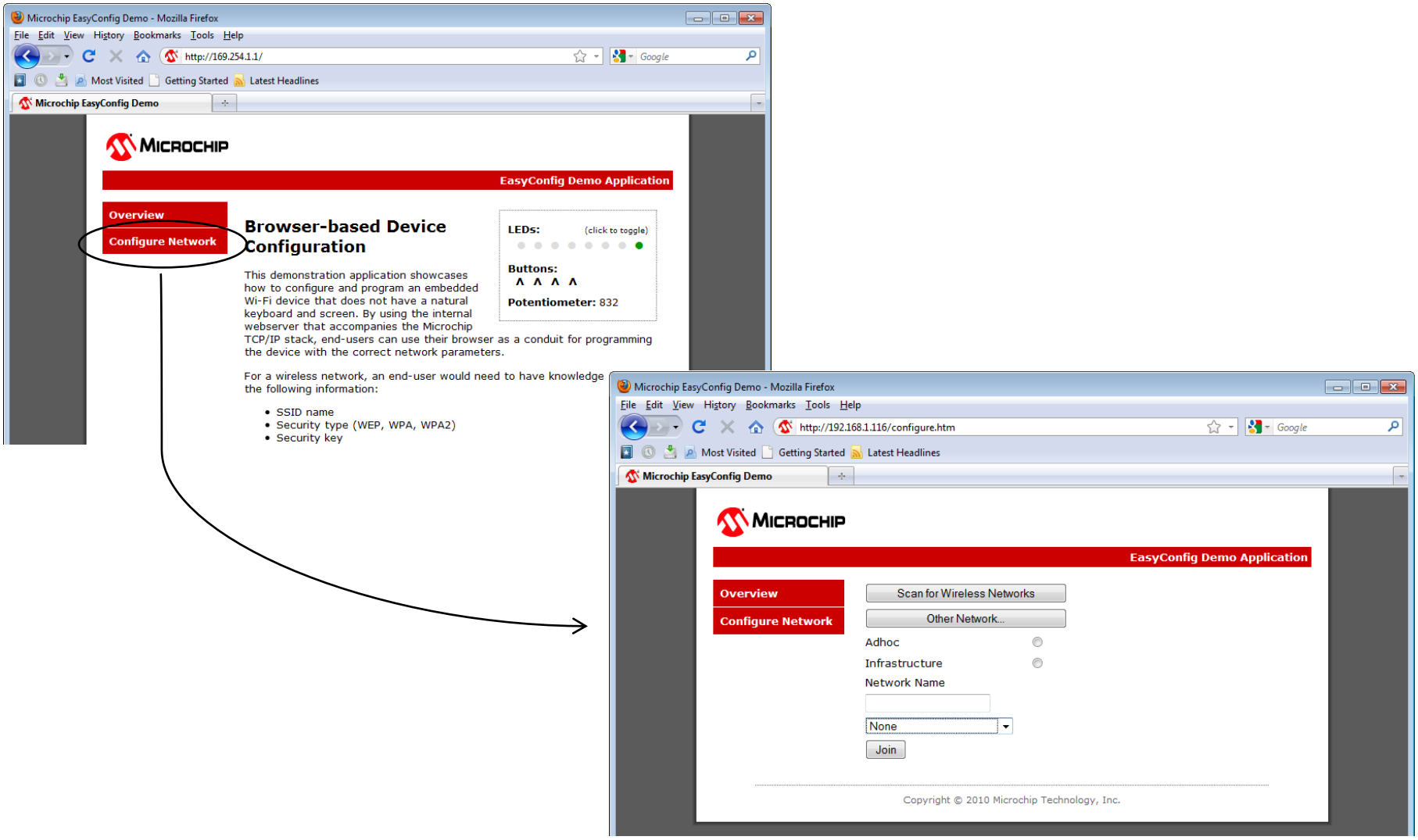
- **Types of products:**
  - **Product has display and data entry – easy**
  - **Product does not have display or keyboard**

MICROCHIP
Regional Training Centers

Configuring Light Clients

- **USB**
  - Can use USB cable from PC or USB stick
- **WPS™**
  - "one-button", Wi-Fi Protected Setup™
  - Routers after Jan. 2008 are compatible
  - Router and client handshakes then shares info
- **Browser Ad hoc configuration**
  - "EasyConfiguration"
  - Start in Ad Hoc and serve page to allow data entry
    - Can connect via any computer supporting Ad Hoc and has a browser
- **SoftAP Browser configuration**
  - Client starts AP-like network allowing connection

EasyConfig Function



MICROCHIP

Regional Training Centers

3. WiFi Demo App—Hand on

MICROCHIP
Regional Training Centers

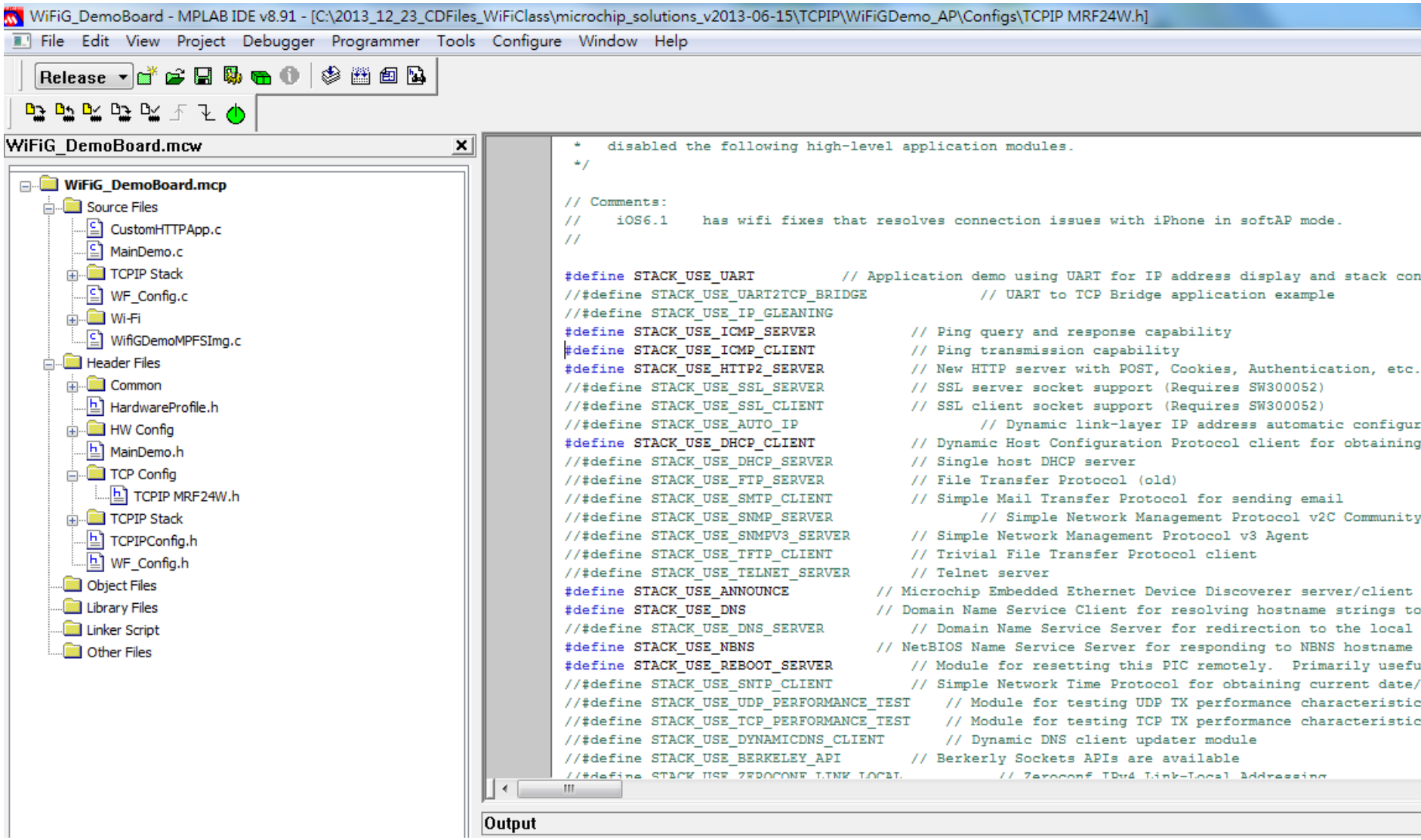
Crash Course in Networking

- **Lab 1: Change SSID**
- **Lab 2: Change Security: WEP**
- **Lab 3: Change Security: WPA/WPA2**
- **Lab 4: Change Channel**

MICROCHIP
Regional Training Centers

Stack Configurations

TCPIP MRF24W.h

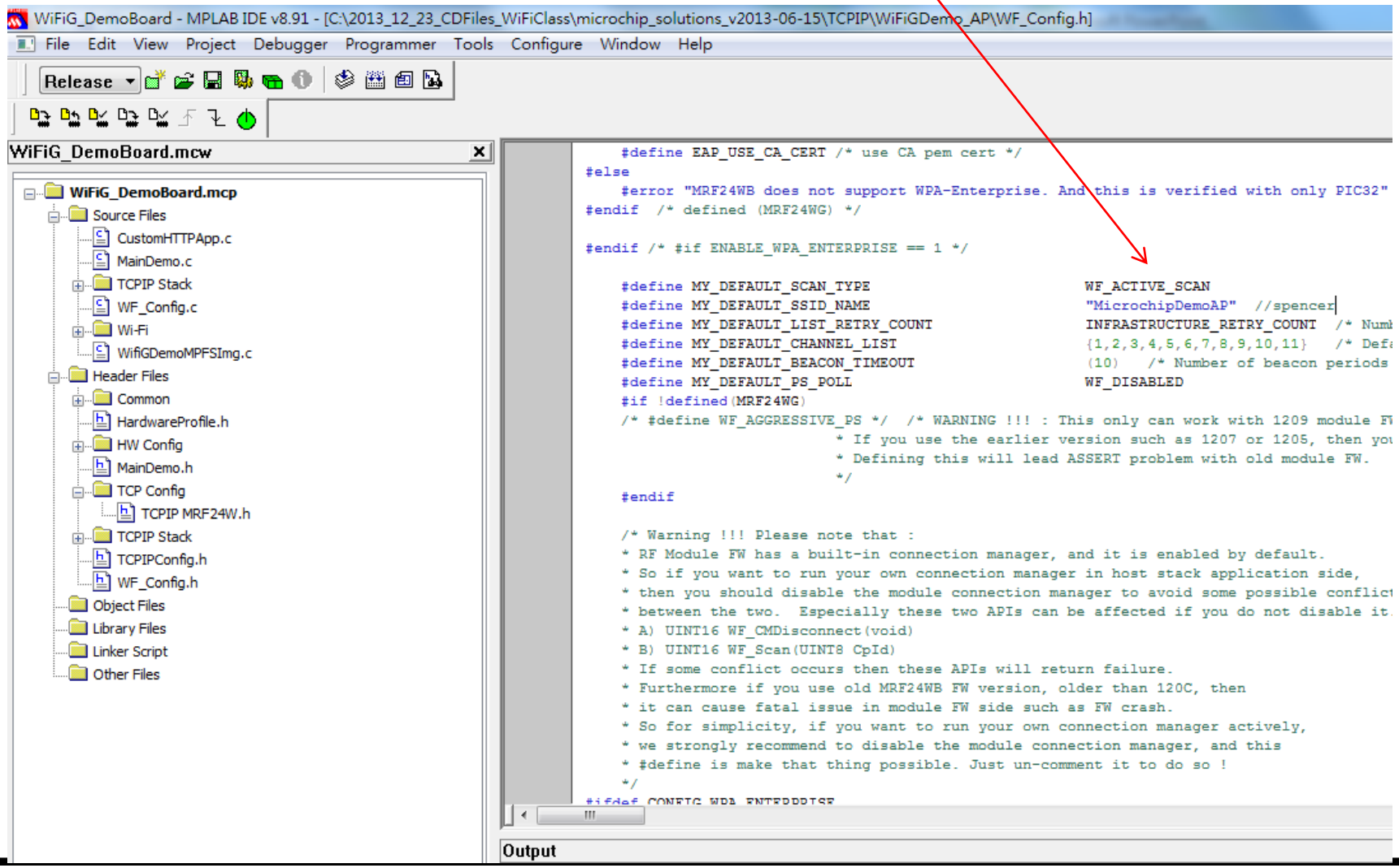


WiFi® Configurations

Regional Training Centers

WF_Config.h

Change SSID



WiFi® Configurations

WF_Config.h

C:\MAL\May2010\TCPIP WiFi Demo App\WF_Config.h

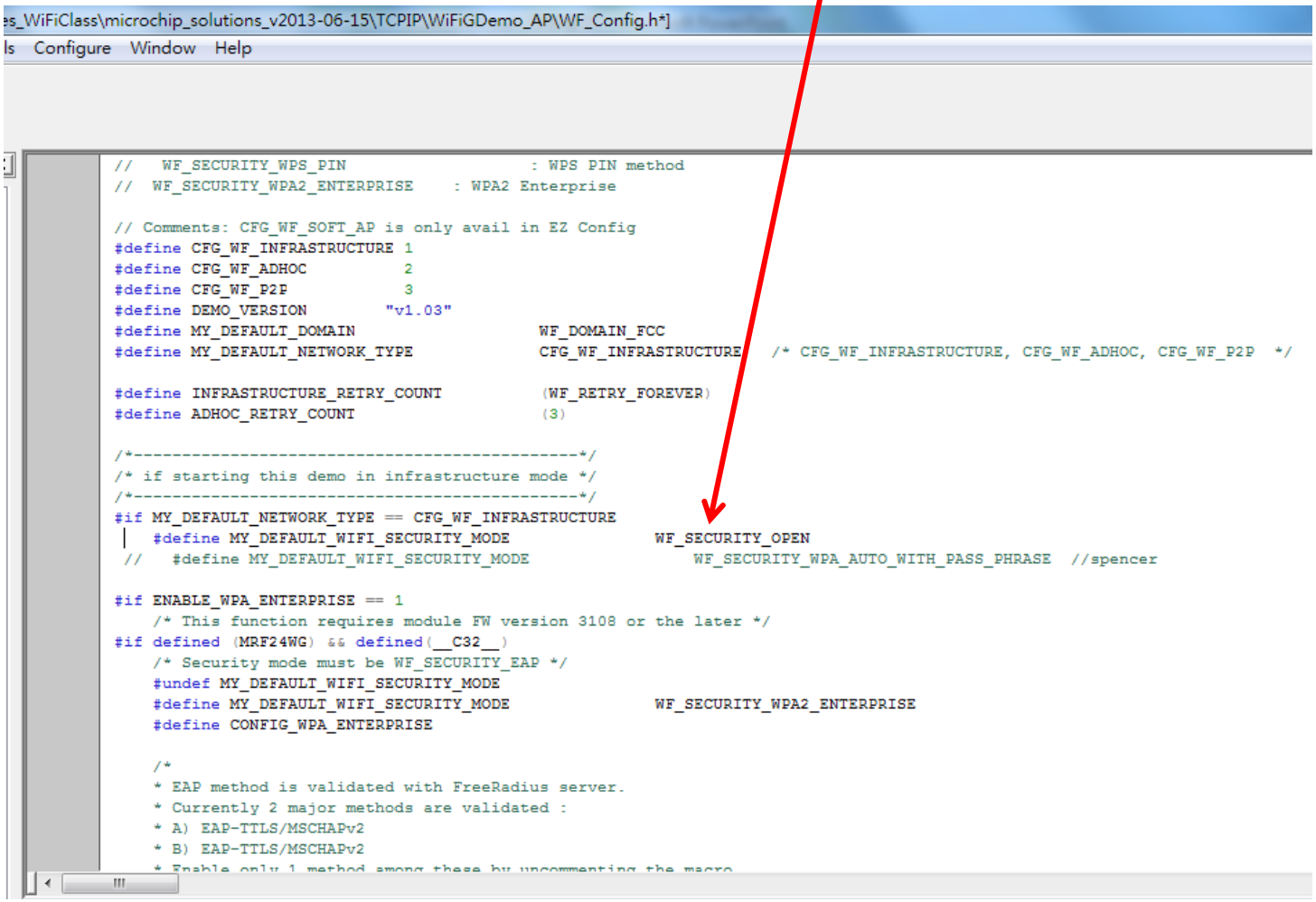
```
/*****************************************************************************/
/*****************************************************************************/
/*                              WIFI SECURITY COMPILE-TIME DEFAULTS           */
/*****************************************************************************/
/*****************************************************************************/
// Security modes available on WiFi network:
//   WF_SECURITY_OPEN                       : No security
//   WF_SECURITY_WEP_40                     : WEP Encryption using 40 bit keys
//   WF_SECURITY_WEP_104                    : WEP Encryption using 104 bit keys
//   WF_SECURITY_WPA_WITH_KEY               : WPA-PSK Personal where binary key is given to MRF24WB0M
//   WF_SECURITY_WPA_WITH_PASS_PHRASE       : WPA-PSK Personal where passphrase is given to MRF24WB0M and it c
//   WF_SECURITY_WPA2_WITH_KEY              : WPA2-PSK Personal where binary key is given to MRF24WB0M
//   WF_SECURITY_WPA2_WITH_PASS_PHRASE      : WPA2-PSK Personal where passphrase is given to MRF24WB0M and it
//   WF_SECURITY_WPA_AUTO_WITH_KEY          : WPA-PSK Personal or WPA2-PSK Personal where binary key is given
//                                            connect at highest level AP supports (WPA or WPA2)
//   WF_SECURITY_WPA_AUTO_WITH_PASS_PHRASE : WPA-PSK Personal or WPA2-PSK Personal where passphrase is given
//                                            calculates the binary key and connects at highest level AP sup
```

Security Type

WiFi® Configurations

Security Details

WF_Config.h



MICROCHIP
Regional Training Centers

Configure Infrastructure

- Demo Applications are pre-configured
- To change defaults, (erase EEPROM)
- To change: security, network

MICROCHIP
Regional Training Centers

Crash Course in Networking

- Lab 1: Change SSID
- Lab 2: Change Security: WEP
- Lab 3: Change Security: WPA/WPA2
- Lab 4: Change Channel

MICROCHIP

Regional Training Centers

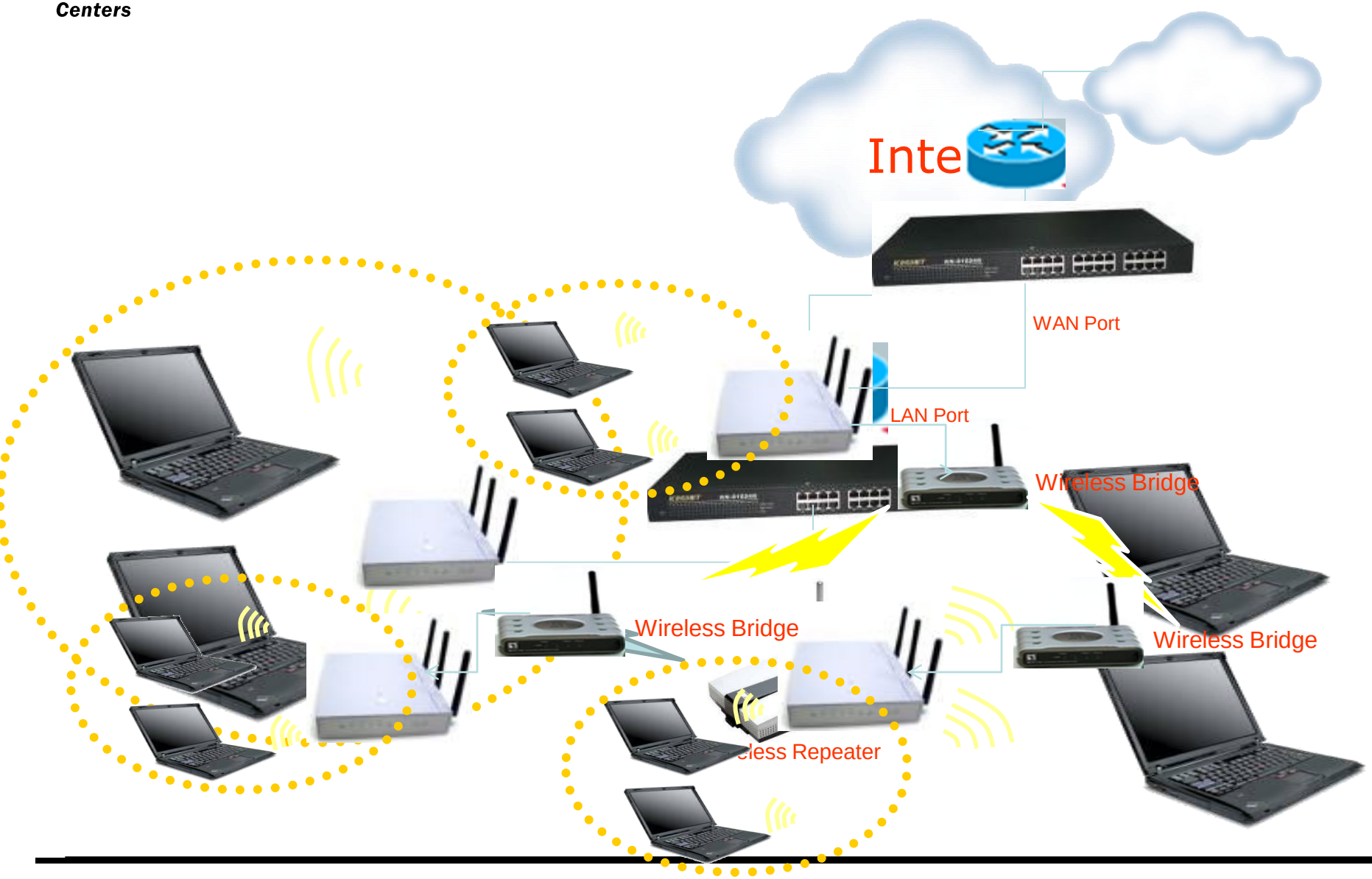
4. Wireless Network Deployment & Wireless AP Feature

MICROCHIP
Regional Training Centers

Wireless Network Deployment

- Root Mode
- Repeater Mode
- Bridge Mode

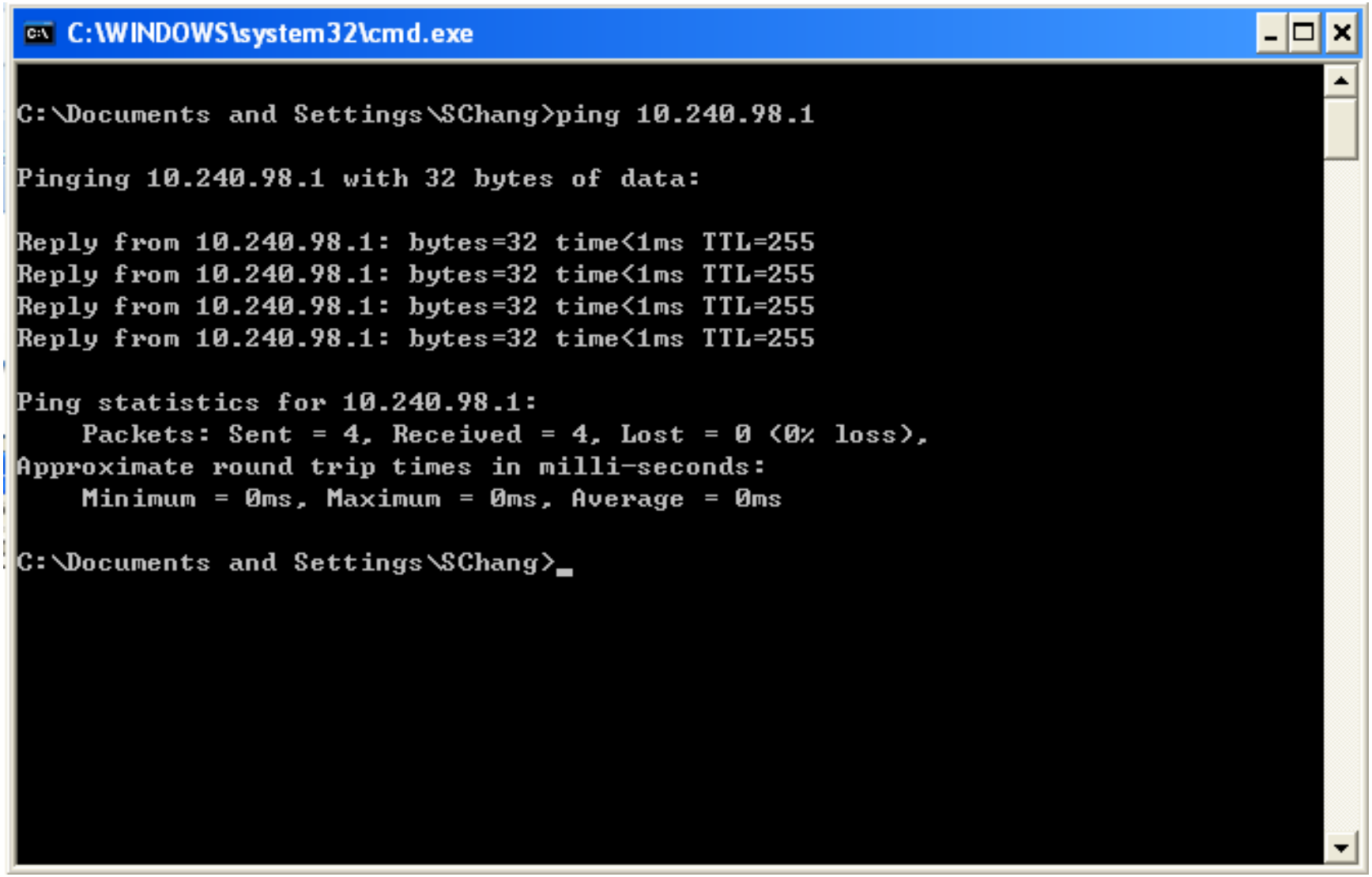
Wireless Network Deployment



Wireless Network Deployment

802.11 Network Deployment

Back

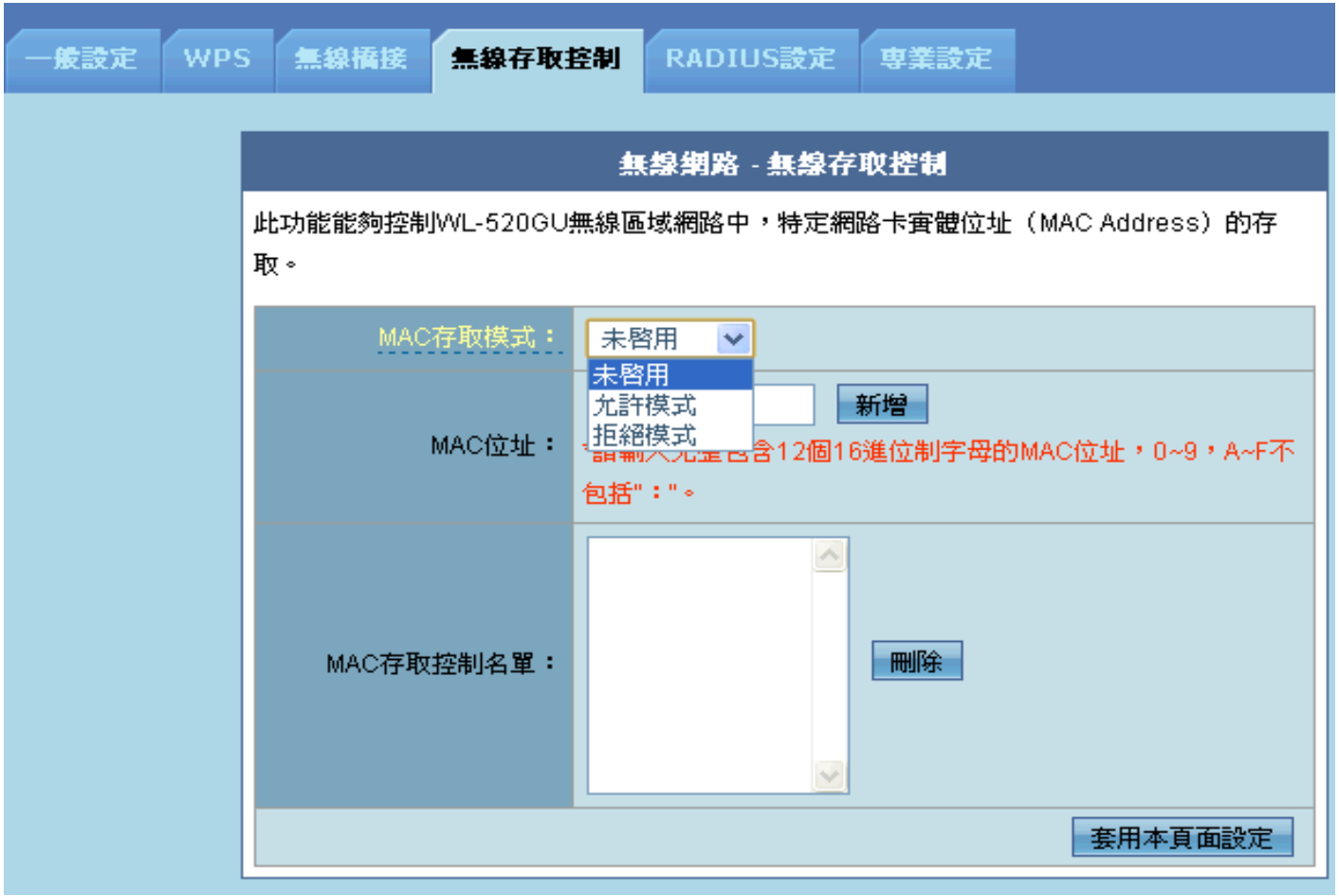


MICROCHIP
Regional Training Centers

Wireless AP Features

- MAC Address Control
- Bandwidth Management
- Port Trigger
- Virtual Server
- Dynamic DNS
- FireWall
- 3G Broadband Router
- Power-over-Ethernet

MAC Address Control



Bandwidth Management

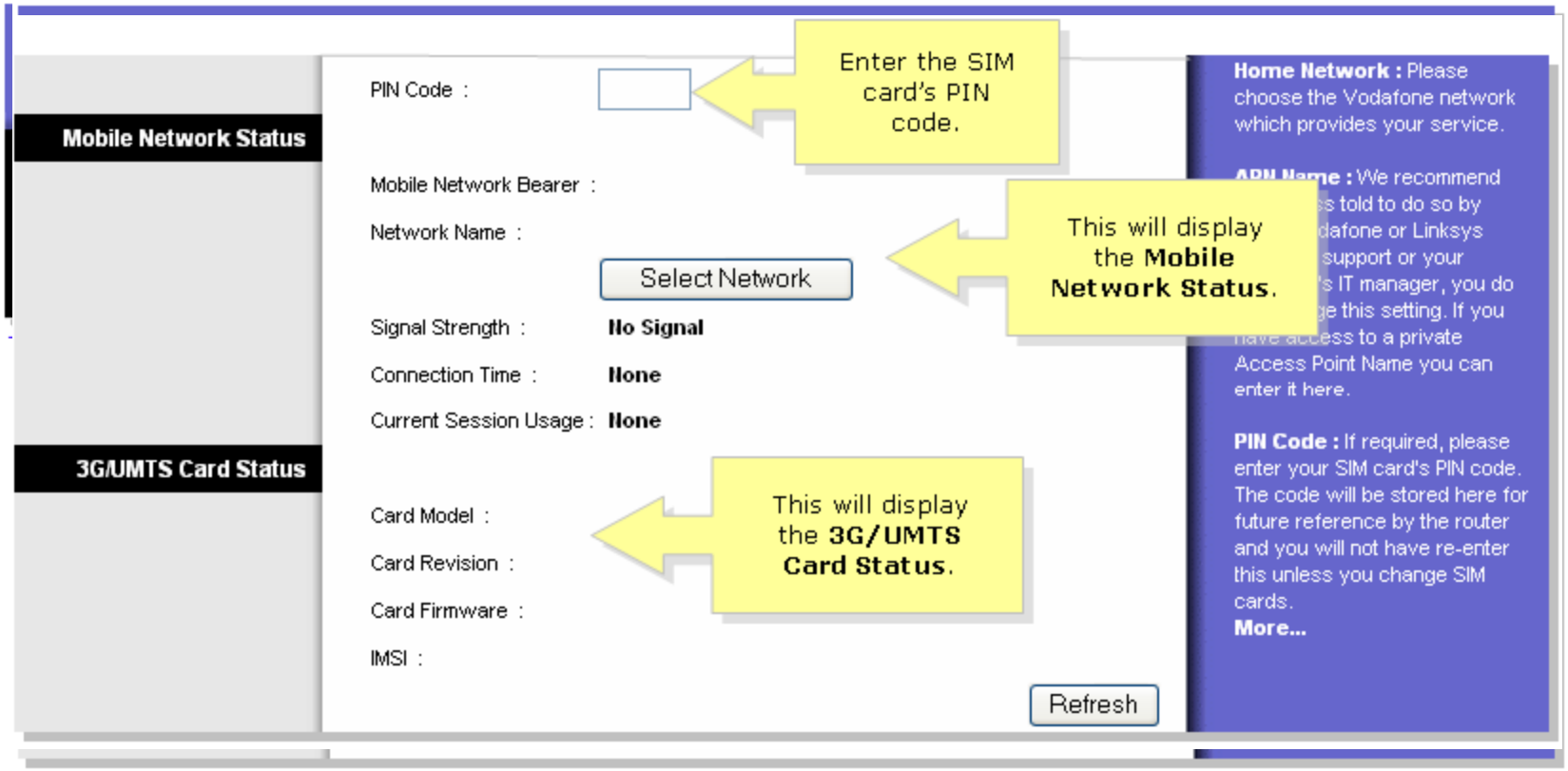
Port Trigger

Virtual Server

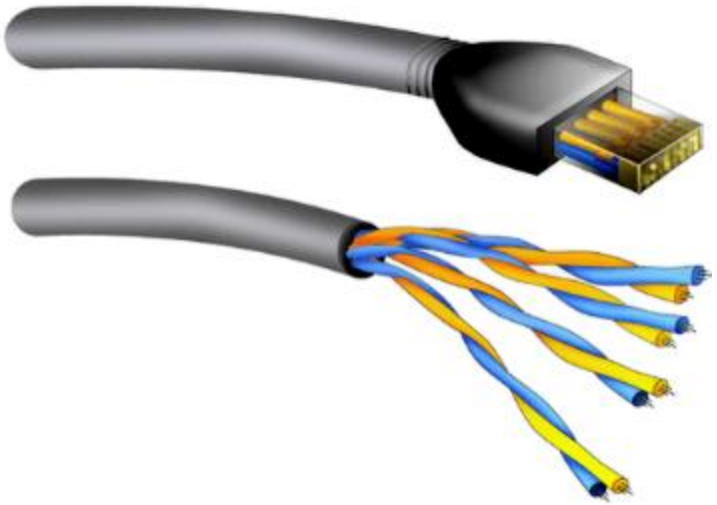
DDNS

FireWall

3G Broadband Router



Power-over-Ethernet



PINS on Switch	10/100 DC on Spares
Pin 1	Rx +
Pin 2	Rx -
Pin 3	Tx +
Pin 4	DC +
Pin 5	DC +
Pin 6	Tx -
Pin 7	DC -
Pin 8	DC -

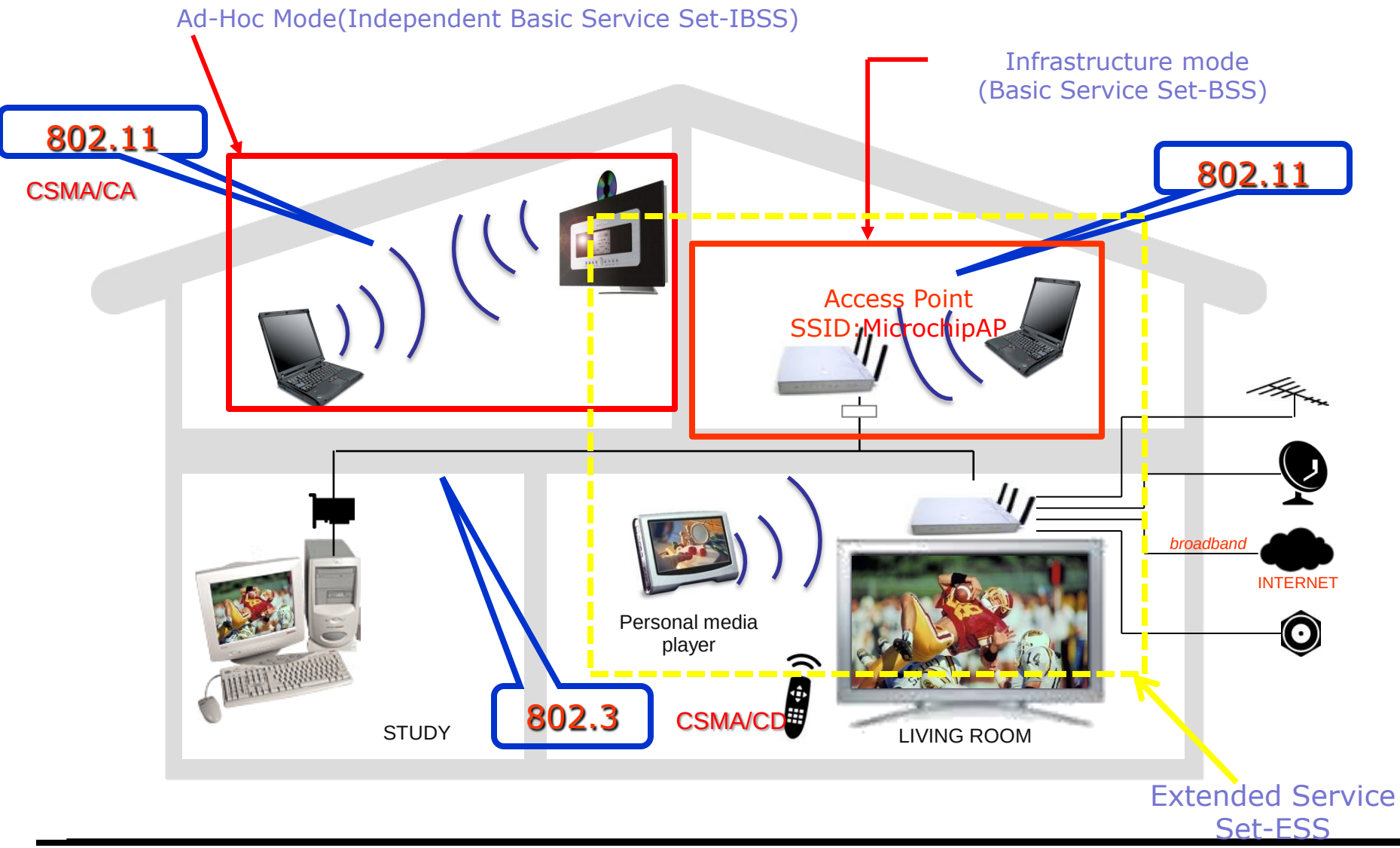
Wireless AP Setting (Handon)

MICROCHIP

Regional Training Centers

5. Overview of 802.11 Networks

Overview of 802.11 Network

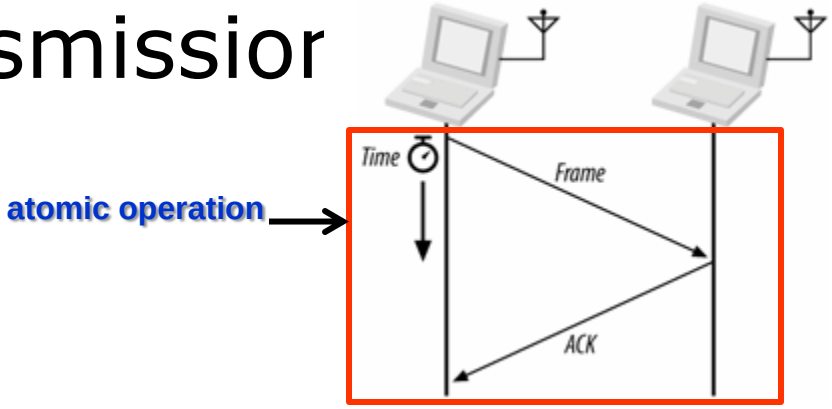


802.11 Media Access and Frame Control

- Challenges for the MAC
- Frame Transmission & Association States
  - Frame types of 802.11 protocol
  - 802.11 Frame Format in Detail

Challenges for the MAC

- Positive acknowledgment of data transmissior



ACK is required

Figure- Positive acknowledgment of data transmissions

- The Hidden Node Problem

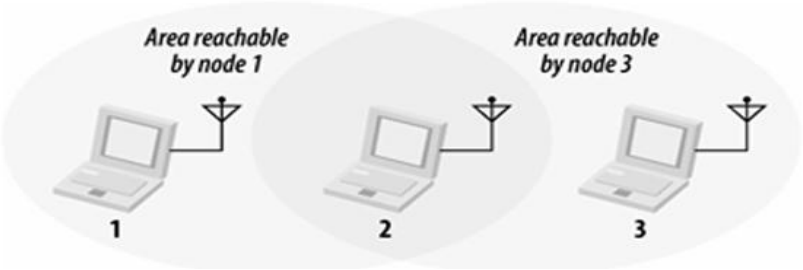
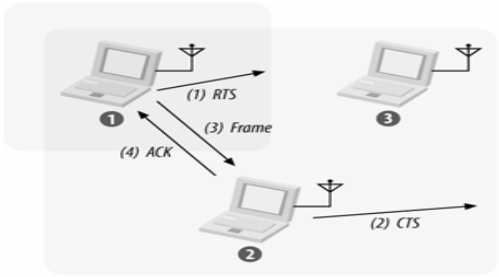


Figure- Nodes 1 and 3 are "hidden"



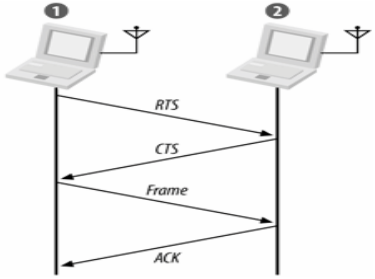
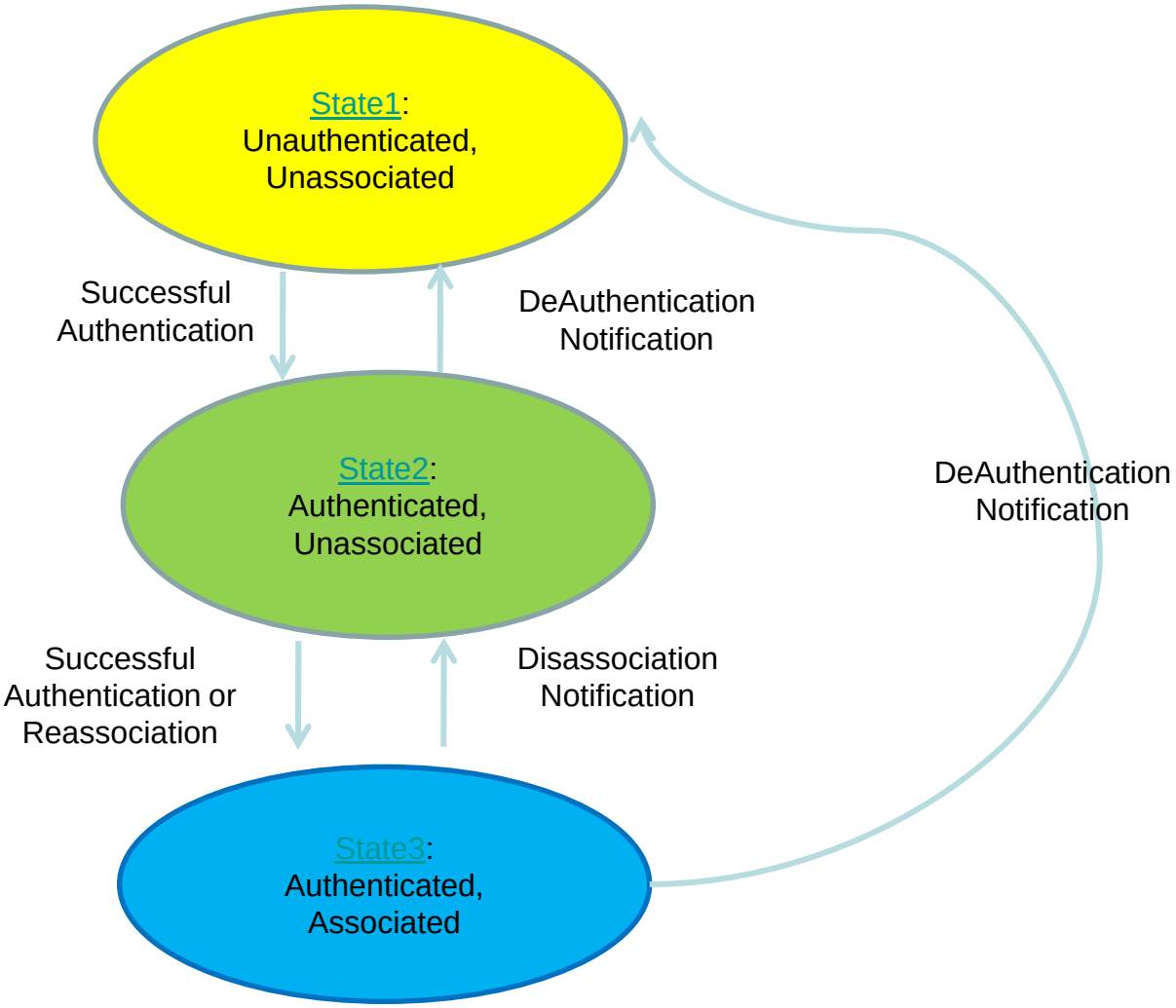


Figure- RTS/CTS clearing

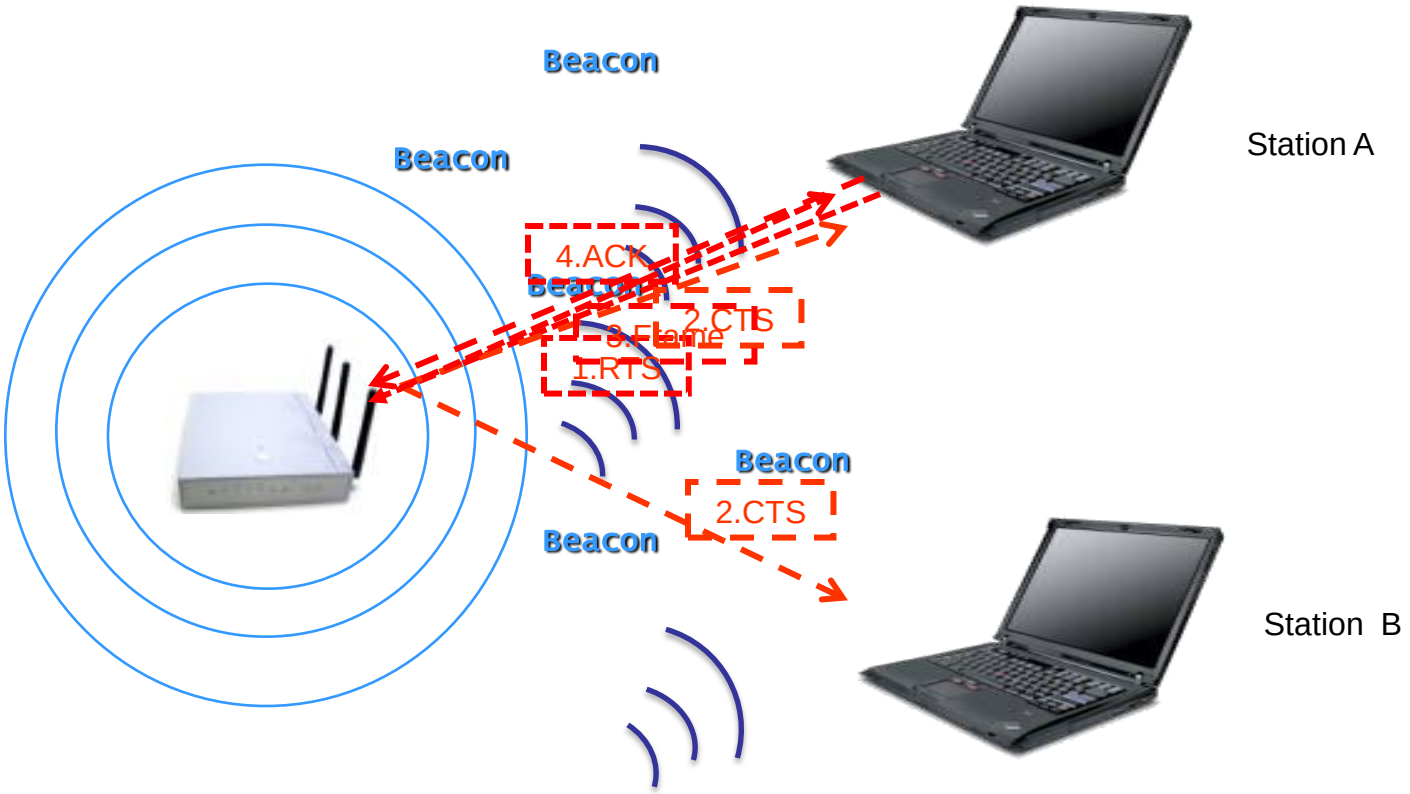
Frame Transmission & Association States

- Station connect to AP(Access Point)



Frame Transmission & Association States

- State 1(Unauthenticated, Unassociated)



Beacon-Management Frame

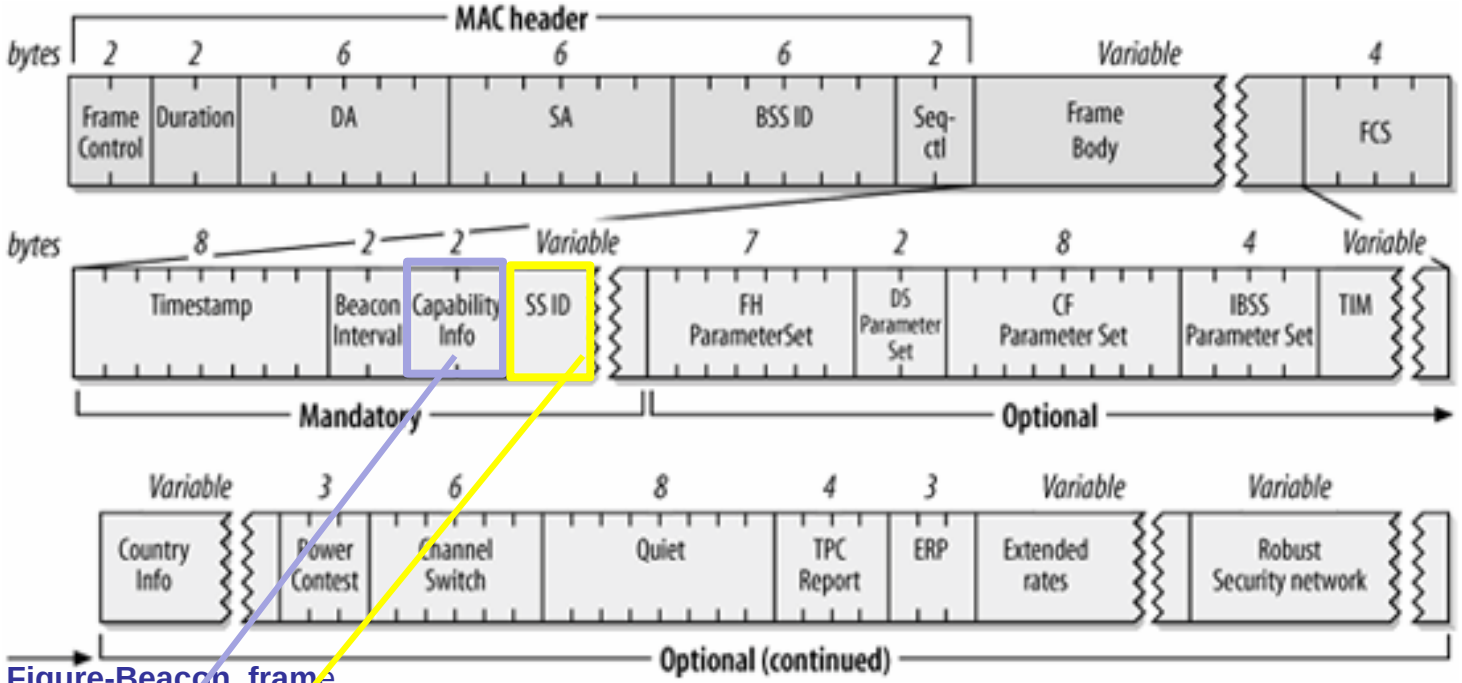
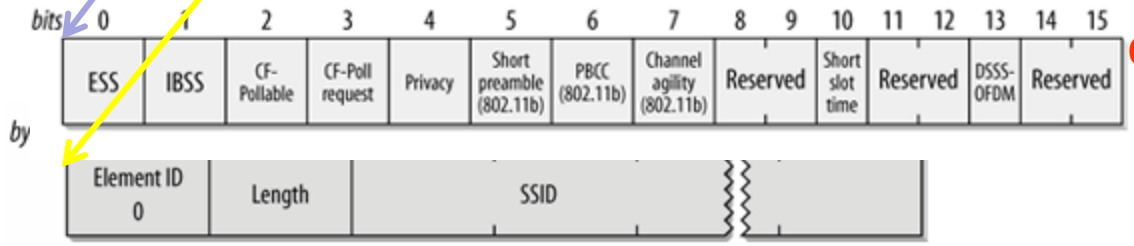


Figure-Beacon frame



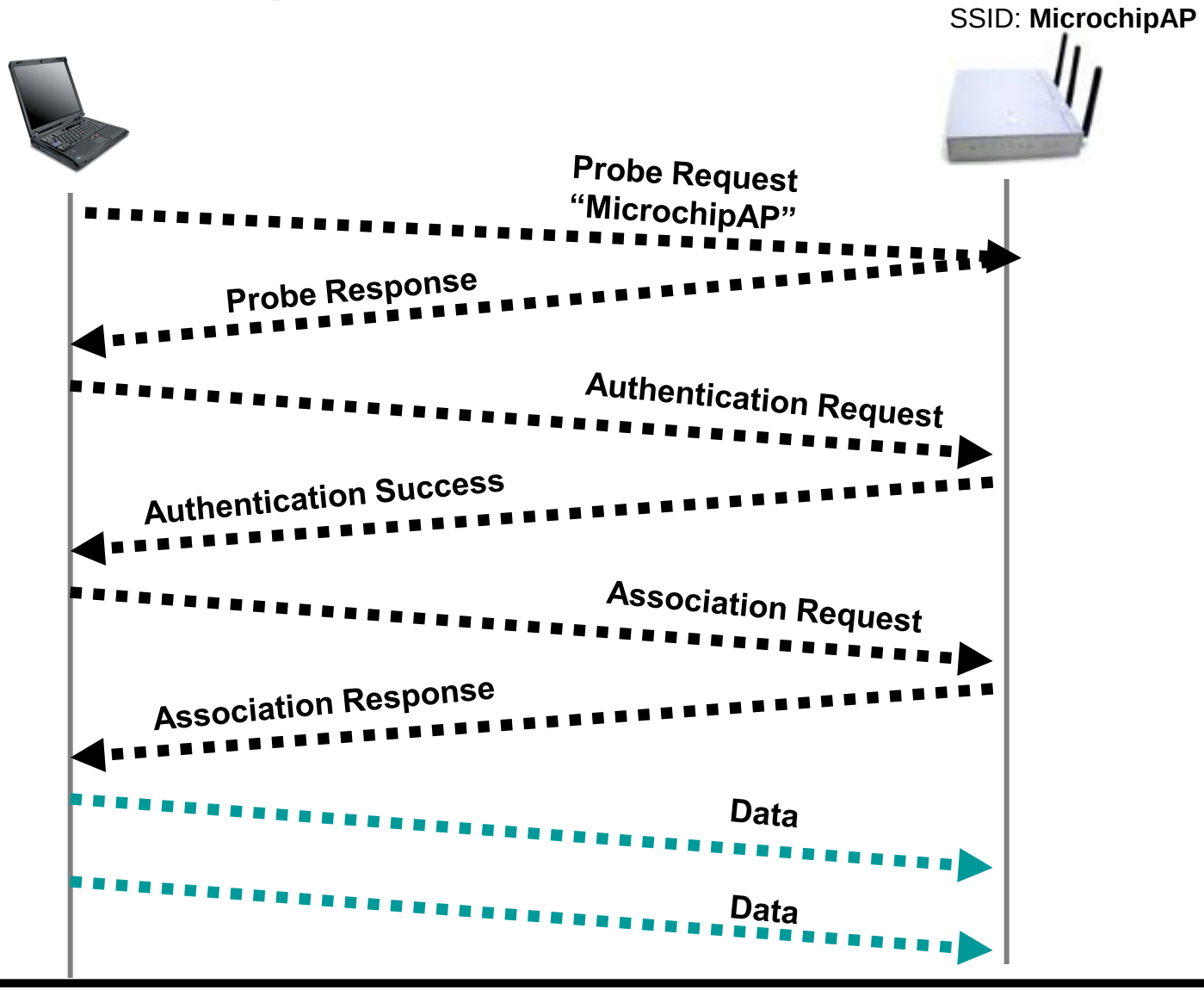
Capability Information field

* ESS/IBSS:1/0 -> *Infrasturce mode*
  0/1-> Ad-Hoc mode
* Privacy=1, WEP required

Figure-Service Set Identity information element

*Most products require that the string be a garden variety, null-terminated ASCII string. The length of SSID ranges 0 ~ 32 bytes.

Probe Request/Response (Management Frame)

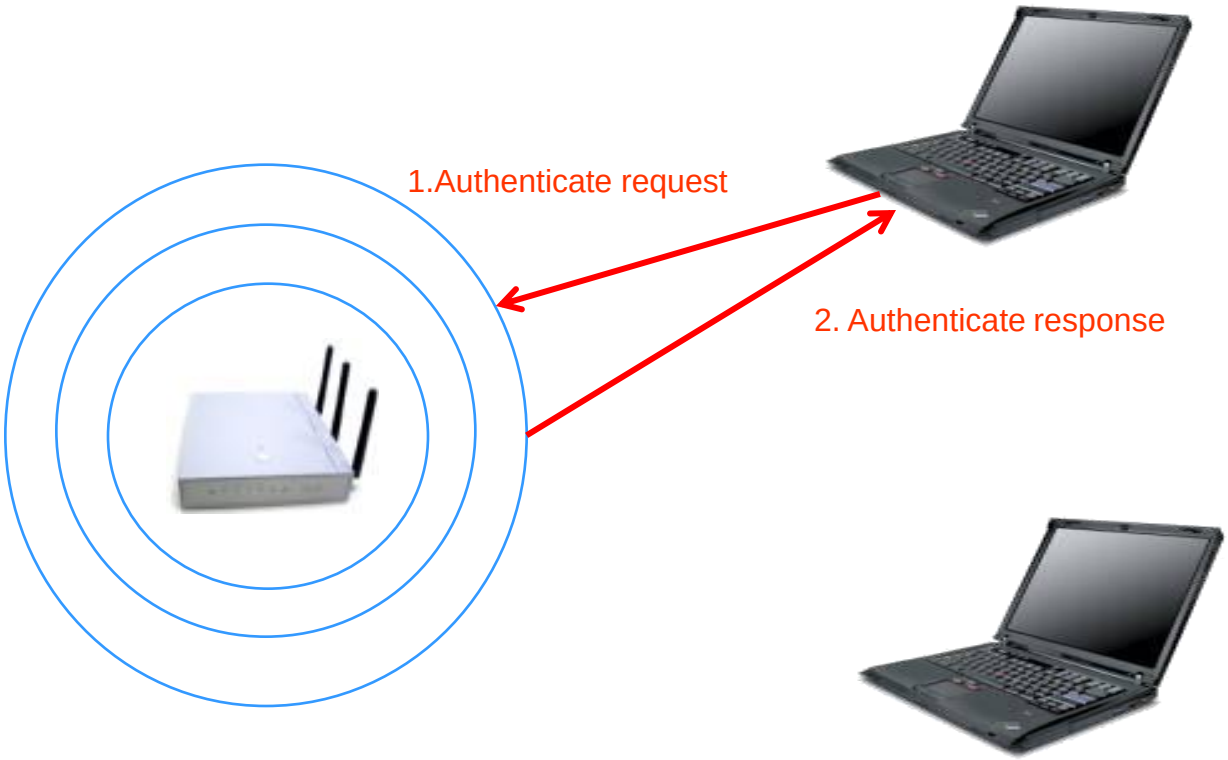


Back State2

802.11 Frame State

Back

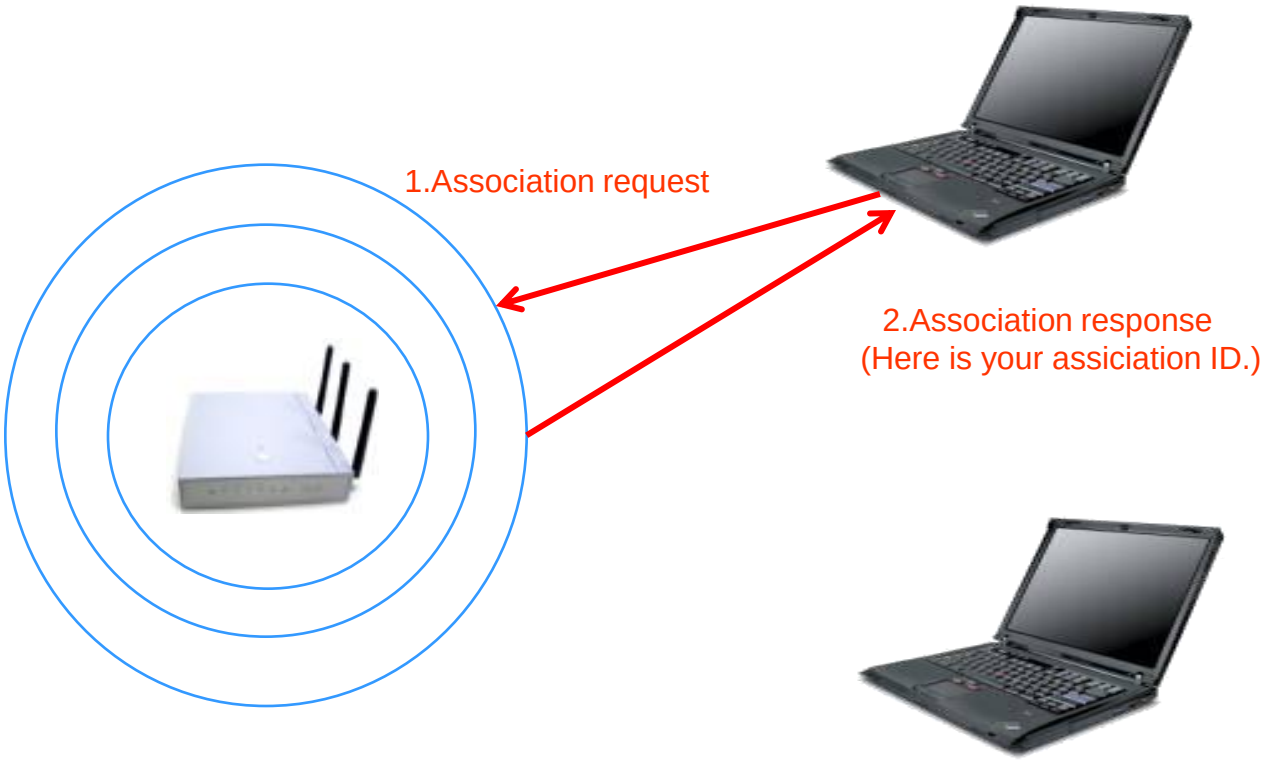
- State 2(Authenticated, Unassociated)



802.11 Frame State

Back

- State 3(Authenticated, Associated)



Disassociation and de-authentication (Management Frame)

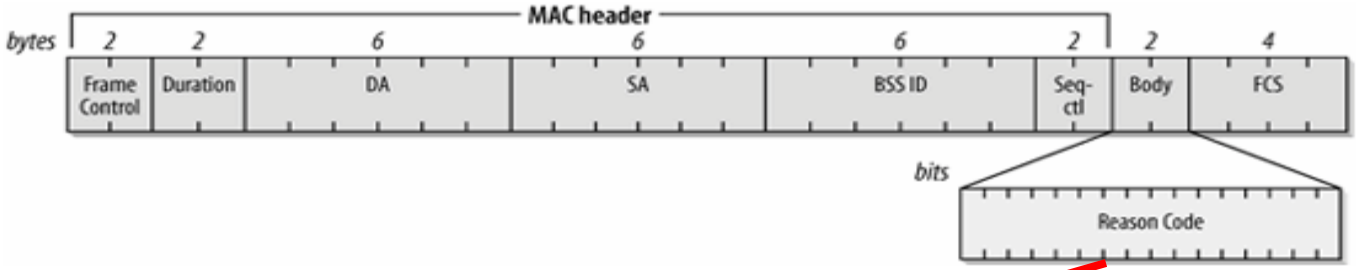


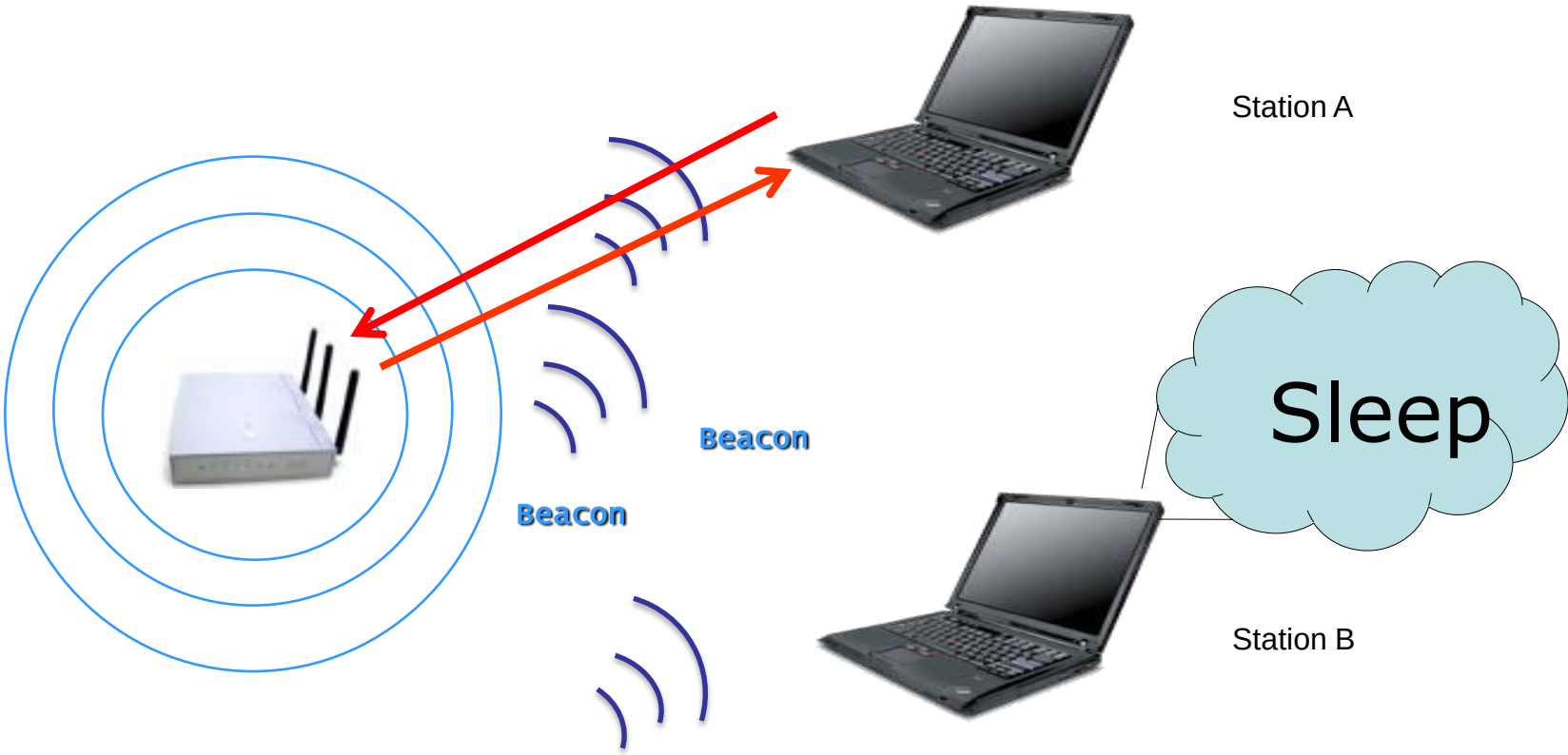
Figure-Disassociation and Deauthentication frames

Code	Explanation
3	Prior authentication is no valid
4	Inactivity time expired and station was disassociated
5	Disassociated due to insufficient resources at the access point

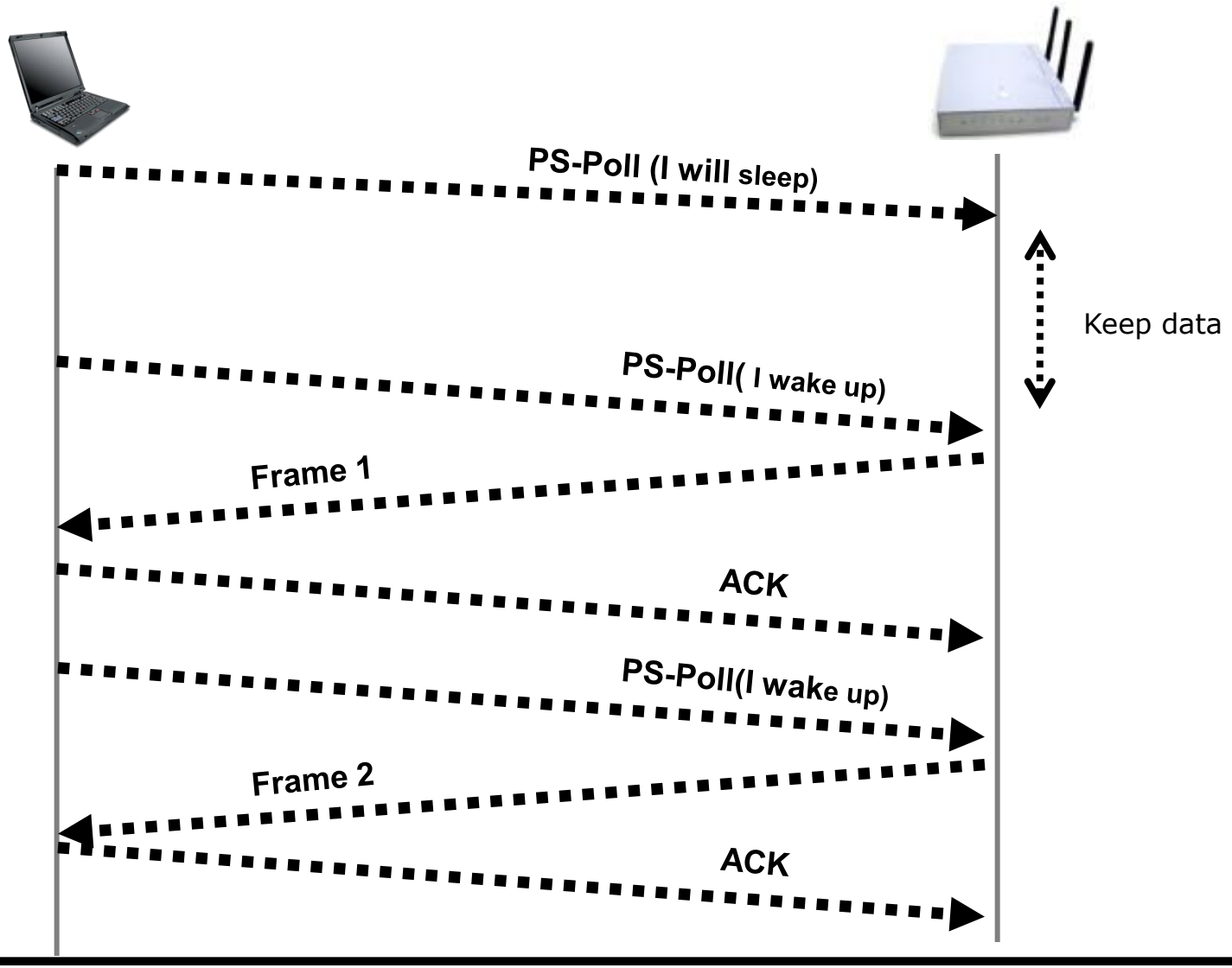
Back State3

Frame Transmission & Association States

- State 3(Authenticated/Associated)



Power-Save Poll (PS-Poll) Control Frame



MICROCHIP

Regional Training Centers

Low Power Operation

Power Management

MICROCHIP
Regional Training Centers

Power Management

- There are 4 modes of operation
  - Active (On)
    - Transmitting or receiving
    - Few hours battery life with Rx on
  - 802.11PS
    - Maintains connection, transparent disconnect
    - 2 months battery life
  - Saved State
    - Connection info stored after first commissioning
    - Can reconnect transmit and sleep in 100mS
    - 1+ year battery life
  - Hibernate
    - Disconnected, session lost
    - 2+ years battery life

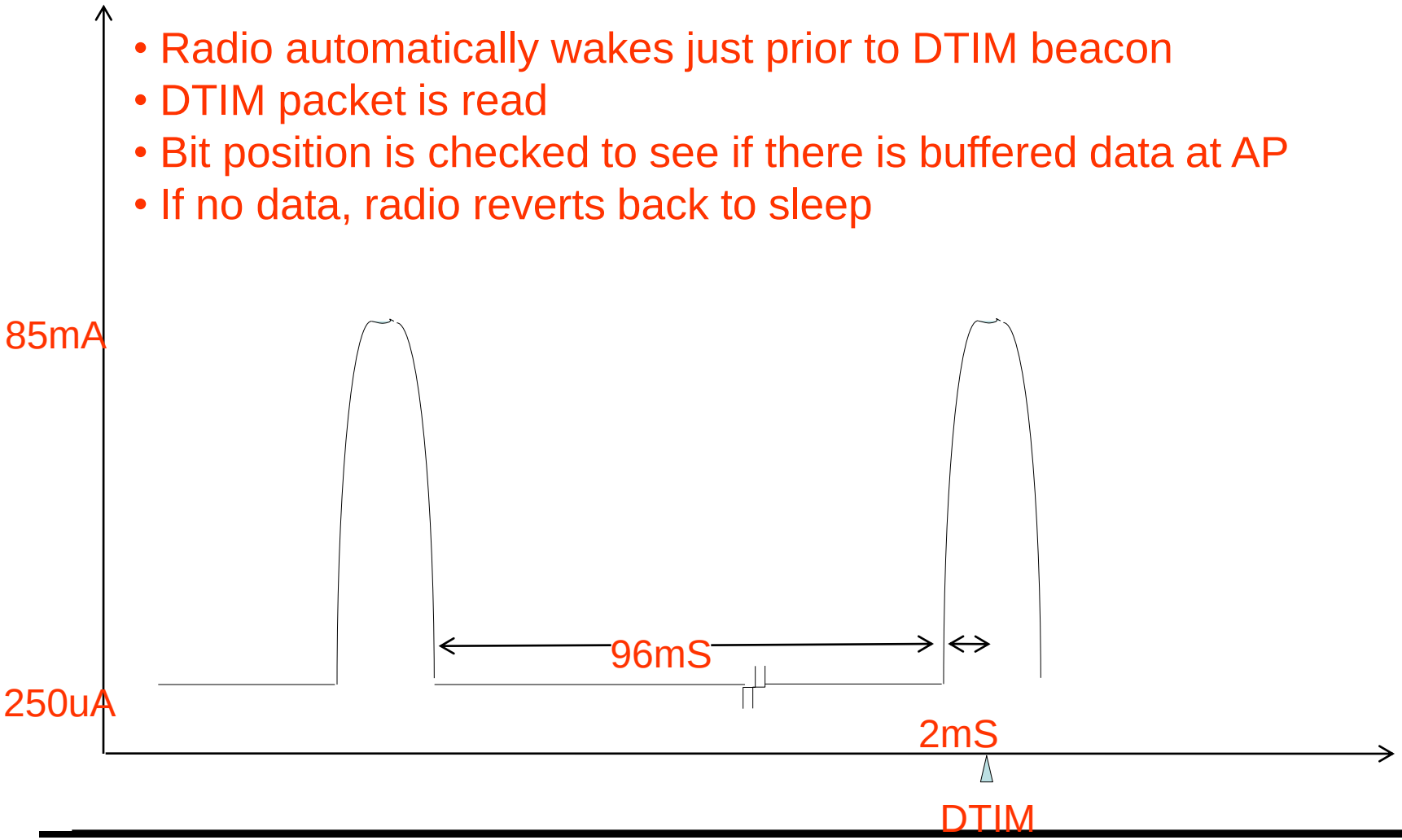
MICROCHIP
Regional Training Centers

802.11 PS

- Power Save or PS-Poll
- Use and limits
  - Provides virtual “always on” power saving mode
  - Very low latency to send and receive
  - Requires PS supported Access Point
- Power example
  - Case: always connected w/sleep, hourly access for 500B transfer
    - 2 week battery life (just radio)

802.11PS

- Radio automatically wakes just prior to DTIM beacon
- DTIM packet is read
- Bit position is checked to see if there is buffered data at AP
- If no data, radio reverts back to sleep



MICROCHIP
Regional Training Centers

Saved State

- Use and limits
  - After first connection, saves configuration data in NVM
  - Provides very low power fast on/off activity
  - Very low latency (100ms) to restart, send turn off
  - Many routers will not allow indefinite sleep
  - Best for sending data more often than every minute
- Power
  - Can achieve years of battery life for hourly connection with dynamic IP
  - RN Sleep: 4uA w/ RTC & Active Sensors

MICROCHIP
Regional Training Centers

Hibernate

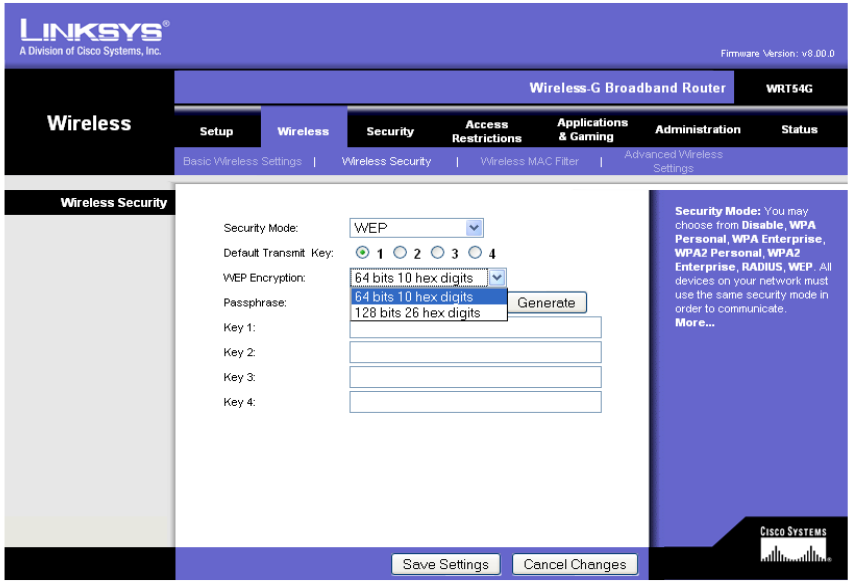
- **Hibernate use and limits**
  - **Radio is turned off on-chip**
  - **Need to reconnect when turned on**
  - **Takes 1-10s to restart**
- **Power example**
  - **Case: unit powers on hourly for 500B transfer, static IP**
    - **MRF24WB0M: Hibernate/sleep**
      - **4 year dual AA battery life (just radio)**

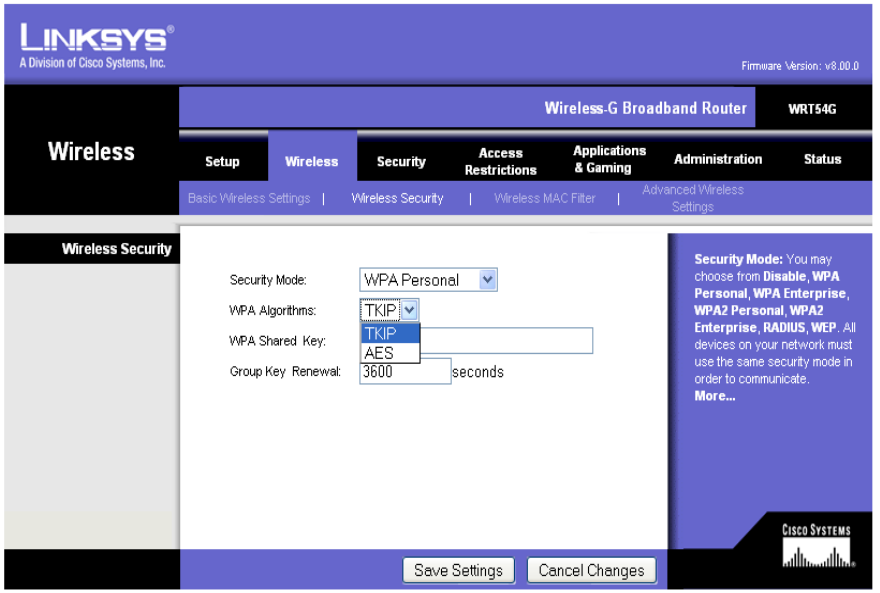
IEEE 802.11 Security

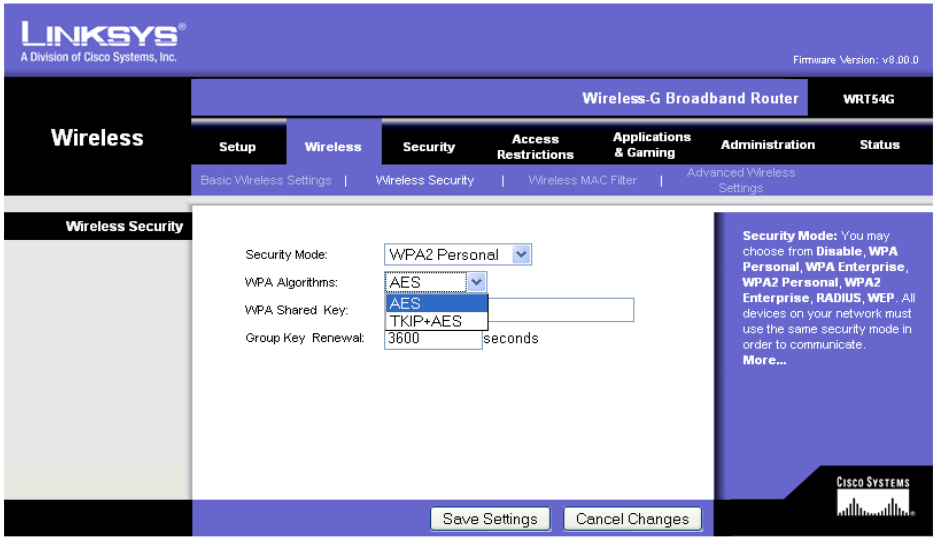
- Wireless Security Setting on Access Point
- Two subsystems:
  - A data encapsulation technique called WEP (Wired Equivalent Privacy)
  - An authentication algorithm called Shared Key Authentication
- Severe security weakness in WEP.
- WPA(Wi-Fi Protected Access), WPA2

MICROCHIP
Regional Training Centers

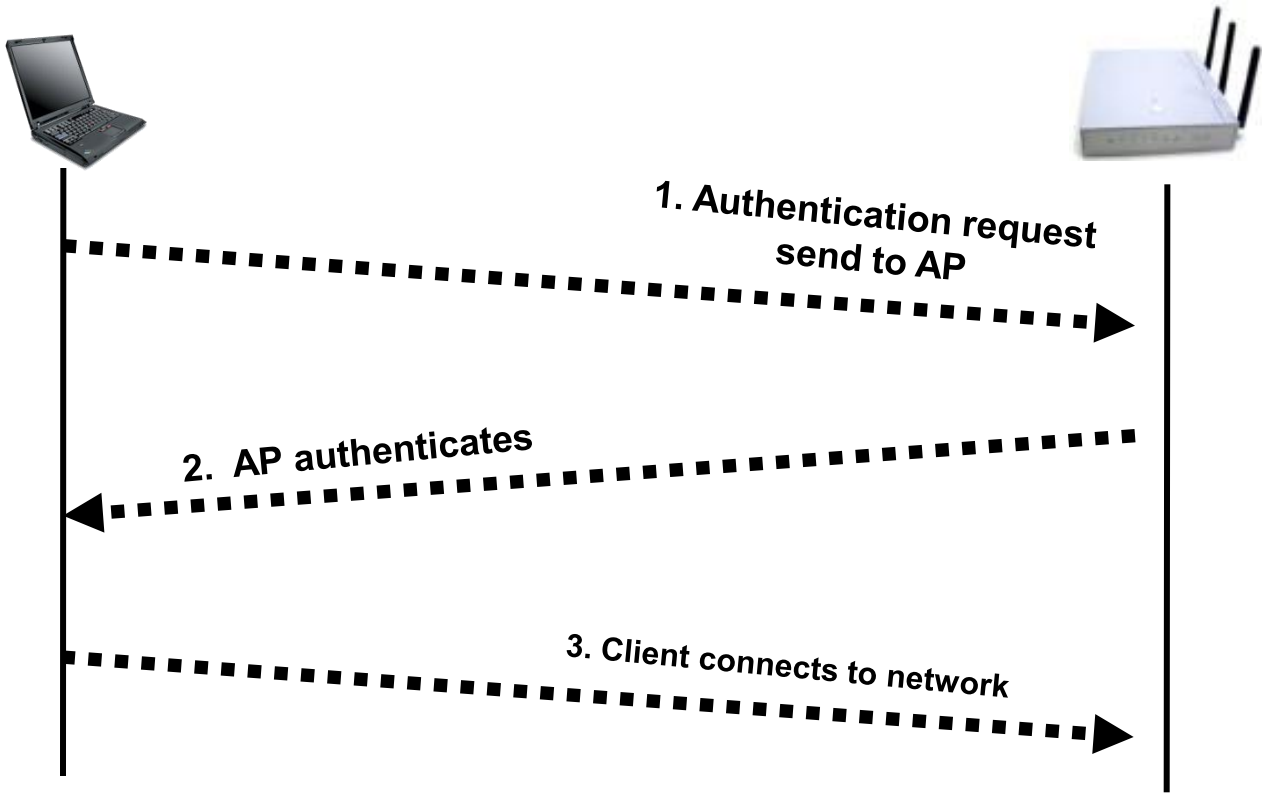
Wireless Security Setting



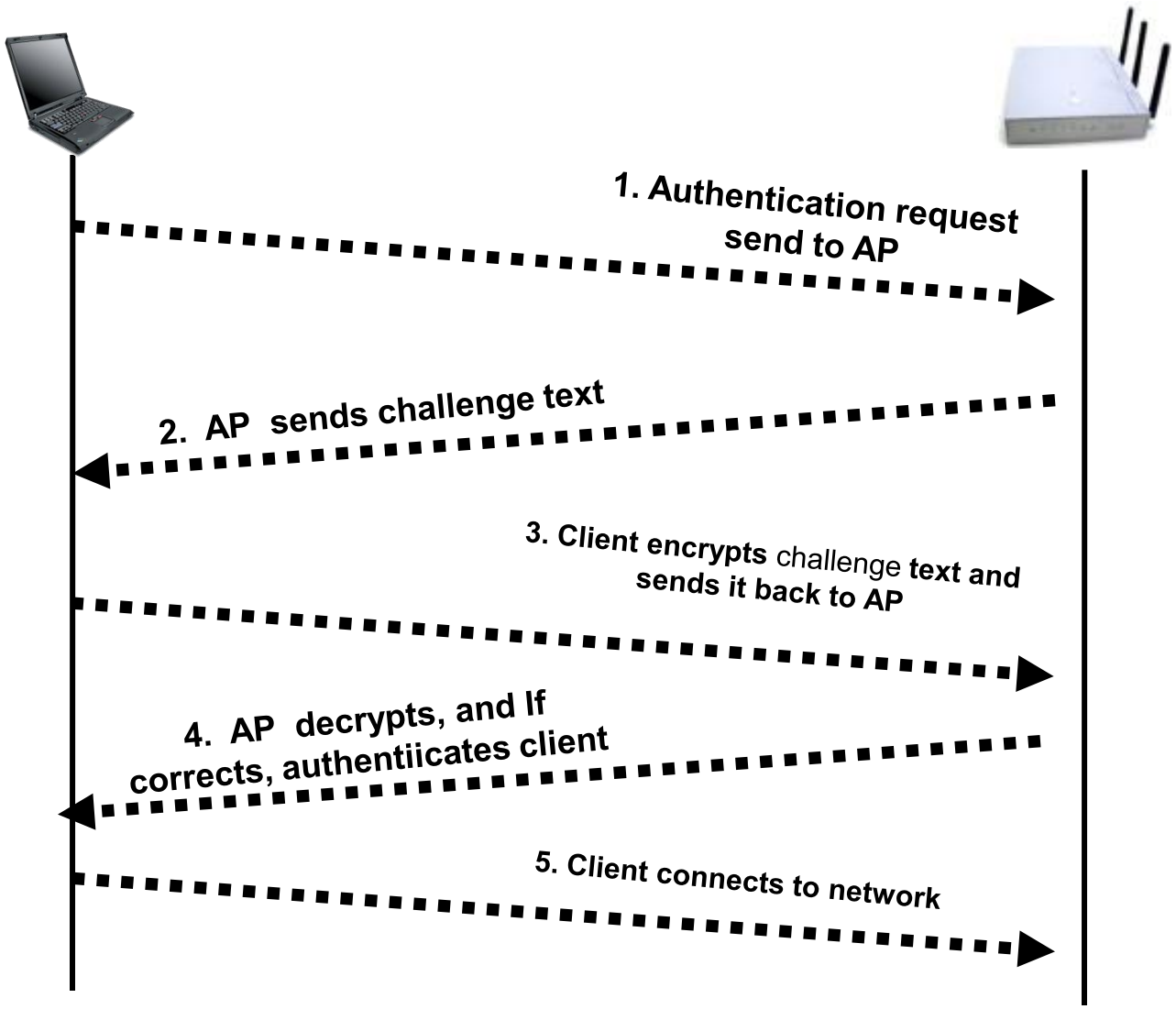




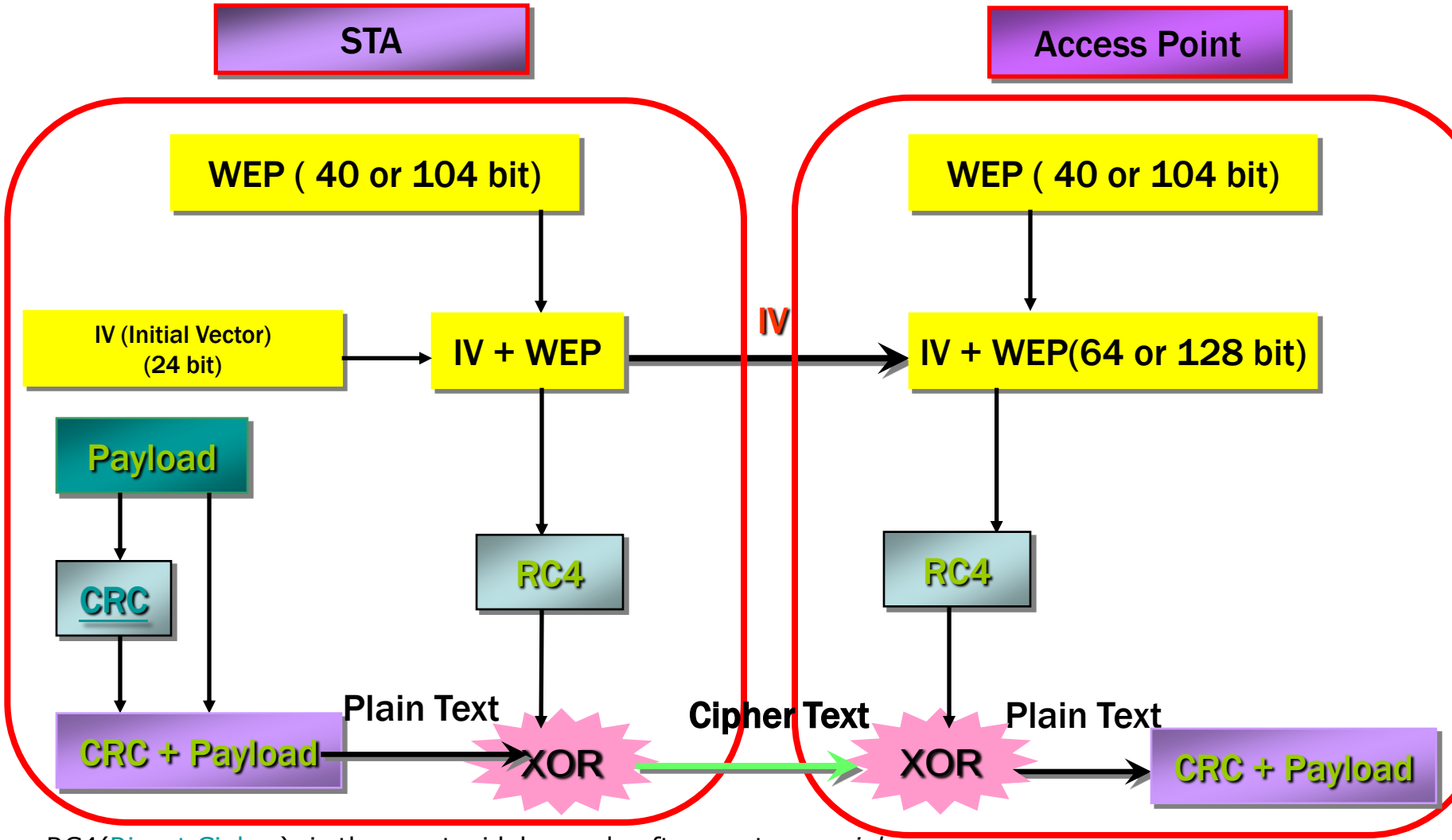
Open System Authentication



WEP Shared Key Authentication



WEP-Encryption/Decryption



RC4(Rivest Cipher), is the most widely-used software stream *cipher* .
IV: 24 bit initialization vector,2^24= 16777216

MICROCHIP
Regional Training Centers

WEP Weaknesses

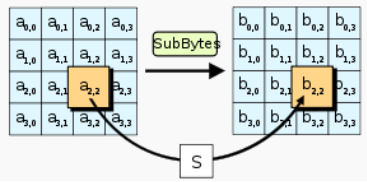
- The IV is too small and in cleartext.
- The IV is static (0→ 16777216)
- The IV makes the key stream vulnerable.
- The IV is a part of the RC4 encryption key
- WEP provides no cryptographic integrity protection

WEP VS WPA/WPA2

	WEP	WPA	WPA2
Cipher	RC4	RC4	AES
Key Size	40 bits	128 bits	128 bits
Key Life	24-bit IV	48-bit IV	48-bit IV
Data Integrity	CRC-32	Michael	CCM
Header Integrity	None	Michael	CCM

WI-FI Alliance: http://www.wi-fi.org/knowledge_center/webcast-wpa-061103/

AES(Advanced Encryption Standard)



Michael: Message Integrity Code(MIC)

CCM（Counter-Mode/CBC-MAC)

Electromagnetic Spectrum

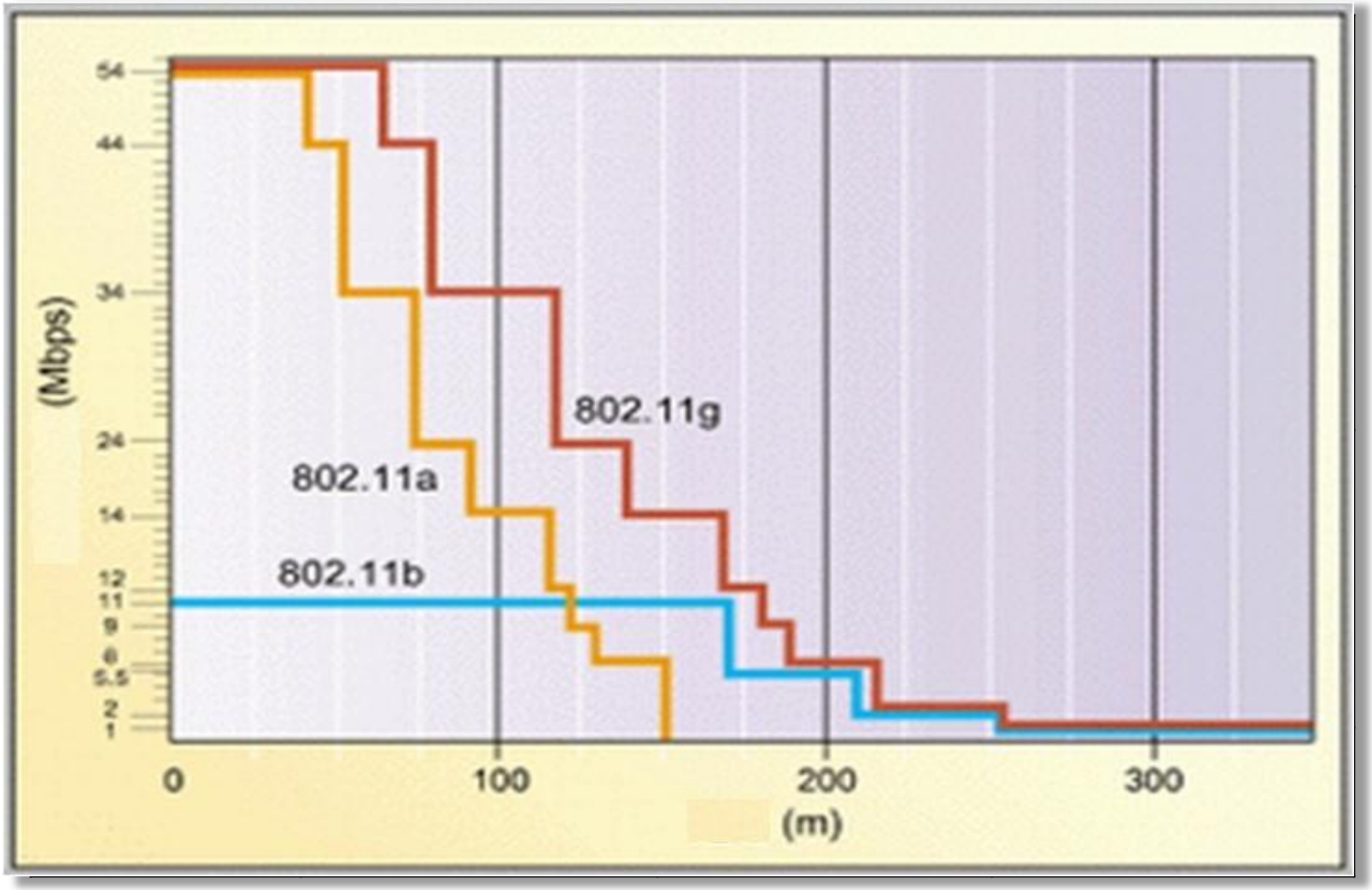
European Union (Spain and France not included)

		Regulatory domains					
CHNL_ID	Frequency	USA domain= FCC	Canada domain= IC	domain= ETSI	domain= Spain	domain= France	domain= JapanA/B
1	2412 MHz	X	X	X	—	—	—
2	2417 MHz	X	X	X	—	—	—
3	2422 MHz	X	X	X	—	—	—
4	2427 MHz	X	X	X	—	—	—
5	2432 MHz	X	X	X	—	—	—
6	2437 MHz	X	X	X	—	—	—
7	2442 MHz	X	X	X	—	—	—
8	2447 MHz	X	X	X	—	—	—
9	2452 MHz	X	X	X	—	—	—
10	2457 MHz	X	X	X	X	X	—
11	2462 MHz	X	X	X	X	X	—
12	2467 MHz	—	—	X	—	X	—
13	2472 MHz	—	—	X	—	X	—
14	2484 MHz	—	—	—	—	—	X

JapanB=domain

JapanA=domain

Comparison of 802.11 Protocol



MICROCHIP

Regional Training Centers

6.Microchip TCP/IP Stack (Software Architecture)

MICROCHIP
Regional Training Centers

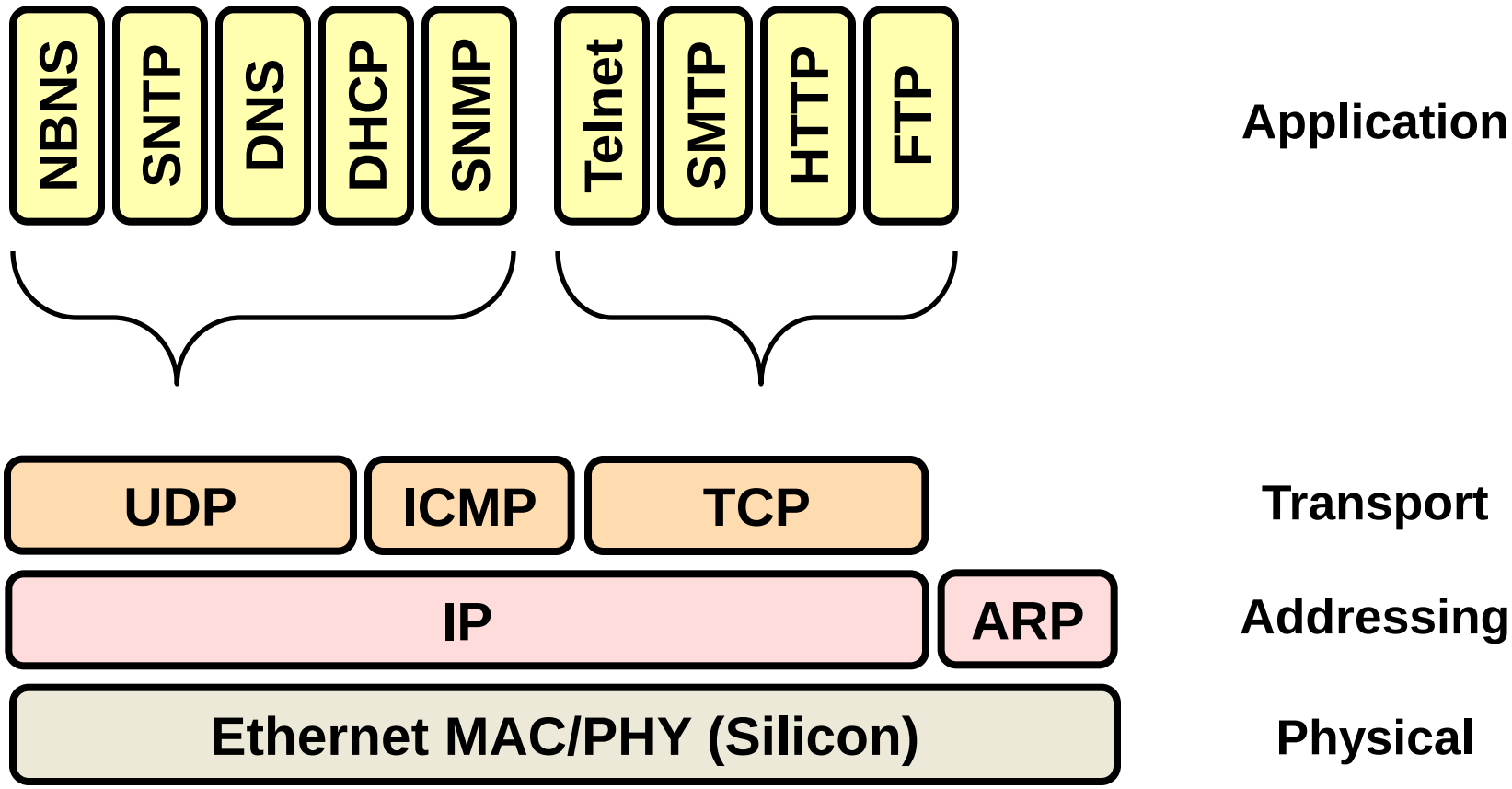
Microchip's TCP/IP Protocol Stack

- Component of the Microchip TCPIP Framework
- FREE!! Royalty-free, no license fees
- Latest version available for download at: www.microchip.com/mla
- Modular design
  - Compile only what you need
- No lower layer design required within the TCP/IP protocol
- Optimized for PIC18, PIC24, dsPIC® DSCs and PIC32
  - Includes BSD RTOS support via a wrapper
- Source code provided
- Support Documentation
  - Compiled HTML page distributed with the stack

Application Examples (Services)

HTTP	Hyper-Text Transfer Protocol (Server) Serves your web pages and processes web form input
SMTP	Simple Mail Transfer Protocol (Client) Sends e-mail messages
Telnet	Telnet (Server) Command-line interface
SNMP	Simple Network Management Protocol (Server) Enterprise monitoring and control (AN870)
SNTP	Simple Network Time Protocol (Client) Determines the absolute time
DHCP	Dynamic Host Configuration Protocol (Both) Automatic IP and network configuration
DNS	Domain Name Service (Client) Global host name resolution
IPv6	IPv6

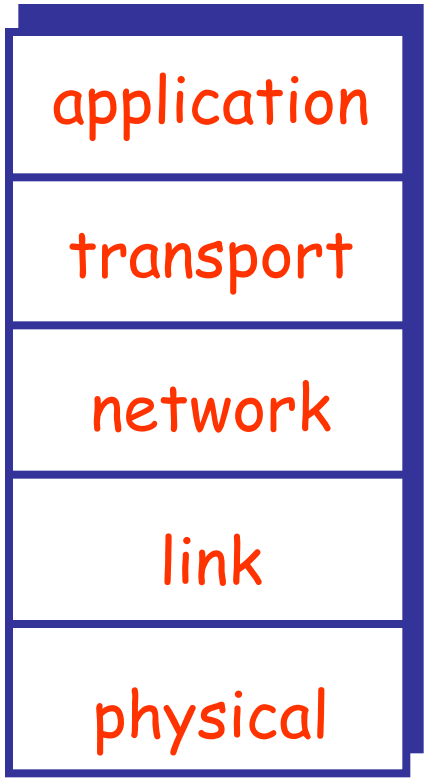
What's Included?



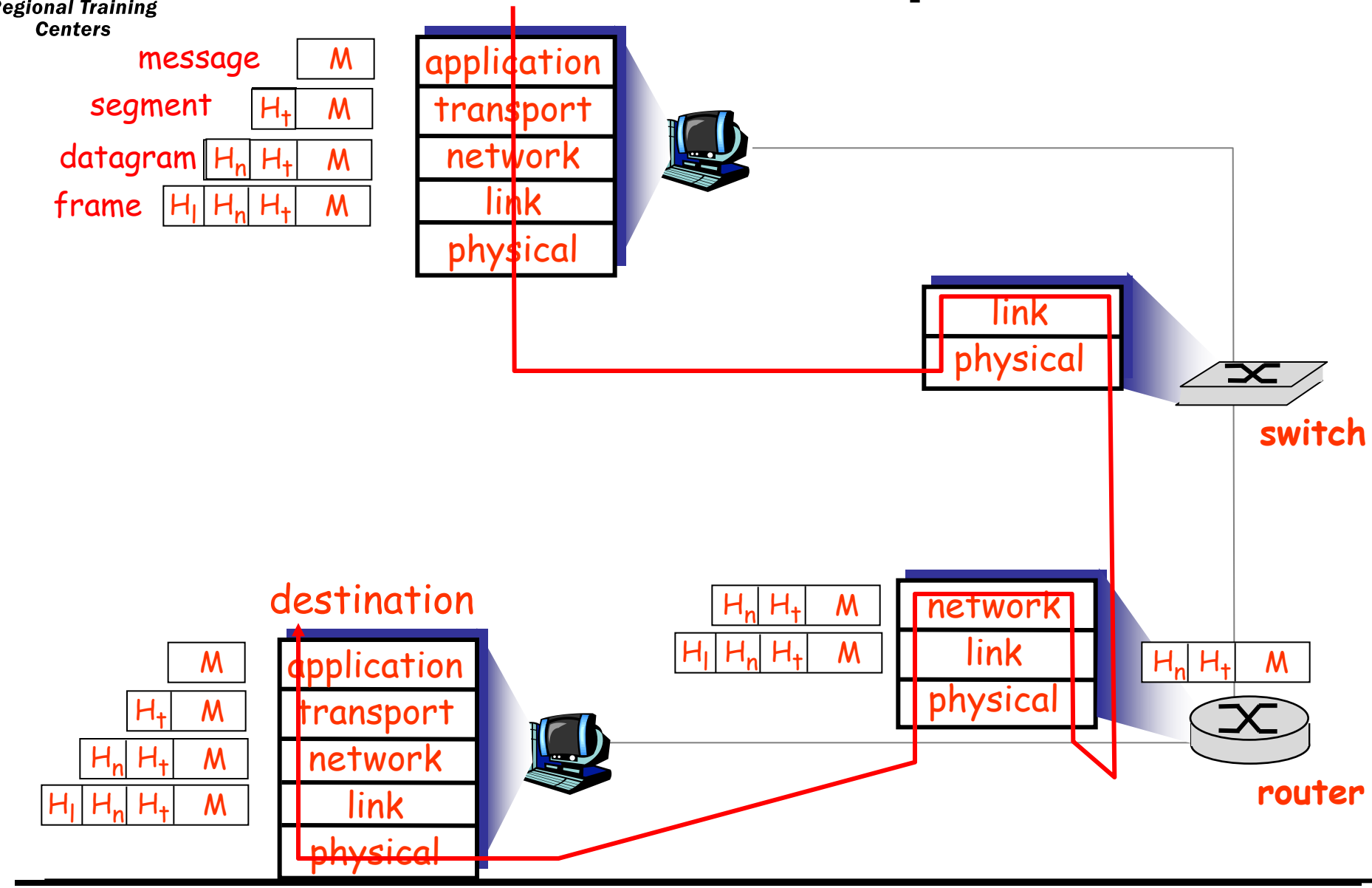
MICROCHIP
Regional Training Centers

Internet (TCP/IP) Protocol Stack

- **Application:** supporting network applications
  - FTP, SMTP, HTTP, Custom
- **Transport:** process-process data transfer
  - TCP, UDP
- **Network:** routing of datagrams from source to destination
  - IP, routing protocols
- **Link:** data transfer between neighboring network elements
  - PPP, Ethernet
- **Physical:** bits “on the wire”



Encapsulation



MICROCHIP
Regional Training Centers

About Addresses

- **Telephone System:**
  - Must supply a number to the system before you can talk to someone
- **Internet:**
  - End-end communication requires
    - MAC Address
    - IP Address
    - Port Number

MICROCHIP
Regional Training Centers

MAC Addresses

- MAC Addresses
  - Associated with hardware on the local network
    - Only usable within the local network
    - Not globally routable
  - Specific to IEEE 802.x networks
    - Ethernet, ZigBee®, Wi-Fi®, etc.
  - Six/Eight bytes: e.g. 00:04:A3:00:12:34
    - IEEE “EUI-48/64” format
    - First 3 bytes are Vendor OUI code
    - Last 3/5 bytes are serialized in production
  - Sold in blocks by the IEEE
    - Globally unique
    - US$550 / 4096 pcs or US$1650 / 16M pcs
  - Preprogrammed into MRF24WB0MA

MICROCHIP
Regional Training Centers

IP Addresses

- Internet Protocol (IP) Addresses
  - Network Address: Globally Routable
  - Tied to the part of the network that the host is connected to
  - Assigned by software (DHCP or manual)
    - Initial/Static address can be defined in TCPConfig.h
  - Four bytes as dotted-quad: 192.168.1.100
    - New version (IPv6) will have 128 bits
  - Administered by regional authorities
    - Typically leased by your ISP
  - Some allocated for private networks
    - 192.168.*, 10.*, 169.254.*, and 172.16.*
    - Behind firewall and/or router - Not globally routable

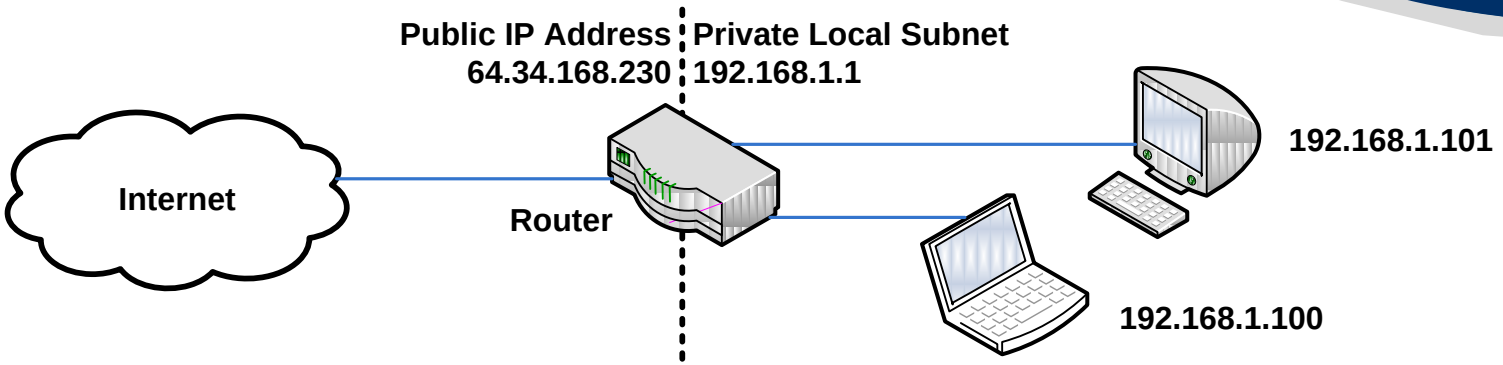
MICROCHIP
Regional Training Centers

Port Number

- Identifier of the target host application
  - 16-bit unsigned integer (1 to 65535, 0 is reserved)
- Some applications have standardized port #s
  - HTTP: 80
- Interpreted relative to an IP address
  - E.g. To send HTTP message to gaia.cs.umass.edu web server:
    - IP address: 128.119.245.12
    - Port number: 80
- Mail Analogy:
  - IP Address is the street address of the building
  - Port number is the room

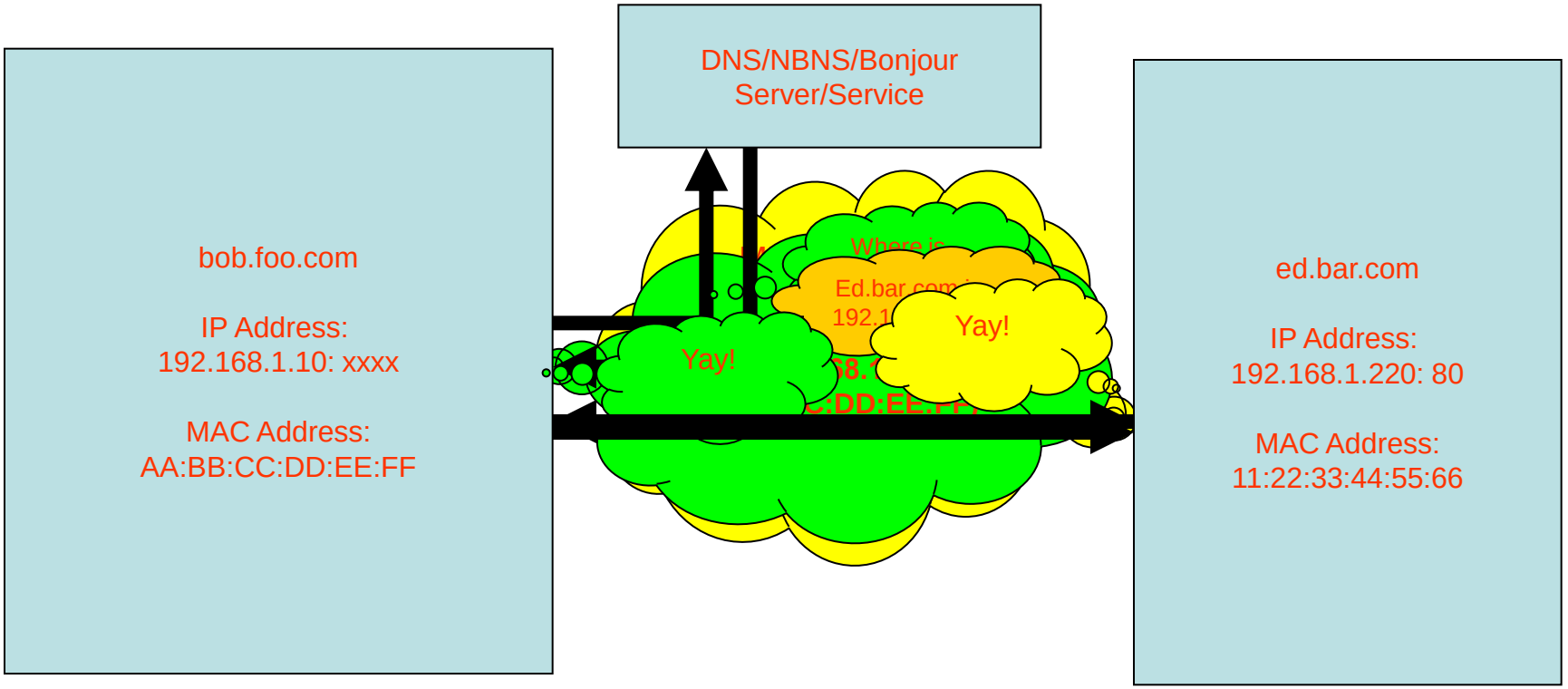
Private Network IP Addressing

- 32-bit public address space insufficient
- NAT (Network Address Translation) provides one-to-many routing
  - Share one global IP among many machines
  - Router/gateway makes all outgoing connections
  - Can port-forward incoming connections
  - Also provides security



Name Resolution

Bob wants to send a Packet to Ed



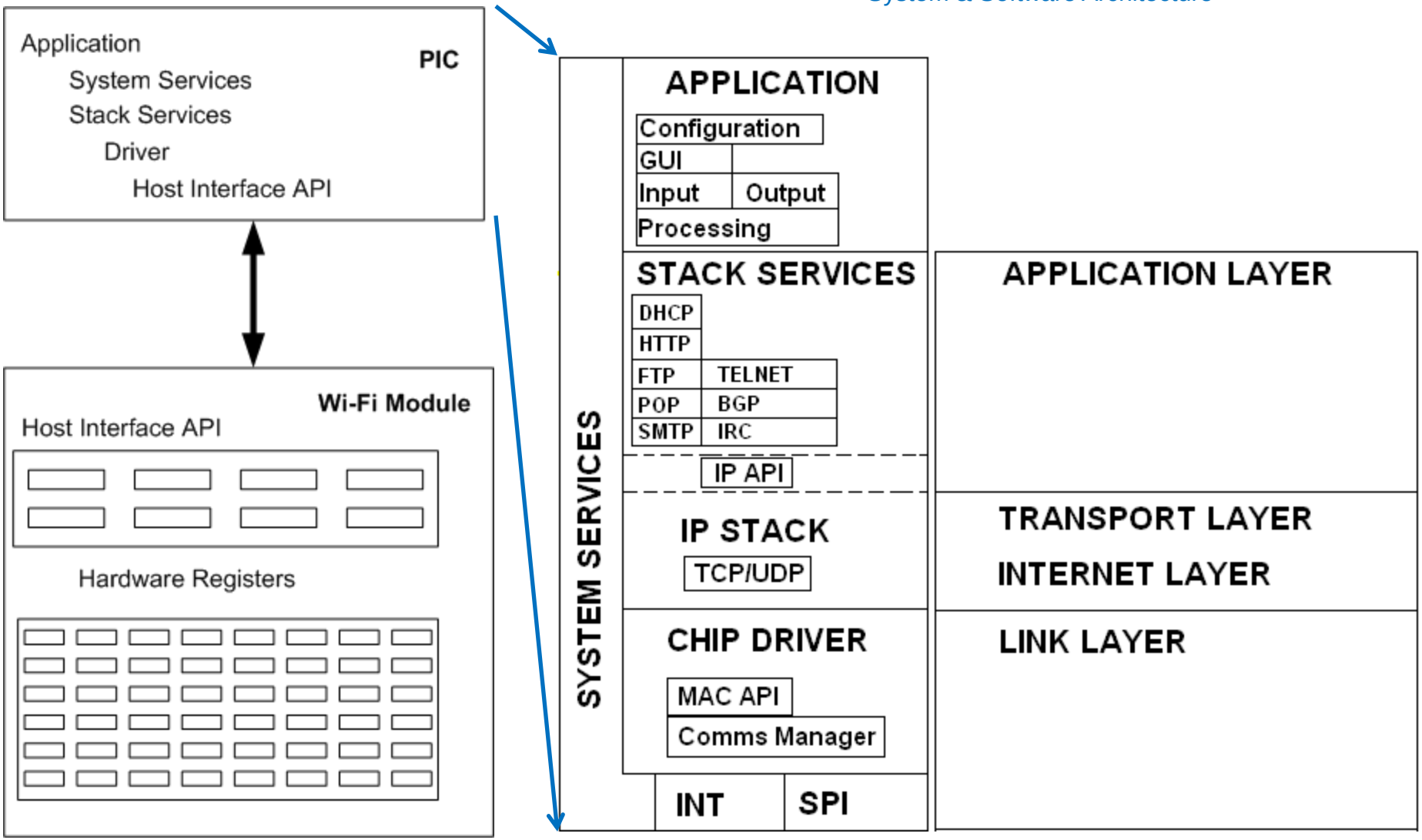
Bob sends an ARP looking for the hardware address associated with Ed's IP address

Bob sends a request to the DNS server asking for the IP address of Ed.

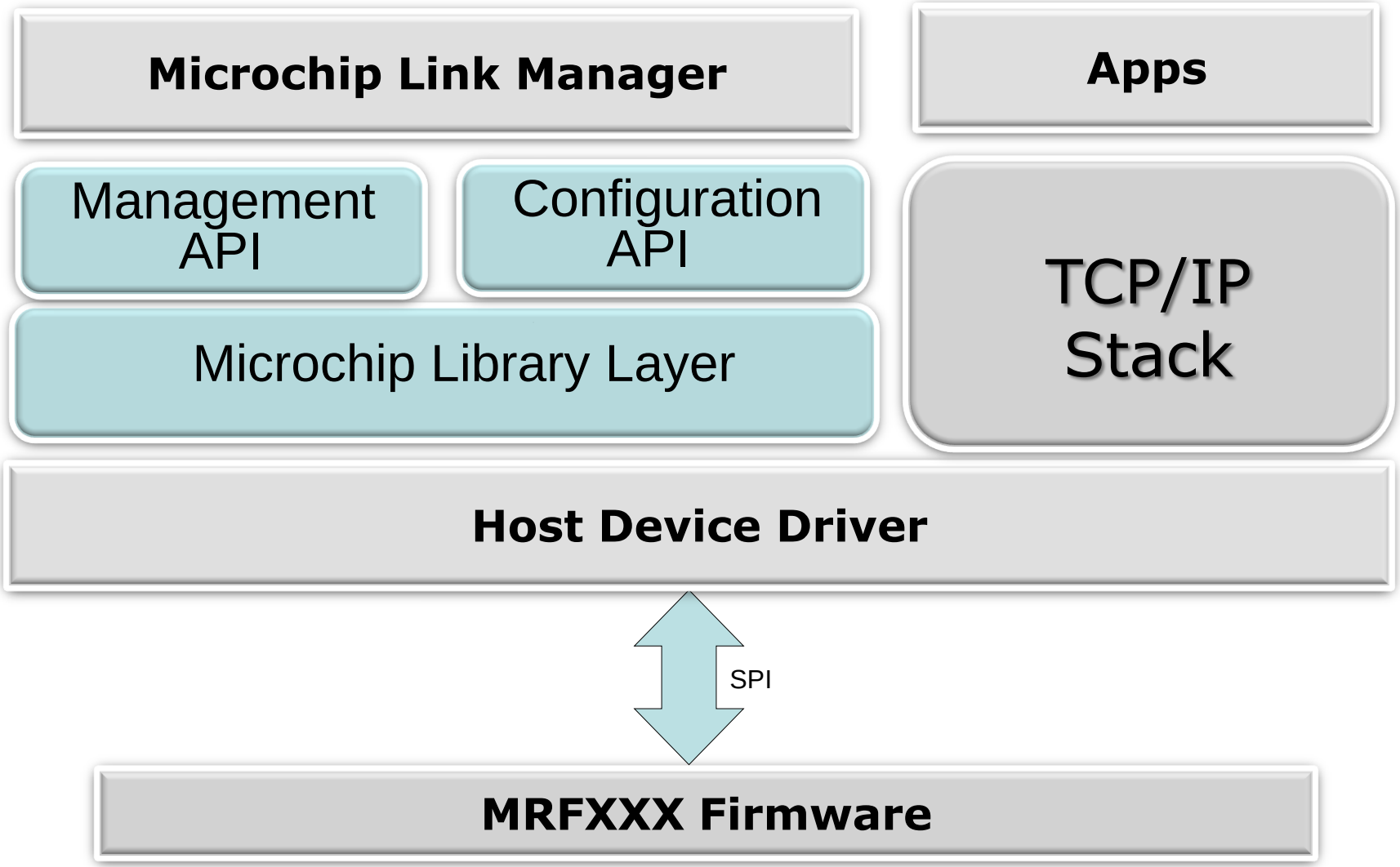
Now both Bob and Ed know where to address packets of their conversation

Microchip 802.11 Solution

System & Software Architecture



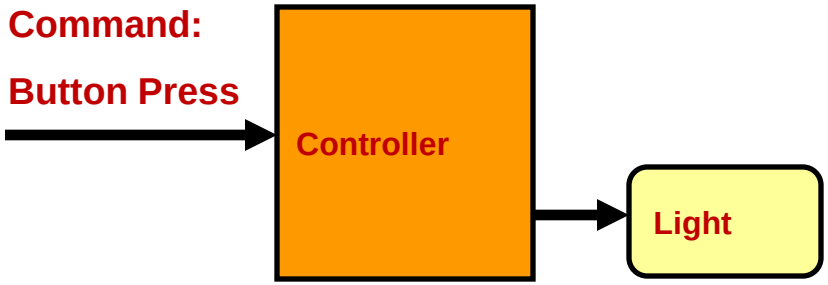
System Block Diagram



FSM(Finite-state machine)
State Machine Basics

- **Definition:**
  - **A model of a system consisting of**
    - **States**
    - **Transitions between states**
    - **Actions**

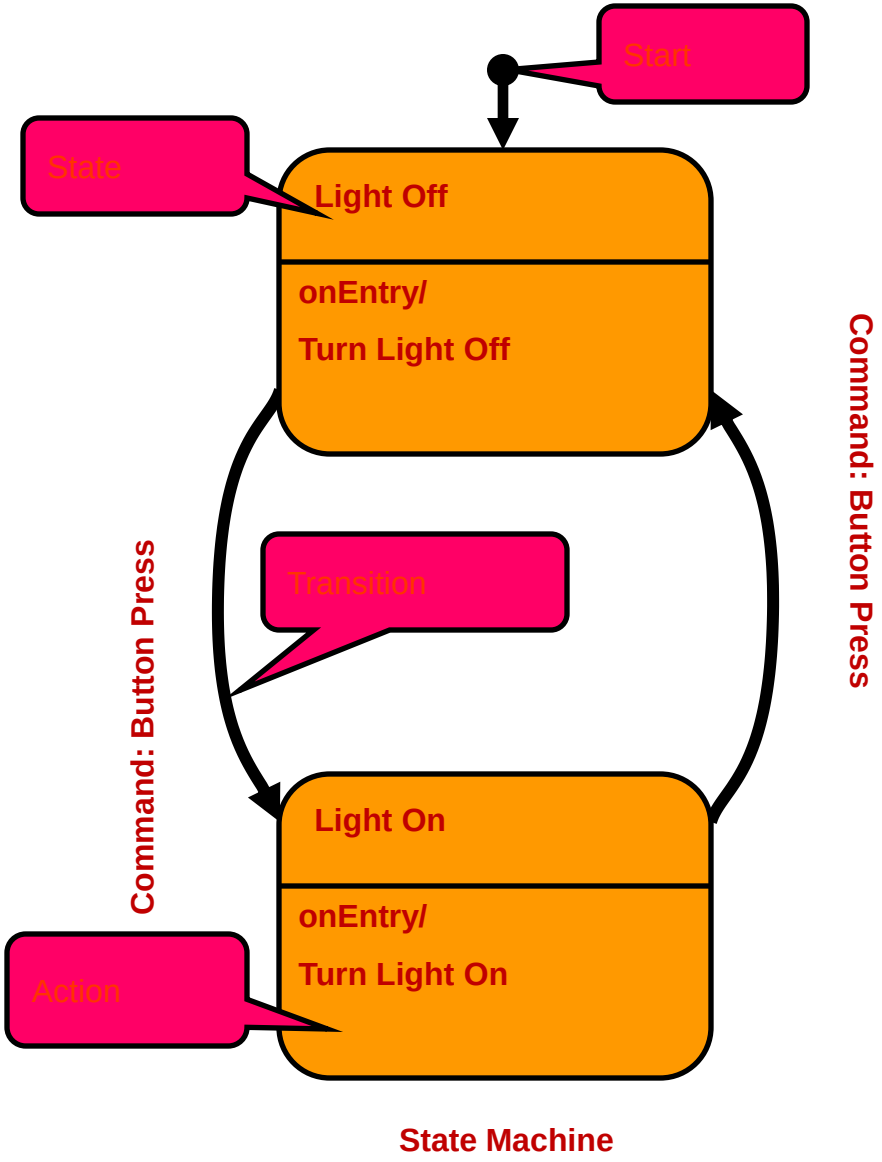
Example



Model of a push button flashlight

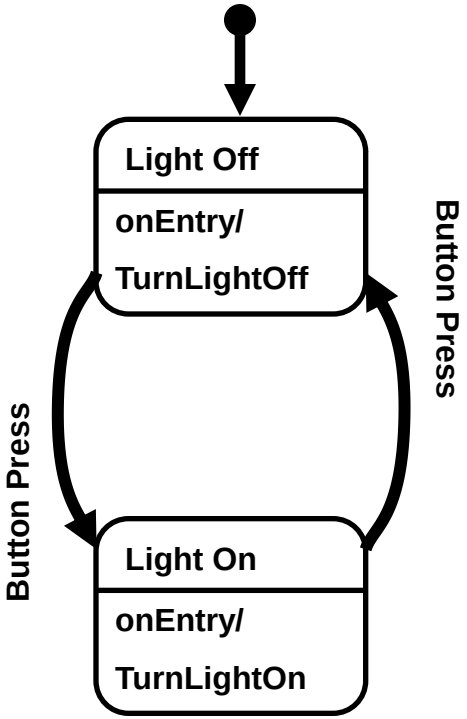
State / Input	Light Off	Light On
Button Press	Light On	Light Off

Transition Table



State Machine

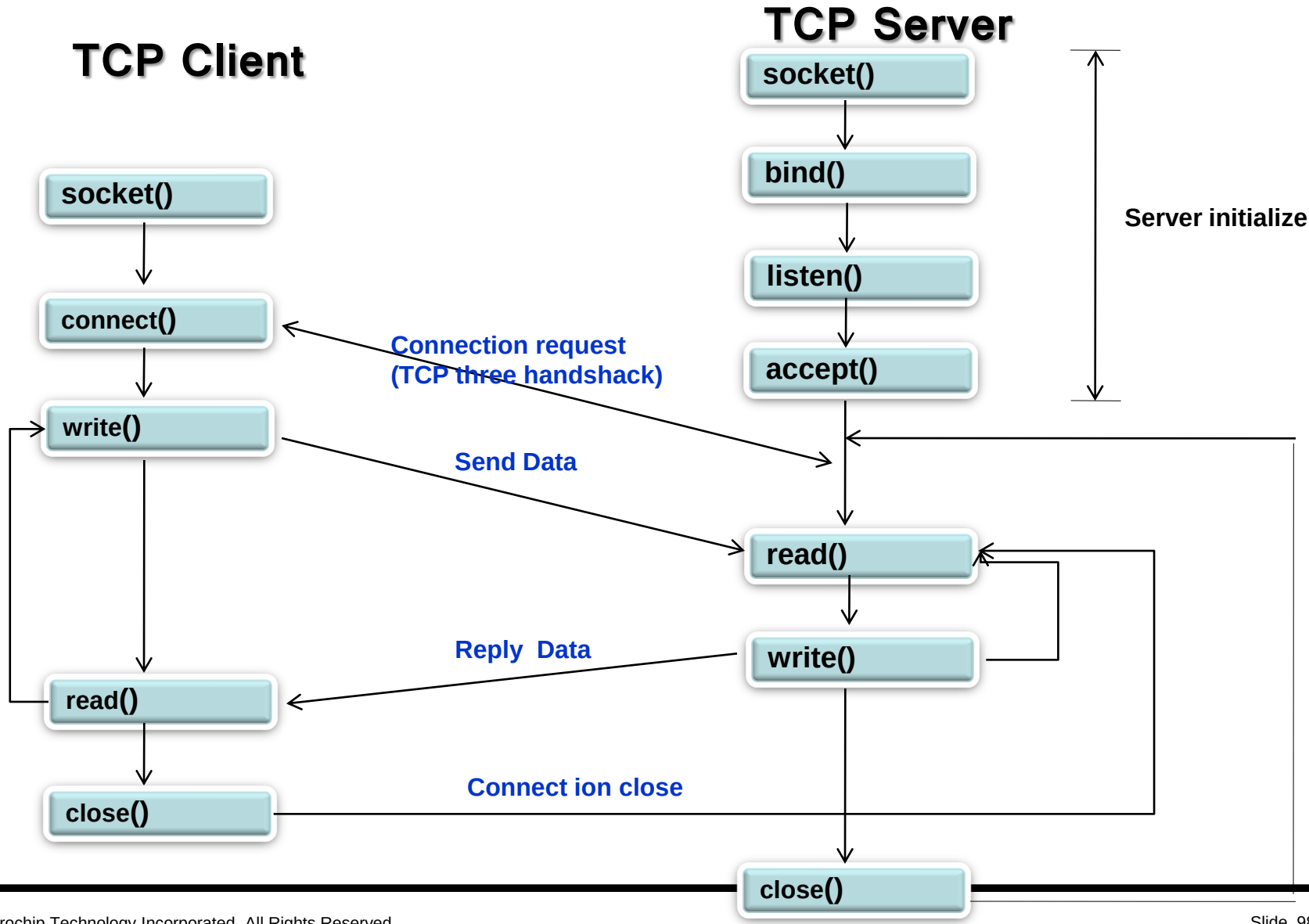
Write the Code : Moore



Moore State Machine

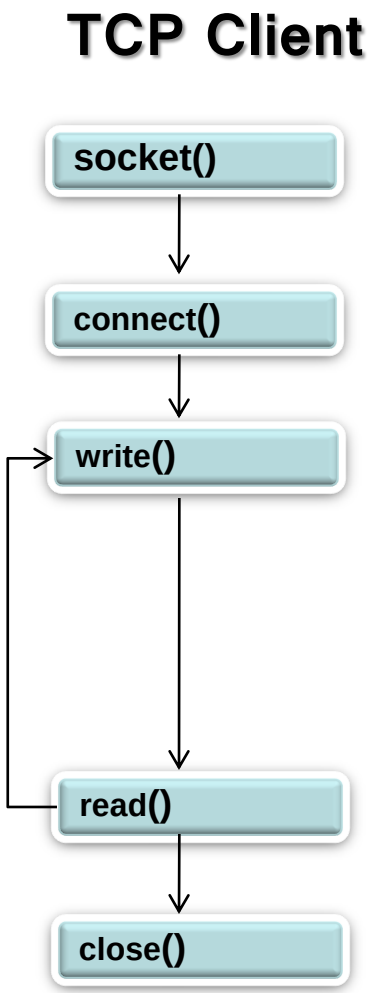
```
hw_init();
state=LightOff_Entry;
while(1) // Run the state machine forever
{
 switch(state)
 {
  case LightOff_Entry:
   TurnLightOff();
   state=LightOff;
  case LightOff:
   if(ButtonPress()) state = LightOn_Entry;
   break;

  case LightOn_Entry:
   TurnLightOn();
   state=LightOn;
  case LightOn:
   if(ButtonPress()) state = LightOff_Entry;
   break;
 }
}
```

TCP Client-Server

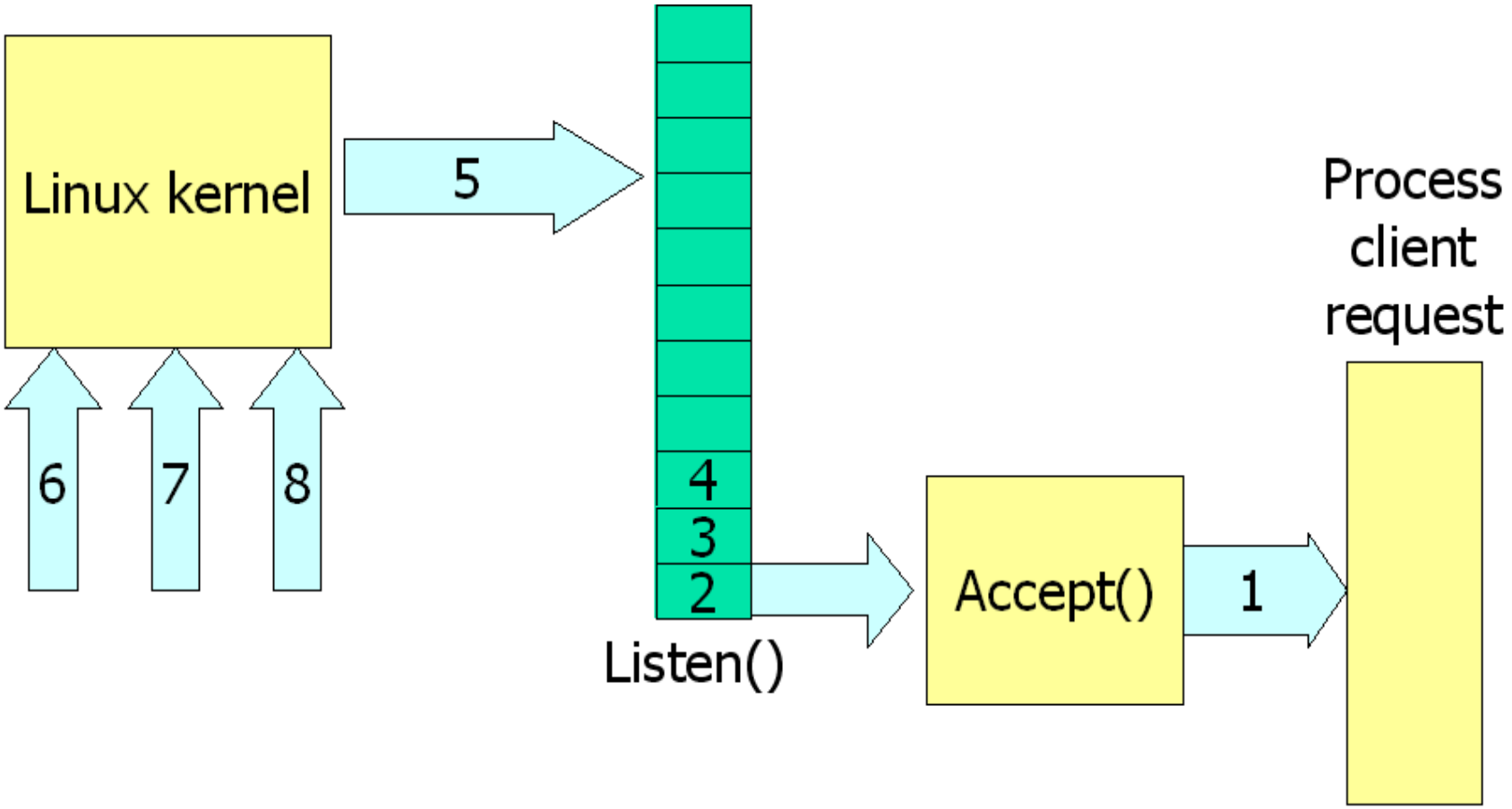


Linux Client Source Code

```
int main(int argc,char **argv)
{
    struct sockaddr_in addr_svr;
    int sockfd;
    char buffer[1024];
         // create server address
    memset(&addr_svr, 0, sizeof(addr_svr));
    addr_svr.sin_family= AF_INET;
    addr_svr.sin_port= htons(1234);
    addr_svr.sin_addr.s_addr = inet_addr("xxx.xxx.xxx.1");
    // create client socket
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    // connect
    connect(sockfd, (struct sockaddr *)&addr_svr, sizeof(addr_svr));
    // write to and read from server
    write(sockfd, buffer, strlen(buffer)+1);
    for(int len=0; ; ){
          len +=read(sockfd, buffer+len, 1024);
          if (len == 0)       break;
    }
    close(sockfd);
    return 0;
}
```



Linux TCP/IP



MICROCHIP
Regional Training Centers

Microchip-- Main Code

```
int main(void)
{
    // Initialize application specific hardware
    InitializeBoard();
    // Initialize Stack and application related NV variables into AppConfig.
    InitAppConfig();
    StackInit();
    ZGConsoleInit( p_cmdStringPtrList, kZGNumCmdsInList);
    while(1)
    {
        // This task performs normal stack task including checking
        // for incoming packet, type of packet and calling
        // appropriate stack entity to process it.
        StackTask();

        // This tasks invokes each of the core stack application tasks
        StackApplications();
        ZGConsoleProcess();
        PingAppCall();
        BarCodeReaderDemo();
        ProcessIO();
    }
}
```

LOOP

Microchip—Client Source Code(1)

```
#define BARCodePORTNUM 1234
BYTE    sendBarCodeRequest[20] = "\0";
void BarCodeReaderDemo(void)
    {
    #if defined(STACK_USE_DNS)
          static SOCKET bsdClientSocket = INVALID_SOCKET;
          static struct sockaddr_in addr;
          char recvBuffer[9];
          int i;
          int addrlen;
          IP_ADDR RemoteIP;
          static enum _BARCodeServerState
          {
                    BARDCode_START,
                    BARDCode_CONNECT,
                    BARDCode_SEND,
                    BARDCode_OPERATION,
                    BARDCode_CLOSE,
                    BARDCode_DONE,
          } BARCodeClientState = BARDCode_DONE;

          switch(BARCodeClientState)
          {
```

Microchip—Client Source Code(2)

```
case BARDCode_START:
if (ZGConsoleIsConsoleMsgReceived() == kZGBoolTrue)
{
   if (isIPAddress(argv[1], address))
   {
      RemoteIP.v[0] = address[0]; RemoteIP.v[1] = address[1];
      RemoteIP.v[2] = address[2]; RemoteIP.v[3] = address[3];
      addr.sin_addr.S_un.S_addr=RemoteIP.Val;
      // Create a socket for this client to connect with
      if((bsdClientSocket = socket(AF_INET, SOCK_STREAM, IPPROTO_TCP))
                                          == INVALID_SOCKET )
           return;

        memset(sendBarCodeRequest,0,strlen((char*) sendBarCodeRequest));
        strcpy((char*) sendBarCodeRequest,(char*) argv[2]);
        BARCodeClientState = BARDCode_CONNECT;
    }
    else
    {
        ZG_PUTSUART( (char *) "Error sendbarcode command\n\r");
        ZGConsoleReleaseConsoleMsg();
    }
 }

 break;
```

Microchip-- Main Code

```
int main(void)
{
    // Initialize application specific hardware
    InitializeBoard();
    // Initialize Stack and application related NV variables into AppConfig.
    InitAppConfig();
    StackInit();
    ZGConsoleInit( p_cmdStringPtrList, kZGNumCmdsInList);
    while(1)
    {
        // This task performs normal stack task including checking
        // for incoming packet, type of packet and calling
        // appropriate stack entity to process it.
        StackTask();

        // This tasks invokes each of the core stack application tasks
        StackApplications();
        ZGConsoleProcess();
        PingAppCall();
        BarCodeReaderDemo();
        ProcessIO();
    }
}
```

LOOP

Microchip—Client Source Code(3)

```
case BARDCode_CONNECT:
 // addr.sin_addr.S_un.S_addr destination IP address was set earlier in DNS step
 addr.sin_port = BARCodePORTNUM;  //PORTNUM;
 addrlen = sizeof(struct sockaddr);
 if(connect( bsdClientSocket, (struct sockaddr*)&addr, addrlen) < 0)
       return;

 BARCodeClientState = BARDCode_SEND;
 break;

case BARDCode_SEND:
  //send TCP data
  send(bsdClientSocket, (const char*)sendBarCodeRequest, strlen((char*)
                                                    sendBarCodeRequest), 0);
  BARCodeClientState = BARDCode_OPERATION;
  break;
```

MICROCHIP
Regional Training Centers

Microchip-- Main Code

```
int main(void)
{
    // Initialize application specific hardware
    InitializeBoard();
    // Initialize Stack and application related NV variables into AppConfig.
    InitAppConfig();
    StackInit();
    ZGConsoleInit( p_cmdStringPtrList, kZGNumCmdsInList);
    while(1)
    {
        // This task performs normal stack task including checking
        // for incoming packet, type of packet and calling
        // appropriate stack entity to process it.
        StackTask();

        // This tasks invokes each of the core stack application tasks
        StackApplications();
        ZGConsoleProcess();
        PingAppCall();
        BarCodeReaderDemo();
        ProcessIO();
    }
}
```

LOOP

Microchip—Client Source Code(4)

```
    case BARDCode_OPERATION:
        if(recv(bsdClientSocket, recvBuffer, 0, 2) < 0) //get the connection status
           BARCodeClientState = BARDCode_CLOSE;
        while(1)
        {
          i = recv(bsdClientSocket, recvBuffer, sizeof(recvBuffer)-1, 0);
          //get the data from the recv queue
          if(i <= 0)
          {
            //putsUART((char*) "receive null data\r\n");
            break;
          }
          putsUART((char*)recvBuffer);
          BARCodeClientState = BARDCode_CLOSE;
          break;
    case BARDCode_CLOSE:
          closesocket(bsdClientSocket);
          BARCodeClientState = BARDCode_DONE;
          break;
    case BARDCode_DONE:
          ZGConsoleReleaseConsoleMsg();
          BARCodeClientState = BARDCode_START;
          break;
    default:
          return;
}
```

Hand on
TCPIP WiFi Console App

- **Lab 1: Uart command→ connect wifi**

  step1: copy TCPIP Demo App→
  TCPIP Demo App_UartCommand

  step2: add uart command

  step3: disable auto connect wifi

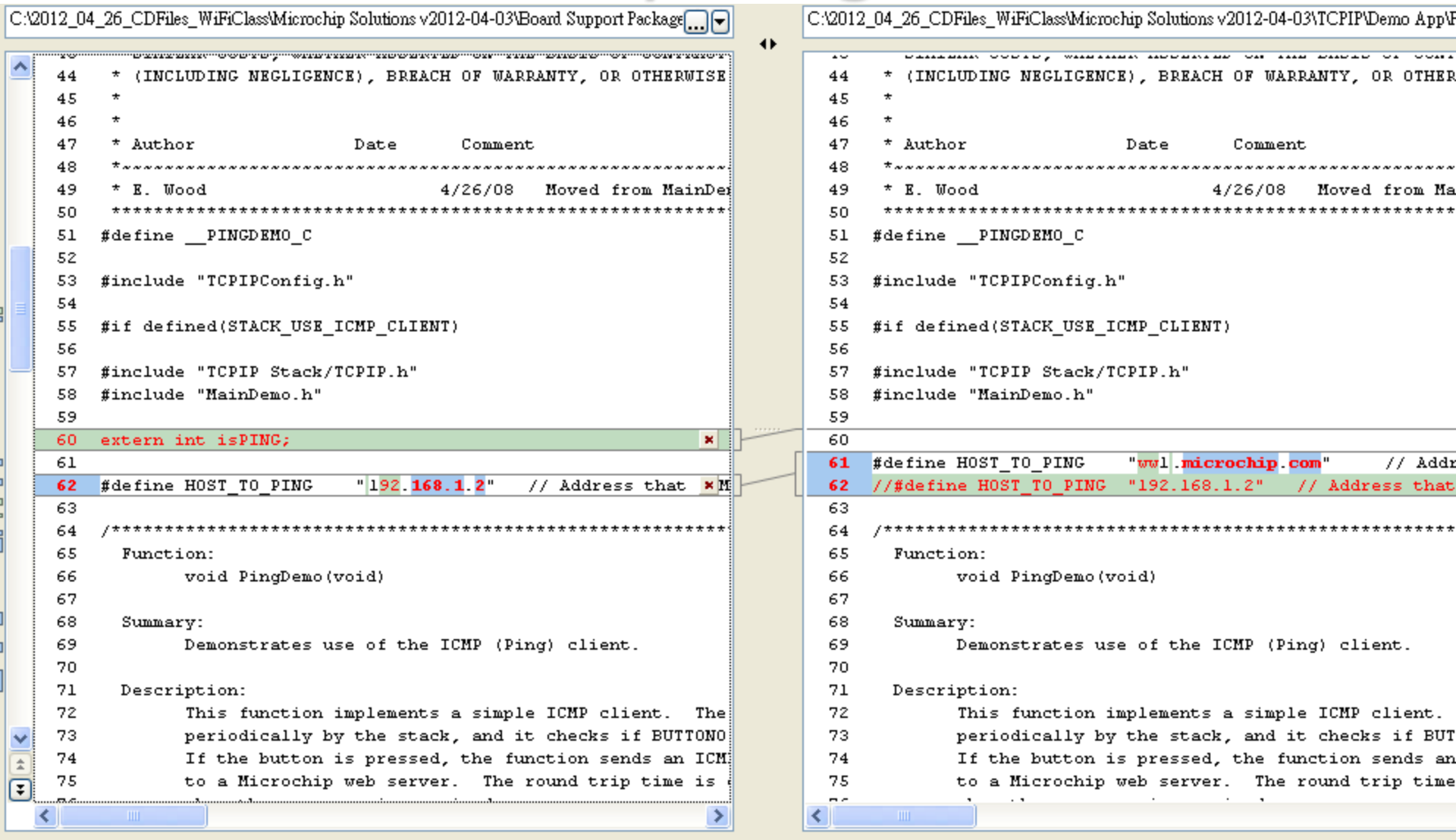
  step4: add connect wifi command

- **Lab 1: Ping command**

  step5: modify PingDemo.c

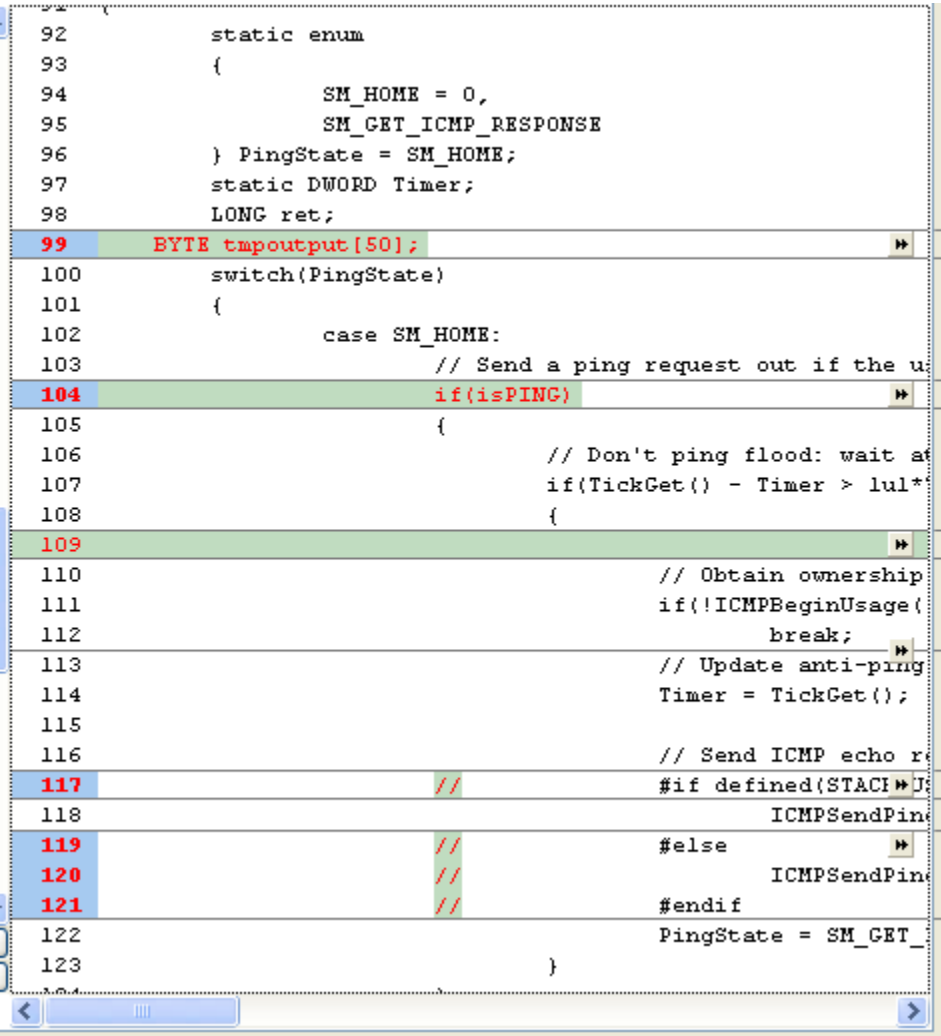
  step6: #define STACK_USE_ICMP_CLIENT

Modify PingDemo.c

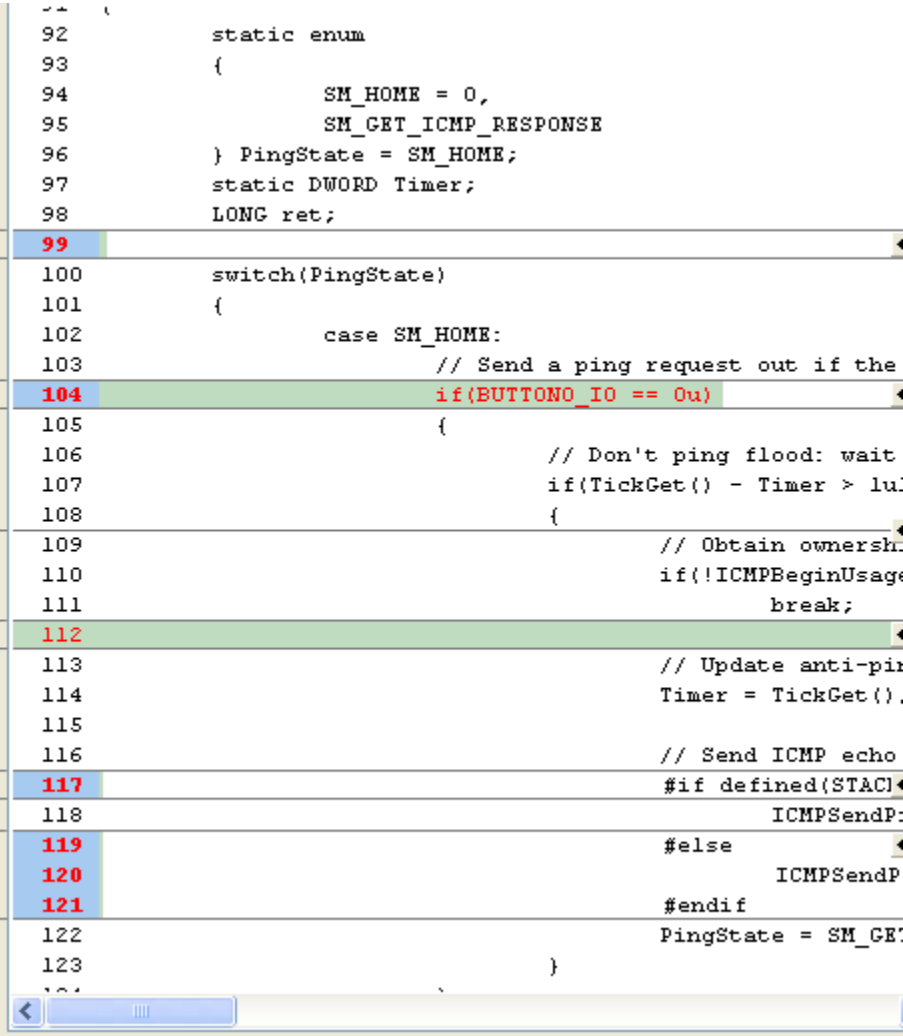


MICROCHIP
Regional Training Centers

Modify PingDemo.c



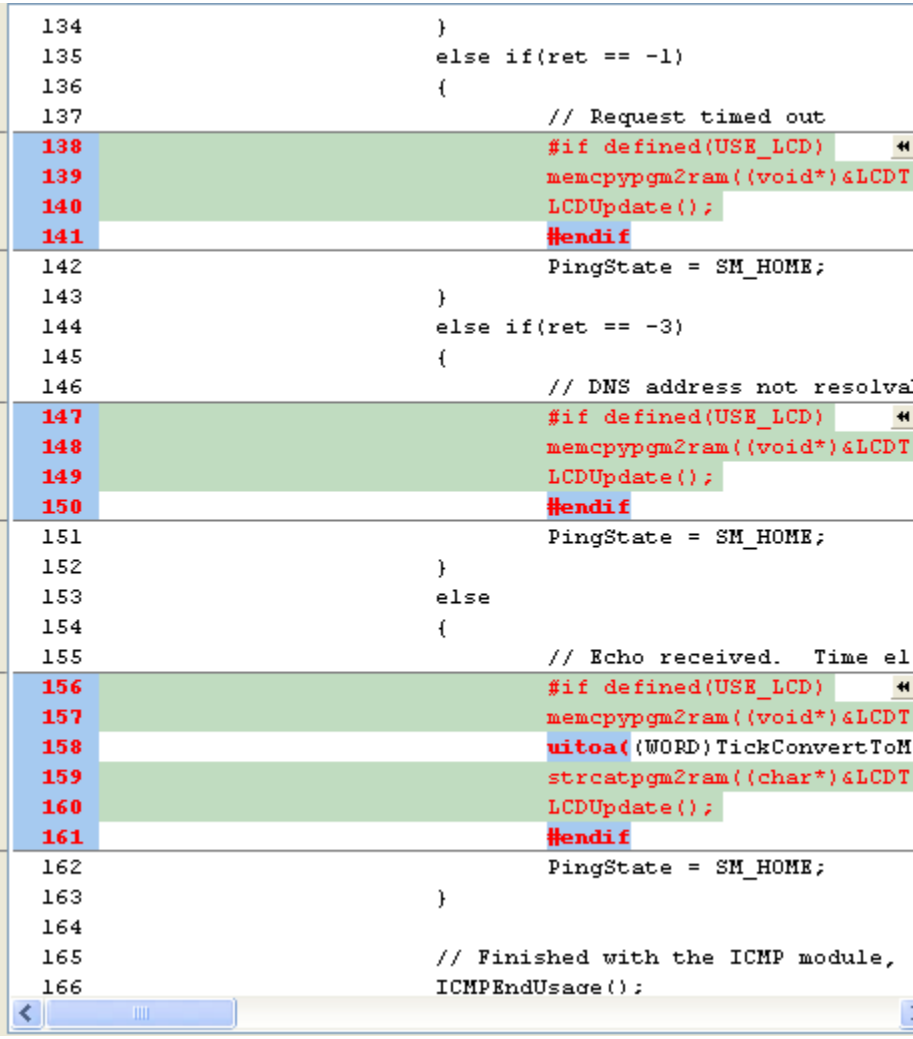
```
92          static enum
93          {
94                  SM_HOME = 0,
95                  SM_GET_ICMP_RESPONSE
96          } PingState = SM_HOME;
97          static DWORD Timer;
98          LONG ret;
99      BYTE tmpoutput[50];
100         switch(PingState)
101         {
102                 case SM_HOME:
103                         // Send a ping request out if the u
104                         if(isPING)
105                         {
106                                 // Don't ping flood: wait a
107                                 if(TickGet() - Timer > lul*
108                                 {
109
110                                         // Obtain ownership
111                                         if(!ICMPBeginUsage(
112                                                 break;
113                                         // Update anti-ping
114                                         Timer = TickGet();
115
116                                         // Send ICMP echo r
117                 //                      #if defined(STACI
118                                                 ICMPSendPin
119                 //                      #else
120                 //                              ICMPSendPin
121                 //                      #endif
122                                         PingState = SM_GET_
123                                 }
```



```
92          static enum
93          {
94                  SM_HOME = 0,
95                  SM_GET_ICMP_RESPONSE
96          } PingState = SM_HOME;
97          static DWORD Timer;
98          LONG ret;
99
100         switch(PingState)
101         {
102                 case SM_HOME:
103                         // Send a ping request out if the
104                         if(BUTTON0_IO == 0u)
105                         {
106                                 // Don't ping flood: wait
107                                 if(TickGet() - Timer > lu
108                                 {
109                                         // Obtain ownersh
110                                         if(!ICMPBeginUsage
111                                                 break;
112
113                                         // Update anti-pi
114                                         Timer = TickGet()
115
116                                         // Send ICMP echo
117                                         #if defined(STAC
118                                                 ICMPSendP
119                                         #else
120                                                 ICMPSendP
121                                         #endif
122                                         PingState = SM_GE
123                                 }
```

Modify PingDemo.c

```
131                     {
132                             // Do nothing: still waitin
133                             break;
134                     }
135                     else if(ret == -1)
136                     {
137                             // Request timed out
138                     putrsUART("\r\n Ping Timed out\r
139
140             isPING=0;
141                             PingState = SM_HOME;
142                     }
143                     else if(ret == -3)
144                     {
145                             // DNS address not resolvab
146                     putrsUART("\r\n Can't resolve IP
147             isPING=0;
148                             PingState = SM_HOME;
149                     }
150                     else
151                     {
152                             // Echo received.  Time ela
153             memset(tmpoutput,0,sizeof(tmpoutput));
154                                     sprintf( (char *) tmpo
155                              "\r\nReply:%dms\r\n", (WORD)TickC
156                                 putsUART(tmpoutput);
157             isPING=0;
158                             PingState = SM_HOME;
159                     }
160
161                     // Finished with the ICMP module, r
162                     ICMPEndUsage();
163                     break;
```



```
134                     }
135                     else if(ret == -1)
136                     {
137                             // Request timed out
138                             #if defined(USE_LCD)
139                             memcpypgm2ram((void*)&LCDT
140                             LCDUpdate();
141                             #endif
142                             PingState = SM_HOME;
143                     }
144                     else if(ret == -3)
145                     {
146                             // DNS address not resolva
147                             #if defined(USE_LCD)
148                             memcpypgm2ram((void*)&LCDT
149                             LCDUpdate();
150                             #endif
151                             PingState = SM_HOME;
152                     }
153                     else
154                     {
155                             // Echo received.  Time el
156                             #if defined(USE_LCD)
157                             memcpypgm2ram((void*)&LCDT
158                             uitoa((WORD)TickConvertToM
159                             strcatpgm2ram((char*)&LCDT
160                             LCDUpdate();
161                             #endif
162                             PingState = SM_HOME;
163                     }
164
165                     // Finished with the ICMP module,
166                     ICMPEndUsage();
```

Modify BerkerlyTCPClientDemo.c

```
47   * Author                Date            Comment
48   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
49   * Aseem Swalah          4/17/08         Original
50   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
51
52   #include "TCPIPConfig.h"
53
54   #if defined(STACK_USE_BERKELEY_API)
55
56   #include "TCPIP Stack/TCPIP.h"
57
58   /*
59   #define PORTNUM 80
60   static ROM BYTE ServerName[] =  "www.google.com";
61   // This is specific to this HTTP Client example
62   static BYTE sendRequest[] = "GET /search?as_q=Microchip&as_
63   */
64   IP_ADDR RemoteIP;
65   #define PORTNUM 443
66   // This is specific to this HTTP Client example
67   static BYTE sendRequest[] = "4710060006145";
68   extern int isGlobalSend;
69
70
71   /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
72    * Function:         void BerkeleyTCPClientDemo(void)
73    *
74    * PreCondition:     Stack is initialized()
75    *
76    * Input:            None
77    *
78    * Output:           None
79    *
```

```
47   * Author                Date            Comment
48   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
49   * Aseem Swalah          4/17/08         Original
50   *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
51
52   #include "TCPIPConfig.h"
53
54   #if defined(STACK_USE_BERKELEY_API)
55
56   #include "TCPIP Stack/TCPIP.h"
57
58
59   #define PORTNUM 80
60   static ROM BYTE ServerName[] =  "www.google.com";
61   // This is specific to this HTTP Client example
62   static BYTE sendRequest[] = "GET /search?as_q=Microchip&as
63
64   /*~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
65    * Function:         void BerkeleyTCPClientDemo(void)
66    *
67    * PreCondition:     Stack is initialized()
68    *
69    * Input:            None
70    *
71    * Output:           None
72    *
73    * Side Effects:     None
74    *
75    * Overview:         None
76    *
77    * Note:             None
78    *~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

MICROCHIP
Regional Training Centers

Modify BerkerlyTCPClientDemo.c

```
104     } BSDClientState = BSD_DONE;
105
106         if(isGlobalSend == 1)
107         {
108           BSDClientState = DNS_START_RESOLUTION;
109           isGlobalSend=0;
110         }
111
112
113     switch(BSDClientState)
114     {
115             case DNS_START_RESOLUTION:
116
117
118                         RemoteIP.v[0] = 192;
119             RemoteIP.v[1] = 168;
120             RemoteIP.v[2] = 1;
121             RemoteIP.v[3] = 2;
122                         addr.sin_addr.S_un.S_addr=RemoteIP
123             BSDClientState = BSD_START;
124                 break;
125
126
127         case BSD_START:
128             // Create a socket for this client to connect w
129             if((bsdClientSocket = socket(AF_INET, SOCK_STRE
130                 return;
131
132             BSDClientState = BSD_CONNECT;
133             break;
134
135         case BSD_CONNECT:
136             // addr.sin_addr.S_un.S_addr destination IP add
```

```
100     switch(BSDClientState)
101     {
102             case DNS_START_RESOLUTION:
103                 if(DNSBeginUsage())
104                 {
105                         DNSResolveROM(ServerName, DNS_TYPE
106                         BSDClientState = DNS_GET_RESULT;
107                     }
108                 break;
109
110             case DNS_GET_RESULT:
111                 if(!DNSIsResolved((IP_ADDR*)&addr.sin_addr
112                         break;
113
114                 if(!DNSEndUsage())
115                 {
116                                 #if defined(STACK_USE_UART
117                                 putrsUART((ROM char*)"Coul
118                                 #endif
119                         BSDClientState = BSD_DONE;
120                         break;
121                     }
122
123                 BSDClientState = BSD_START;
124                 // No break; here.
125
126         case BSD_START:
127             // Create a socket for this client to connect
128             if((bsdClientSocket = socket(AF_INET, SOCK_STR
129                 return;
130
131                         #if defined(STACK_USE_UART)
132                         putrsUART((ROM char*)"\r\n\r\nConn
```

MICROCHIP
Regional Training Centers

Modify BerkerlyTCPClientDemo.c

```
104     } BSDClientState = BSD_DONE;
105
106         if(isGlobalSend == 1)
107         {
108           BSDClientState = DNS_START_RESOLUTION;
109           isGlobalSend=0;
110         }
111
112
113     switch(BSDClientState)
114     {
115             case DNS_START_RESOLUTION:
116
117
118                         RemoteIP.v[0] = 192;
119             RemoteIP.v[1] = 168;
120             RemoteIP.v[2] = 1;
121             RemoteIP.v[3] = 2;
122                         addr.sin_addr.S_un.S_addr=RemoteIP
123             BSDClientState = BSD_START;
124                 break;
125
126
127         case BSD_START:
128             // Create a socket for this client to connect w
129             if((bsdClientSocket = socket(AF_INET, SOCK_STRE
130                 return;
131
132             BSDClientState = BSD_CONNECT;
133             break;
134
135         case BSD_CONNECT:
136             // addr.sin_addr.S_un.S_addr destination IP add
```

```
100     switch(BSDClientState)
101     {
102             case DNS_START_RESOLUTION:
103                 if(DNSBeginUsage())
104                 {
105                         DNSResolveROM(ServerName, DNS_TYPE
106                         BSDClientState = DNS_GET_RESULT;
107                     }
108                 break;
109
110             case DNS_GET_RESULT:
111                 if(!DNSIsResolved((IP_ADDR*)&addr.sin_addr
112                         break;
113
114                 if(!DNSEndUsage())
115                 {
116                                 #if defined(STACK_USE_UART
117                                 putrsUART((ROM char*)"Coul
118                                 #endif
119                         BSDClientState = BSD_DONE;
120                         break;
121                     }
122
123                 BSDClientState = BSD_START;
124                 // No break; here.
125
126         case BSD_START:
127             // Create a socket for this client to connect
128             if((bsdClientSocket = socket(AF_INET, SOCK_STR
129                 return;
130
131                         #if defined(STACK_USE_UART)
132                         putrsUART((ROM char*)"\r\n\r\nConn
```

Modify BerkerlyTCPClientDemo.c

```
155                         i = recv(bsdClientSocket, recvBuffe
156
157         if(i == 0)
158                                 break;
159
160         if(i< 0) //error condition
161         {
162             BSDClientState = BSD_CLOSE;
163                                 putsUART((char*)"Connect
164             break;
165         }
166
167         #if defined(STACK_USE_UART)
168         recvBuffer[i] = '\0';   // Null terminate data
169         putsUART((char*)recvBuffer);
170                         BSDClientState = BSD_CLOSE;
171         #endif
172
173         if(BSDClientState == BSD_OPERATION)
174             break;
175     }
176     break;
177
178 case BSD_CLOSE:
179     closesocket(bsdClientSocket);
180     BSDClientState = BSD_DONE;
181     // No break needed
182       break;
183 case BSD_DONE:
184    // isGlobalSend=0;
185      putsUART((char*) "\n");
186     break;
187
```

```
159     while(1)
160     {
161                         i = recv(bsdClientSocket, recvBuff
162
163         if(i == 0)
164                                 break;
165
166         if(i< 0) //error condition
167         {
168             BSDClientState = BSD_CLOSE;
169             break;
170         }
171
172         #if defined(STACK_USE_UART)
173         recvBuffer[i] = '\0';   // Null terminate data
174         putsUART((char*)recvBuffer);
175         #endif
176
177         if(BSDClientState == BSD_OPERATION)
178             break;
179     }
180     break;
181
182 case BSD_CLOSE:
183     closesocket(bsdClientSocket);
184     BSDClientState = BSD_DONE;
185     // No break needed
186
187 case BSD_DONE:
188     if(BUTTON2_IO == 0u)
189         BSDClientState = DNS_START_RESOLUTION;
190     break;
191
```

MICROCHIP

Regional Training Centers

Appendix

Regional Training Centers

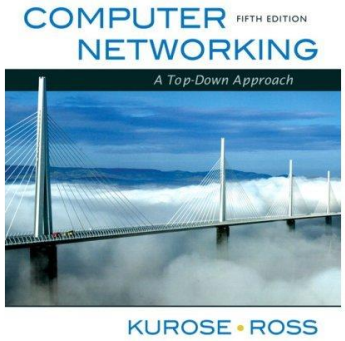
WiFi Remote Controller (SoftAP mode)

MICROCHIP

Regional Training Centers

Summary & References

References



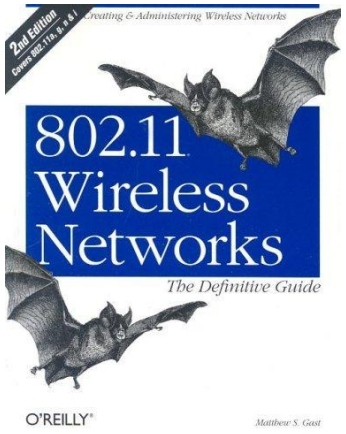
Computer Networking

A Top-Down Approach, 5th Edition

by James Kurose and Keith Ross

ISBN-10: 0136079679

ISBN-13: 978-0136079675



802.11 Wireless Networks

The Definitive Guide, 2nd Edition

by Matthew Gast

ISBN-10: 0596100523

ISBN-13: 978-0596100520

References

Wireshark Network Analysis

The Official Wireshark Certified Network Analyst Guide

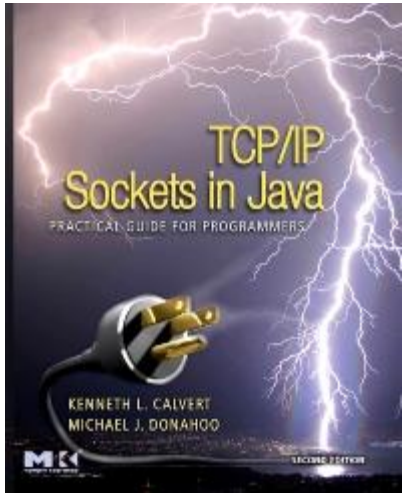
by Laura Chappell

ISBN-10: 1-893939-99-5

ISBN-13: 978-1-893939-8

www.wiresharkbook.com

www.wireshark.org



TCP/IP Sockets in Java

Practical Guide for Programmers

by Kenneth Calvert and Michael Donahoo

ISBN-10: 0123742552

ISBN-13: 978-0123742551

www.amazon.com

MICROCHIP
Regional Training Centers

Additional Resources

- Weblinks
  - www.microchip.com/wifi
  - www.microchip.com/tcpip
- Forums
  - forum.microchip.com
- Training (HTTP, etc)
  - Microchip RTC courses
- App Notes
  - AN833 Original Microchip TCP/IP Stack
  - AN1120 Ethernet Theory of Operation

MICROCHIP

Regional Training Centers

Thank You