MICROCHIP

Regional Training Centers

Designing a WiFi -Based Monitor and Control Device

COM-WiFi

MICROCHIP
Regional Training Centers

Agenda

1. Embedded Wi-Fi®
   - **What's that?**
   - **Microchip 802.11 Solution**
2. Overview of 802.11 Networks
   - **IEEE 802.11 Protocol**
   - **802.11 Media Access and Frame Control**
   - **802.11 Security**
3. Wireless Network Deployment & Wireless AP Feature
4. Creating Wi-Fi Enabled Applications
   - **TCPIP WiFi Demo App**
5. Software Architecture
6. WiFi Programming

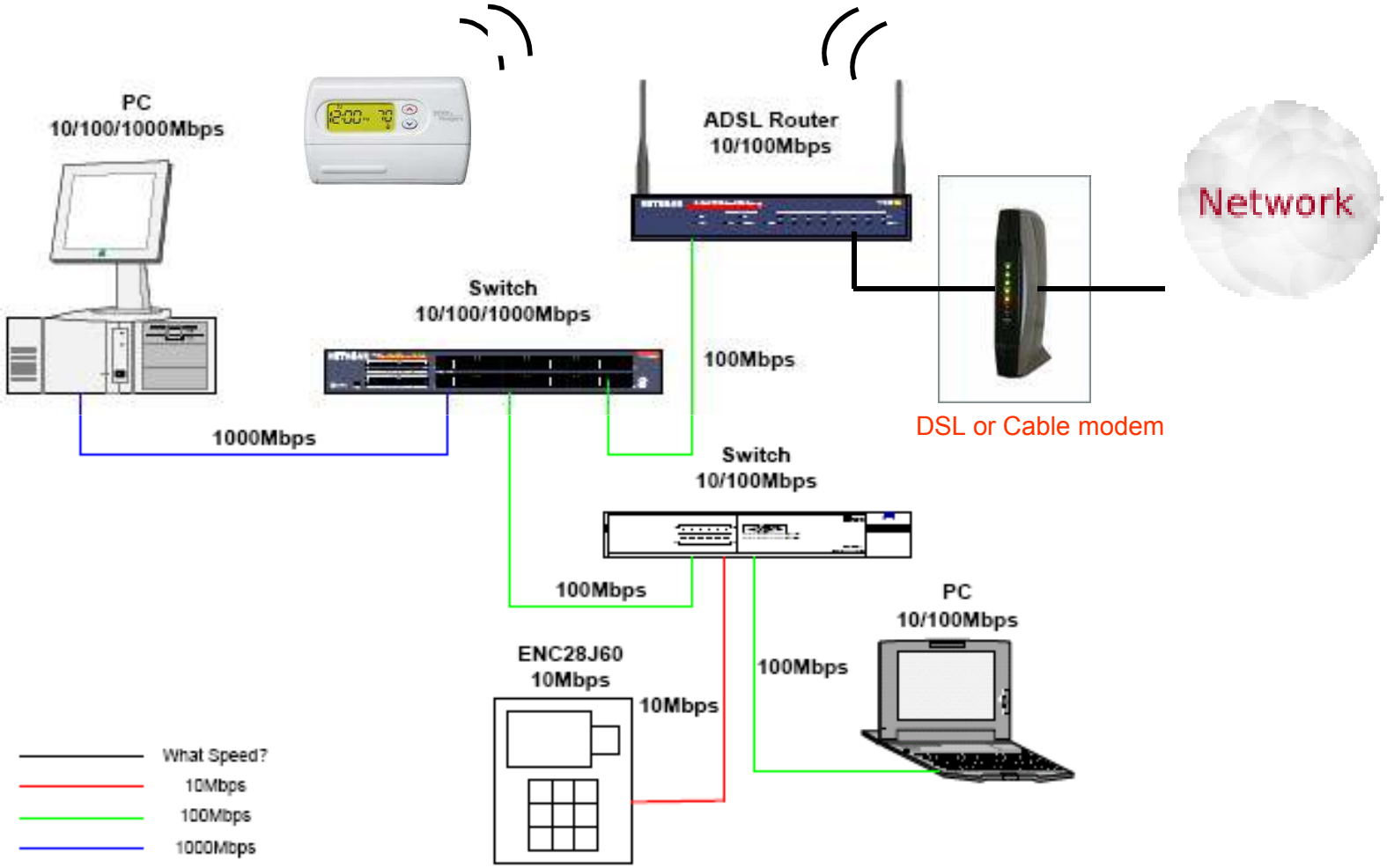
MICROCHIP

Regional Training Centers

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

MICROCHIP
Regional Training Centers

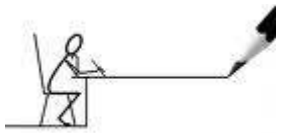
Embedded Wi-Fi®

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

MICROCHIP
Regional Training Centers

Development Desirables

- Seamless support of standards
- Don’t require networking expertise
- Usable examples to start
- Higher level application modules
- Small investment to investigate



Microchip 802.11 Solution

- **Infrastructure (BSS)**

- **Ad Hoc (IBSS)**

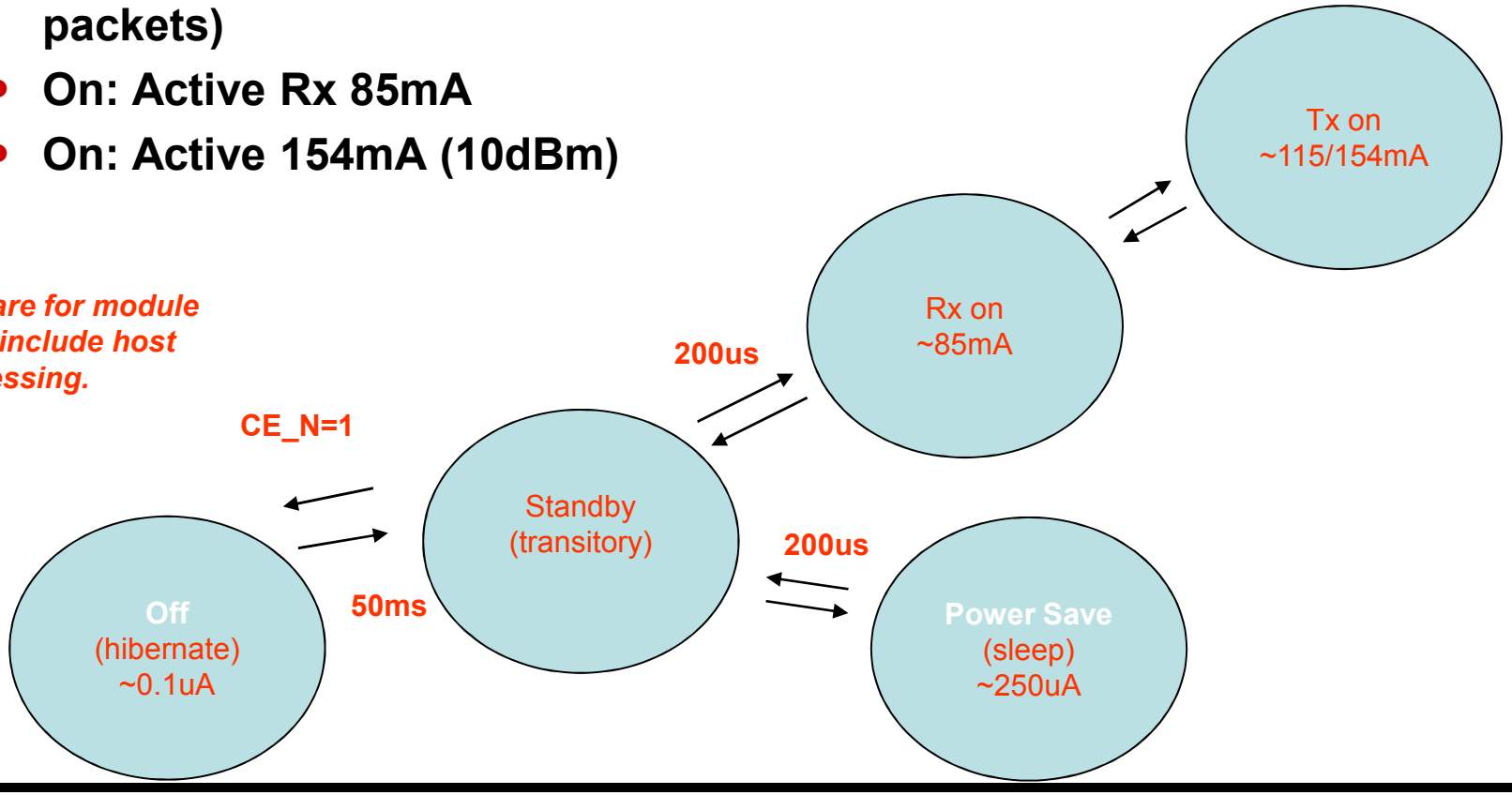
Microchip 802.11 Solution

- Wi-Fi® for low resource embedded applications
- Highly integrated, certified module solution
  - Includes MAC, baseband, RF and PA
  - Easy integration into 8/16/32-bit systems
  - Standards-based, 1 & 2Mb/s, 802.11b
  - Full support of ad-hoc and AP modes
  - FCC, IC certified and ETSI tested
  - Built-in antenna (A version)
  - Supports external antennas (B version)
  - Built-in secure WEP, WPA-PSK, WPA2-PSK
  - Support built-in on Microchip TCPIP Stack

Understanding Power Modes

- 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.



MICROCHIP
Regional Training Centers

Product Fit

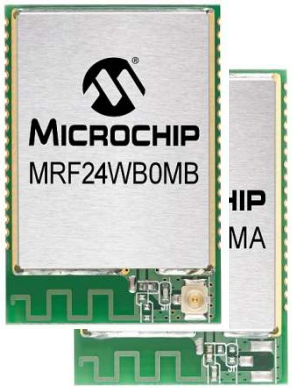
- Types of products that are ideal
  - Small data quantity to transfer
  - Non-continuous transfer
  - Data is not high bandwidth streaming



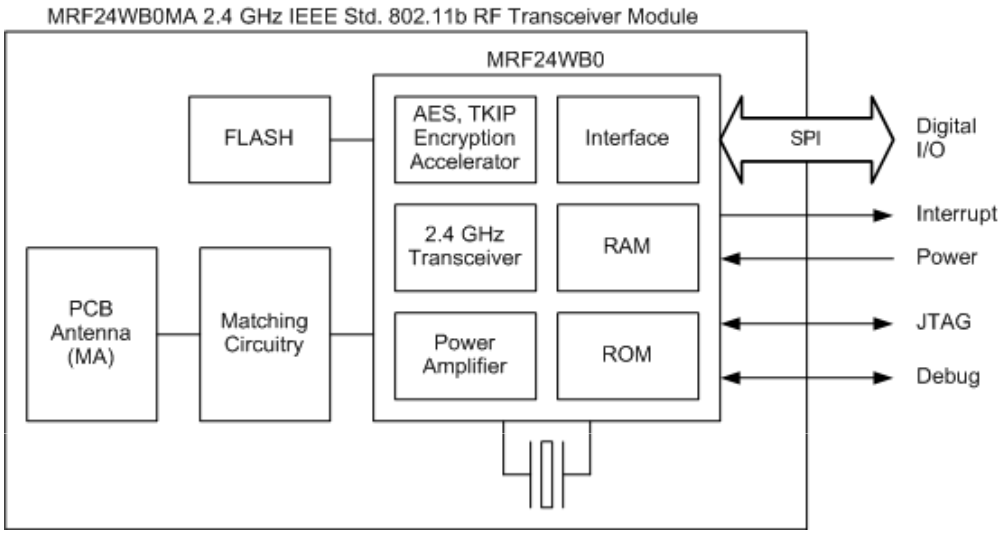




Microchip Wi-Fi® Modules



MRF24WB0MA/B

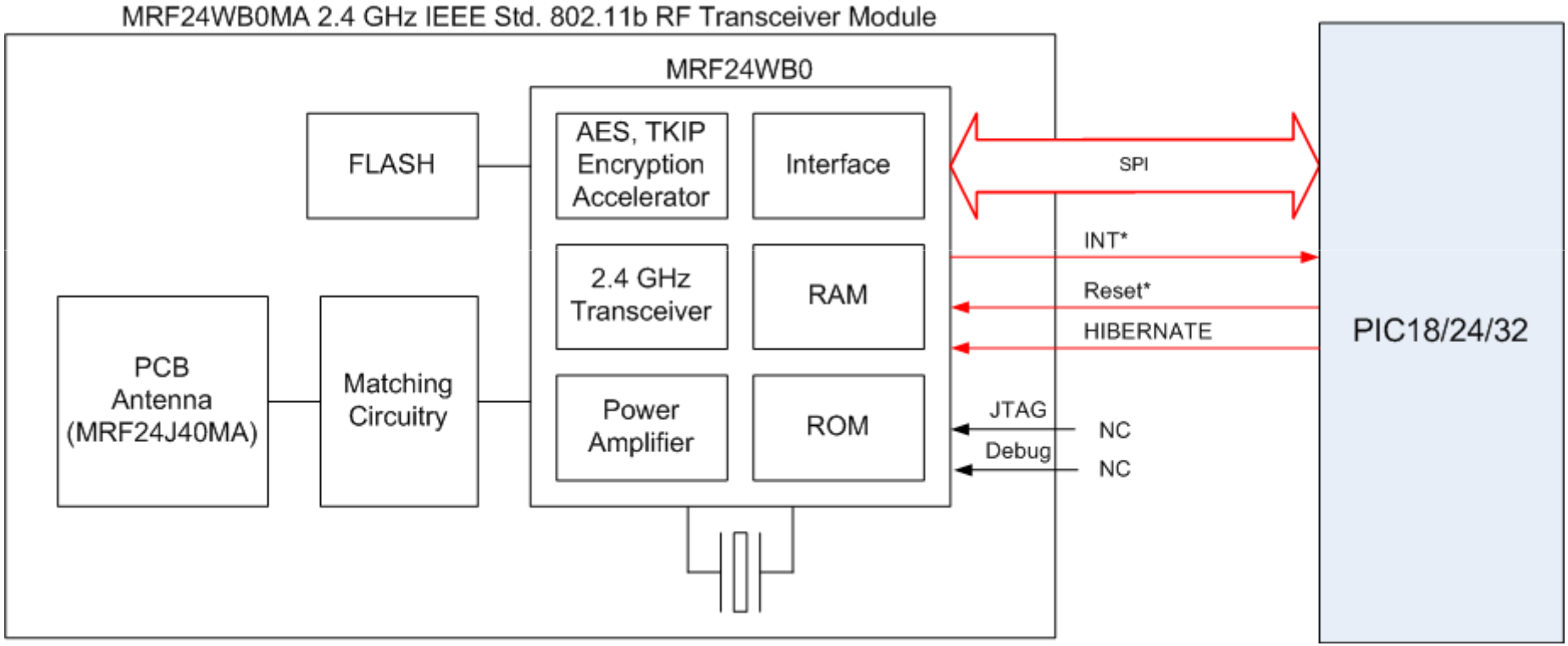


AC164136-4

PICtail™ for development
- PICtail connector
- PICtail Plus connector

Microchip Wi-Fi® Modules

Connection to PIC® MCU



MICROCHIP
Regional Training Centers

Product Versions

	Gen 2
Module	MRF24WB0MA/MB
Roadmap	Supports new Parts
Stack	V5.25 onwards
EZConfig	Built-in application (stack)
ZeroConfig	Built-in application (stack)
Connection Manager	Built-in (module) Saves 500B RAM
Scanning	Anytime
Stack customization	API interface
Configuration	TCPIPConfig.h & WF_Config.h
PICtail™	AC164136-4
Documentation	Stack Help Files

MICROCHIP
Regional Training Centers

Stack Performance

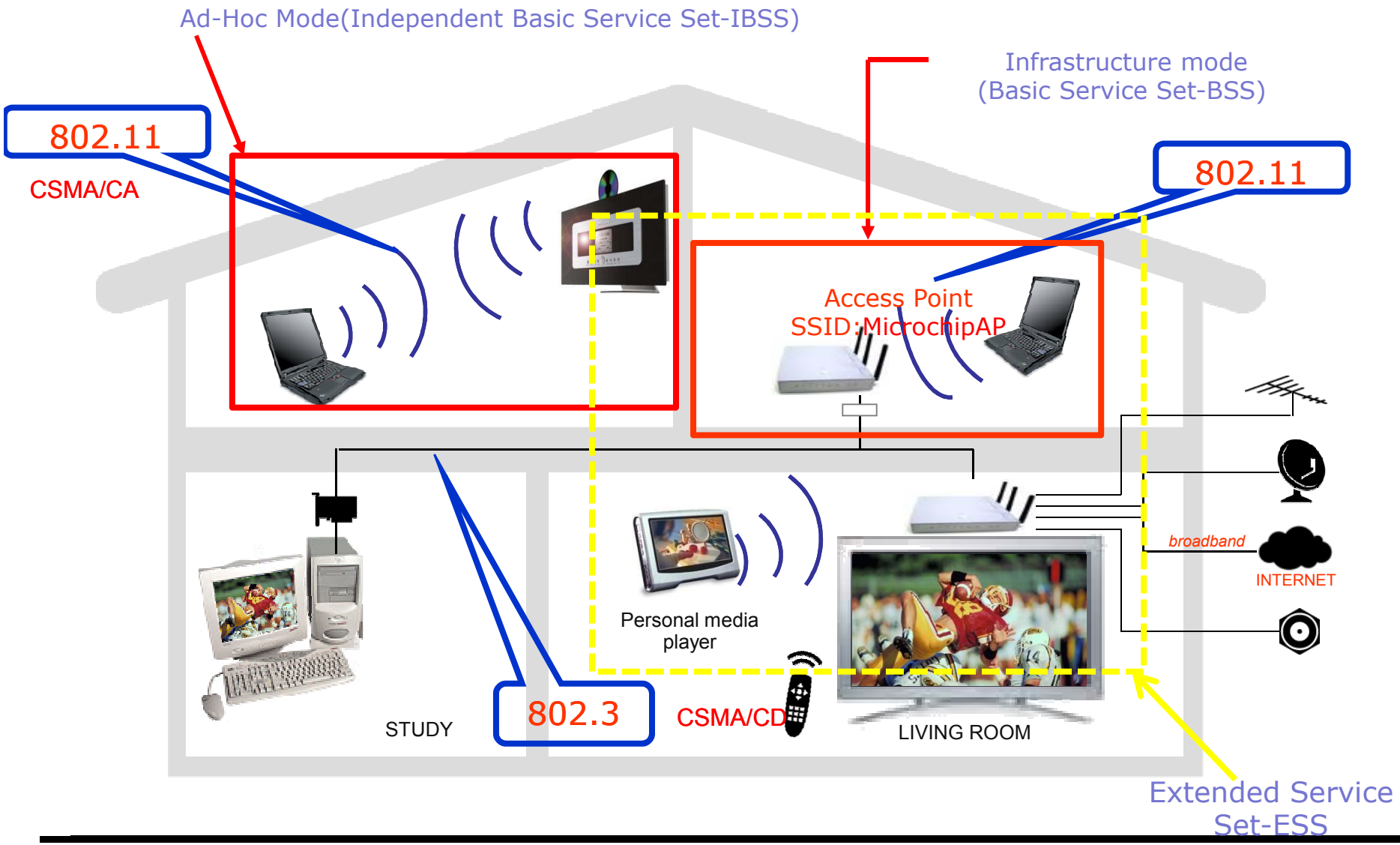
Microcontroller	MIPS	Network Controller	Interface	LAN Transmit Throughput (Kbytes/sec)				Internet Transmit Throughput at 100ms ping (estimated Kbytes/sec)[1]				Compiler	Optimization
				TCP with 200 byte TX FIFO	TCP with 2000 byte TX FIFO	TCP with 8000 byte TX FIFO	UDP	TCP with 200 byte TX FIFO	TCP with 2000 byte TX FIFO	TCP with 8000 byte TX FIFO	UDP		
PIC18F97J60	10.4	Internal 10BaseT	-	33	72	N/A	114	2	20	N/A	114	C18	Debug
PIC18F8722	10	ENC28J60	SPI, 10 MHz	20	44	N/A	64	2	20	N/A	64	C18	Debug
PIC24FJ128GA010	16	ENC28J60	SPI, 8 MHz	49	119	N/A	195	2	20	N/A	195	C30	s (min size)
dsPIC33FJ256GP710[2]	40	ENC28J60	SPI, 8 MHz	68	176	N/A	249	2	20	N/A	249	C30	s (min size)
PIC32MX360F512L	80	ENC28J60	SPI, 20 MHz	109	287	N/A	428	2	20	N/A	428	C32	s (min size)
PIC32MX795F512L	80	ENC28J60	SPI, 20 MHz	120	300	N/A	435	2	20	N/A	435	C32	s (min size)
PIC18F8722	10	ENC624J600[3]	SPI, 10 MHz	28	75	80	115	2	20	80	115	C18	Debug
PIC24FJ128GA010	16	ENC624J600[3]	SPI, 8 MHz	56	161	175	368	2	20	80	368	C30	s (min size)
dsPIC33FJ256GP710[2]	40	ENC624J600[3]	SPI, 8 MHz	91	276	310	559	2	20	80	559	C30	s (min size)
PIC32MX360F512L	80	ENC624J600[3]	SPI, 13.33 MHz	139	356	412	687	2	20	80	687	C32	s (min size)
PIC32MX795F512L	80	ENC624J600[3]	SPI, 13.33 MHz	152	396	457	801	2	20	80	801	C32	s (min size)
PIC24FJ128GA010	16	ENC624J600[3]	PSP Mode 5, PMP	101	300	339	1043	2	20	80	1043	C30	s (min size)
dsPIC33FJ256GP710[2]	40	ENC624J600[3]	PSP Mode 5, Bitbang	196	659	813	2063	2	20	80	2063	C30	s (min size)
PIC32MX360F512L	80	ENC624J600[3]	PSP Mode 5, PMP	203	786	996	2115	2	20	80	2115	C32	s (min size)
PIC32MX795F512L[3]	80	Internal 100BaseTX	-	434	1555	2562	8723	2	20	80	8723	C32	s (min size)
PIC18F97J60	10.4	MRF24WB0M	SPI, 10.4 MHz	5	24	30	45	2	20	30	45	C18	Debug
PIC24FJ128GA010	16	MRF24WB0M	SPI, 8 MHz	8	45	67	45	2	20	67	45	C30	s (min size)
dsPIC33FJ256GP710[2]	40	MRF24WB0M	SPI, 8 MHz	9	66	97	45	2	20	80	45	C30	s (min size)
PIC32MX360F512L	80	MRF24WB0M	SPI, 20 MHz	9	66	100	47	2	20	80	47	C32	s (min size)
PIC32MX795F512L	80	MRF24WB0M	SPI, 20 MHz	9	67	102	48	2	20	80	48	C32	s (min size)

MICROCHIP

Regional Training Centers

Overview of 802.11 Networks

Overview of 802.11 Network



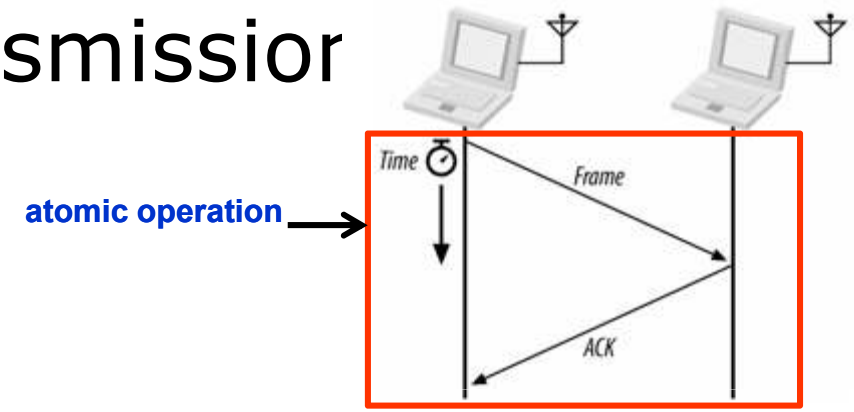
MICROCHIP
Regional Training Centers

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

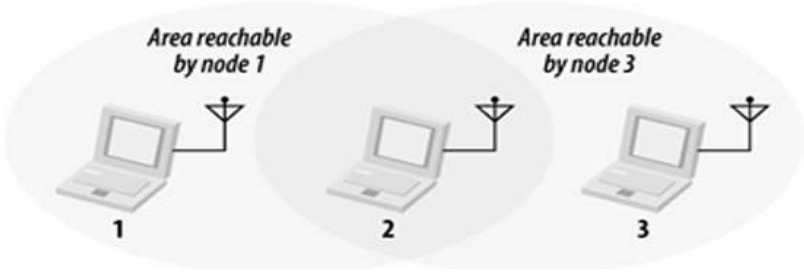
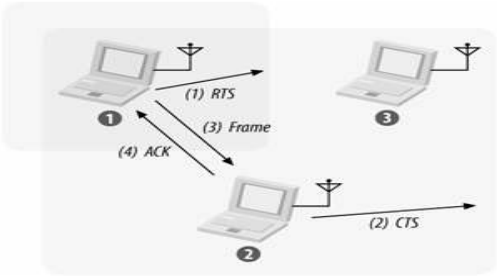


Figure- Nodes 1 and 3 are "hidden"



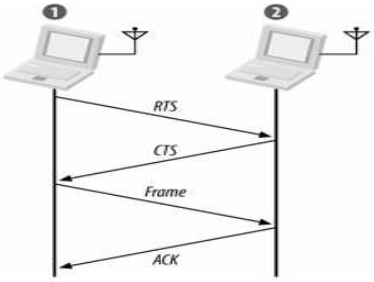
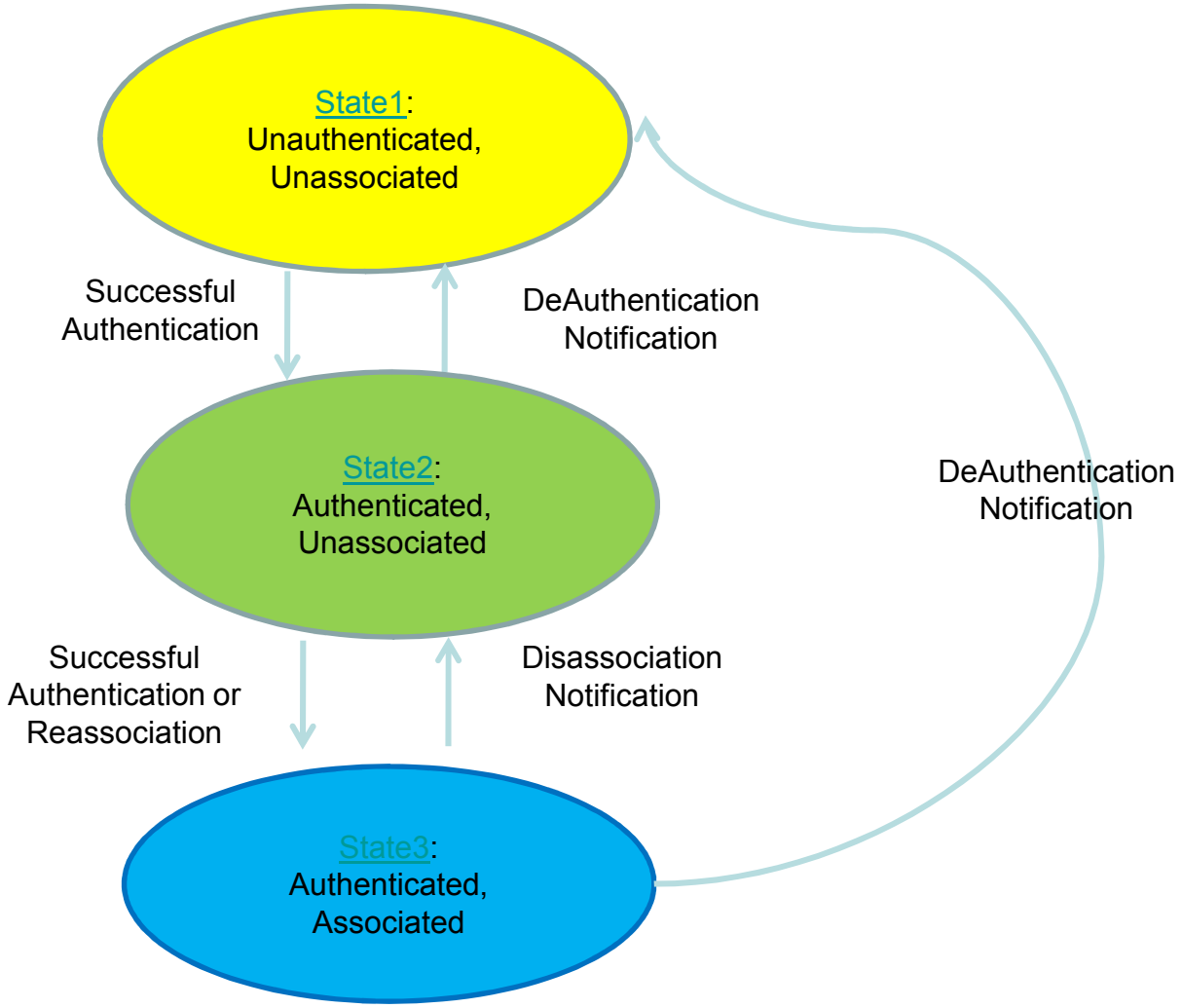


Figure- RTS/CTS clearing

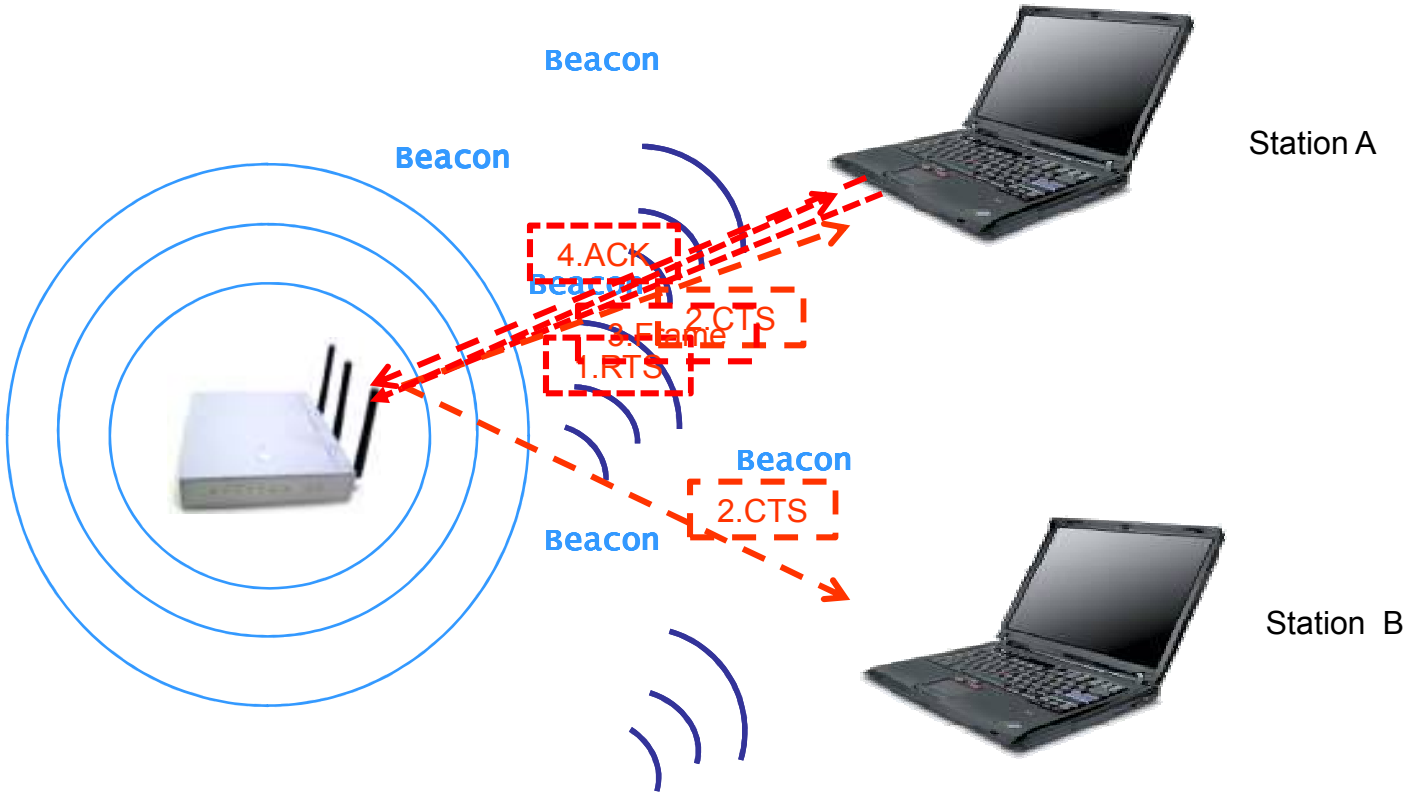
Frame Transmission & Association States

- Station connect to AP(Access Point)



Frame Transmission & Association States

- State 1(Unauthenticated, Unassociated)



Beacon-Management Frame

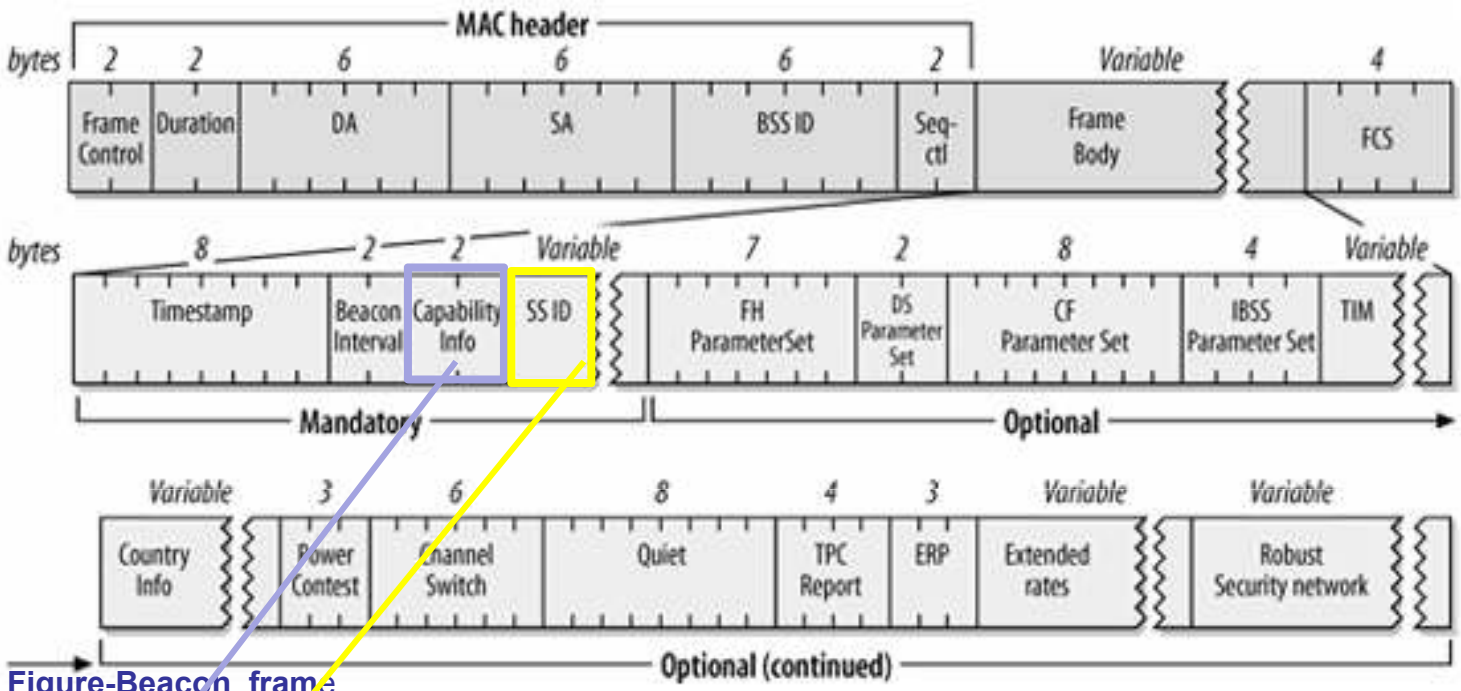
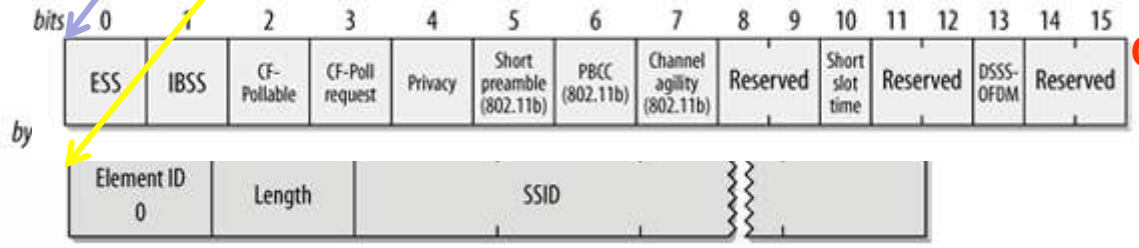


Figure-Beacon frame



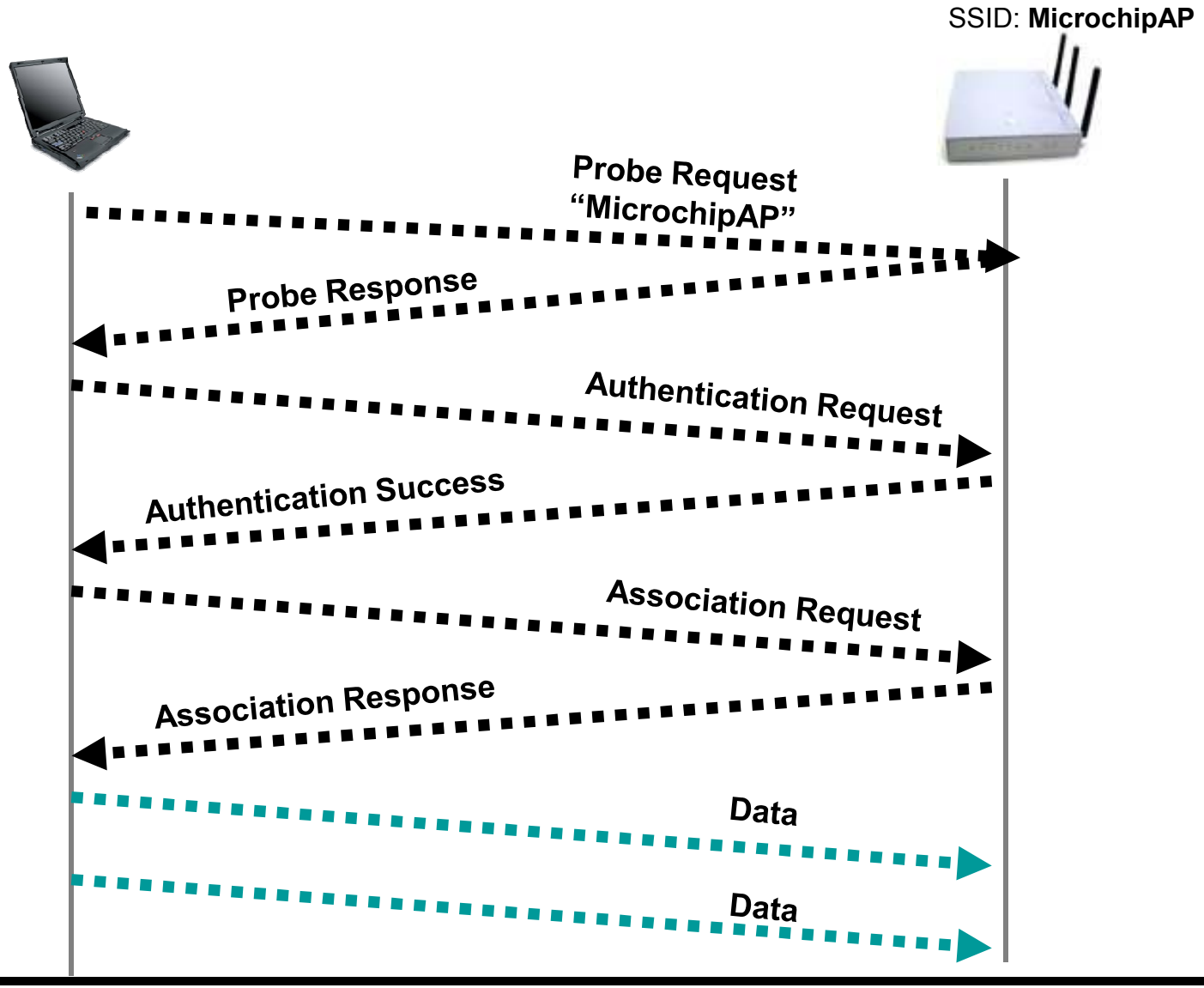
Capability Information field

* 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.

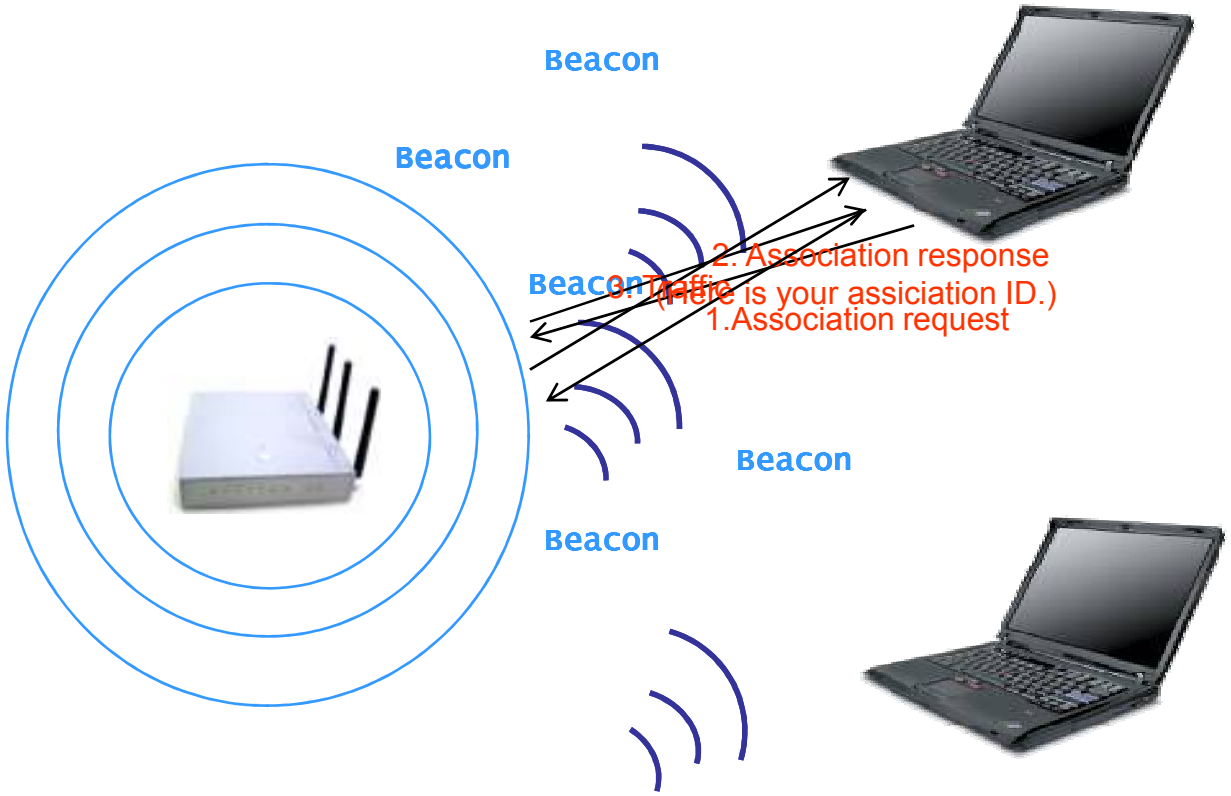
Probe Request/Response (Management Frame)



Back State2

Frame Transmission & Association States

- State 2(Authenticated, Unassociated)



Disassociation and de-authentication (Management Frame)

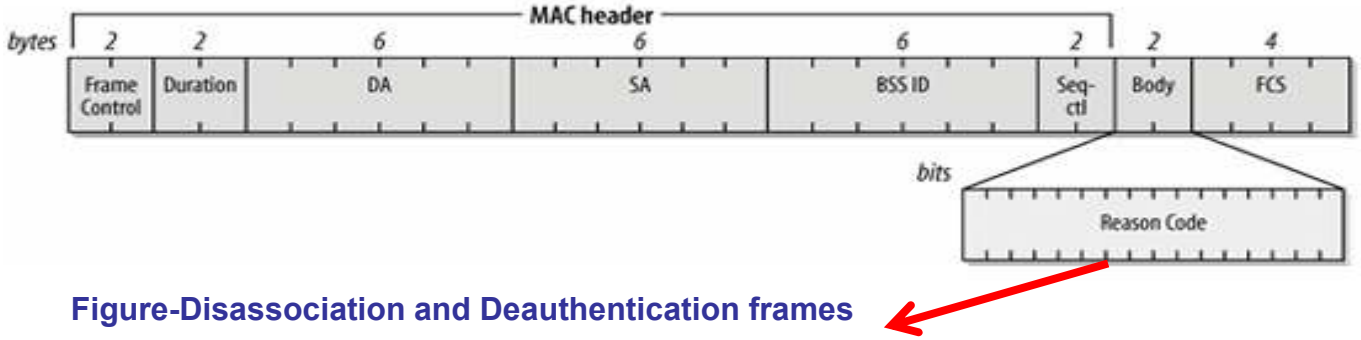


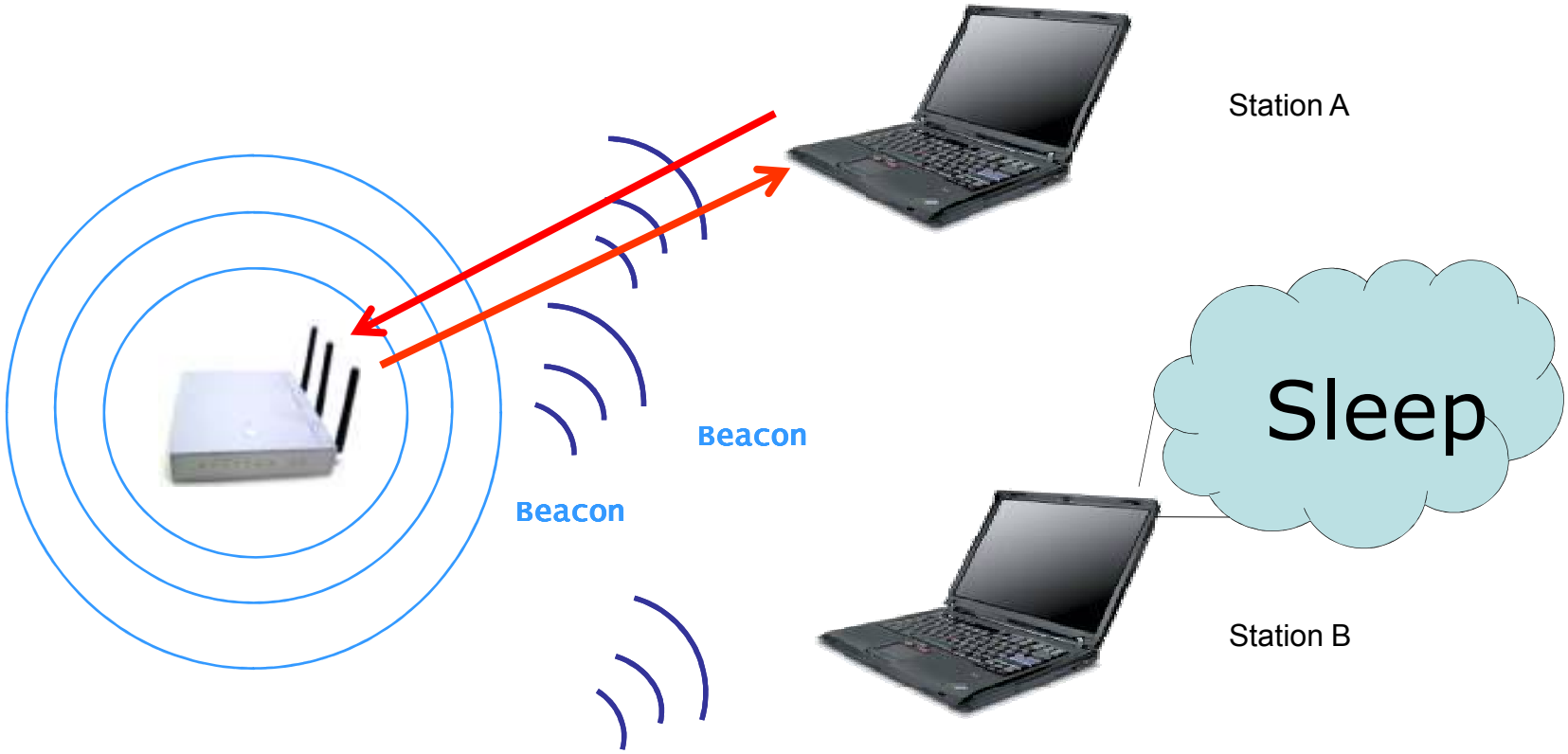
Figure-Disassociation and Deauthentication frames

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

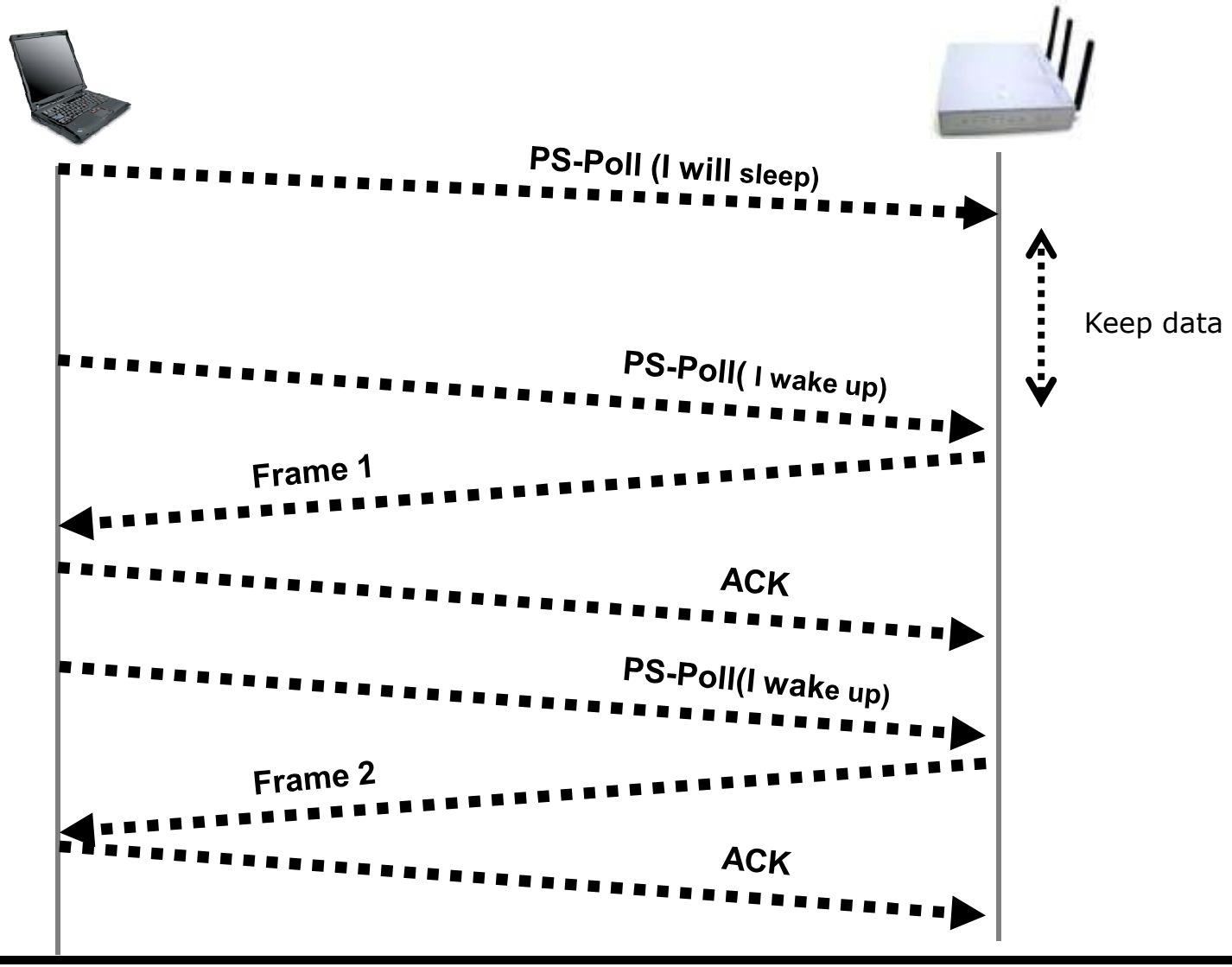
Back State3

Frame Transmission & Association States

- State 3(Authenticated/Associated)



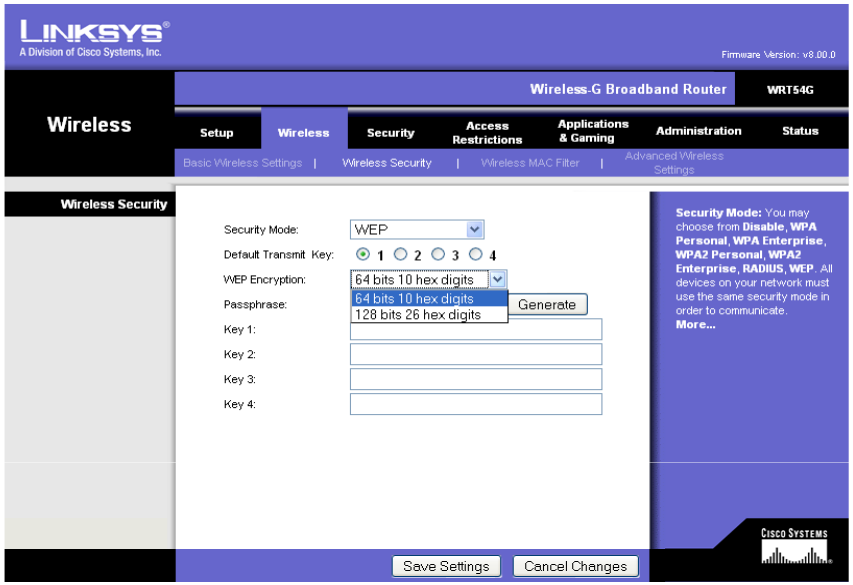
Power-Save Poll (PS-Poll) Control Frame

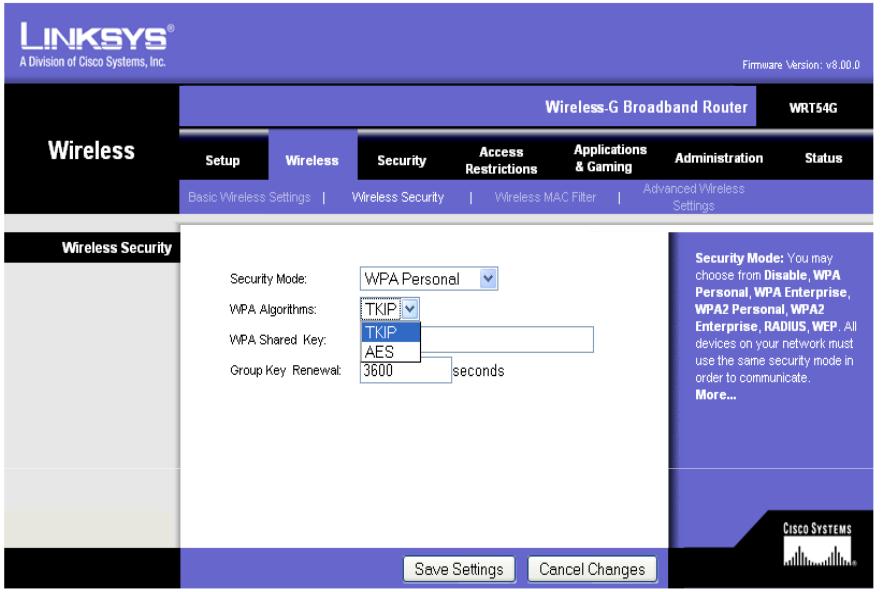


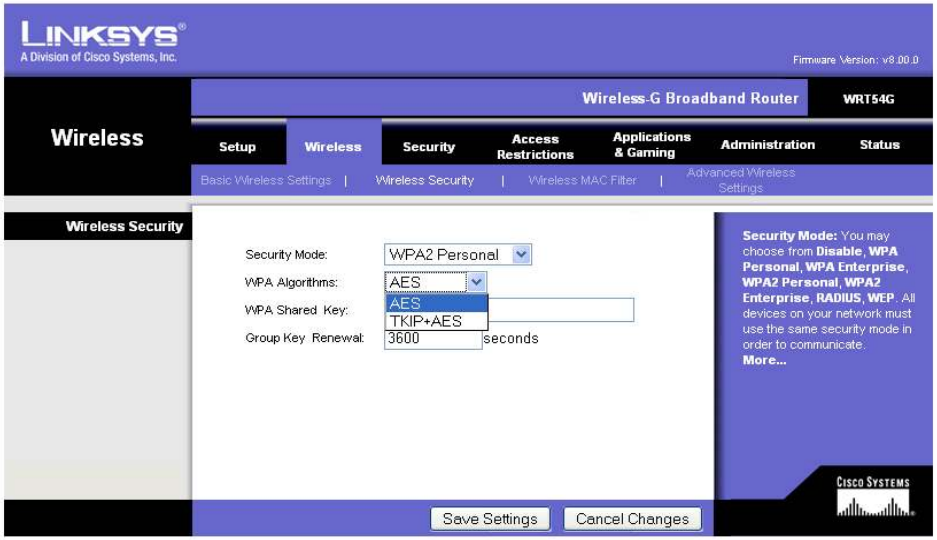
IEEE 802.11 Security

- 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

Wireless Security Setting

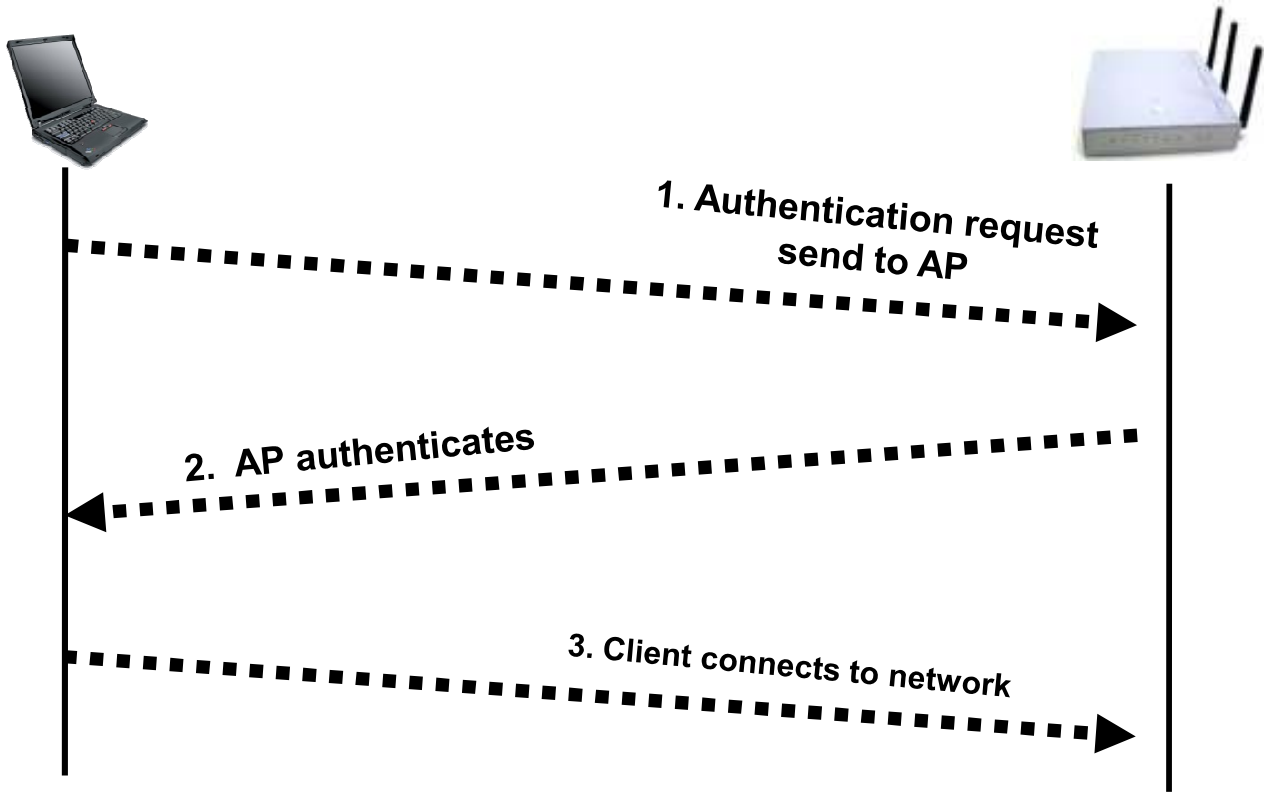






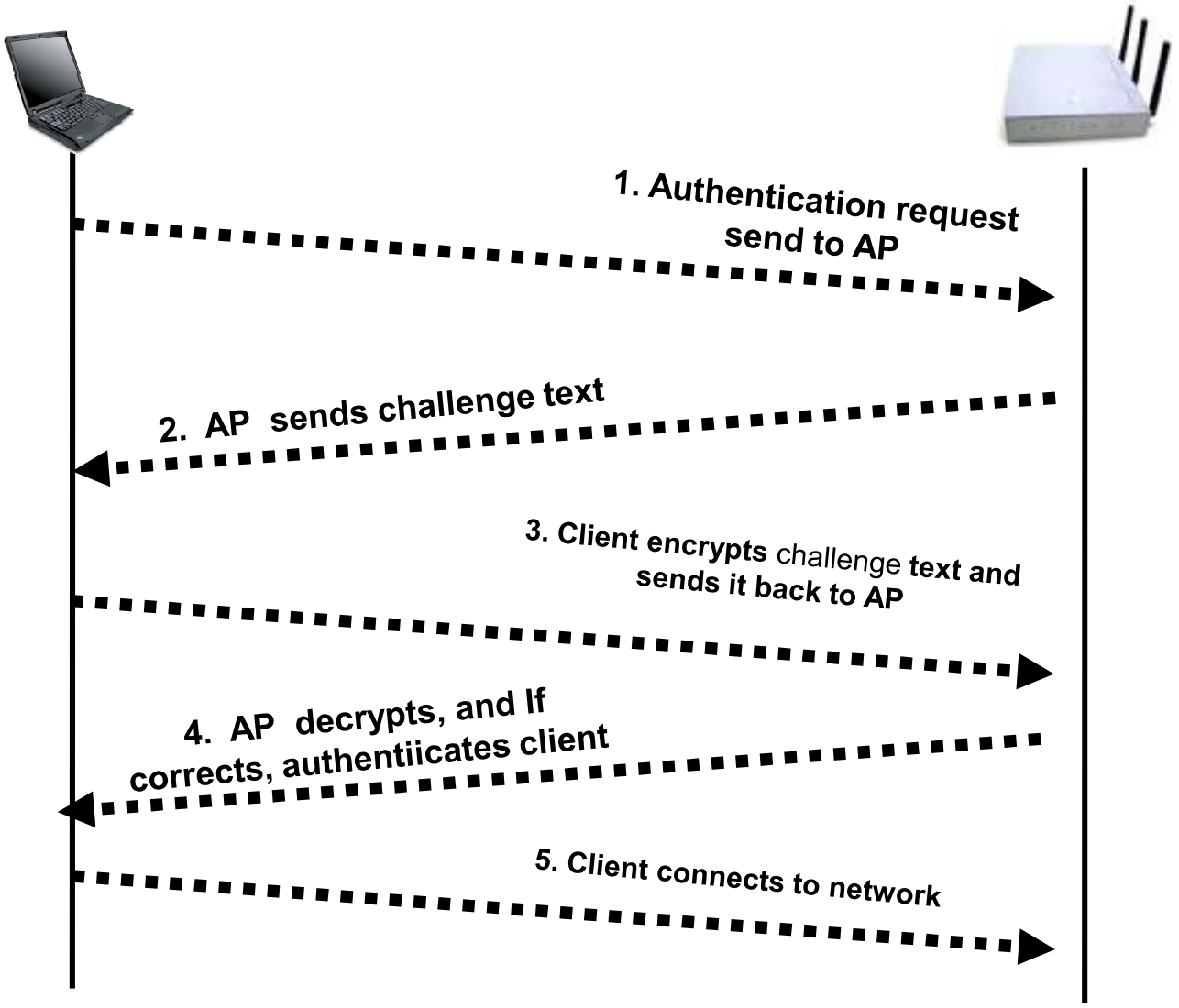
MICROCHIP
Regional Training Centers

Open System Authentication

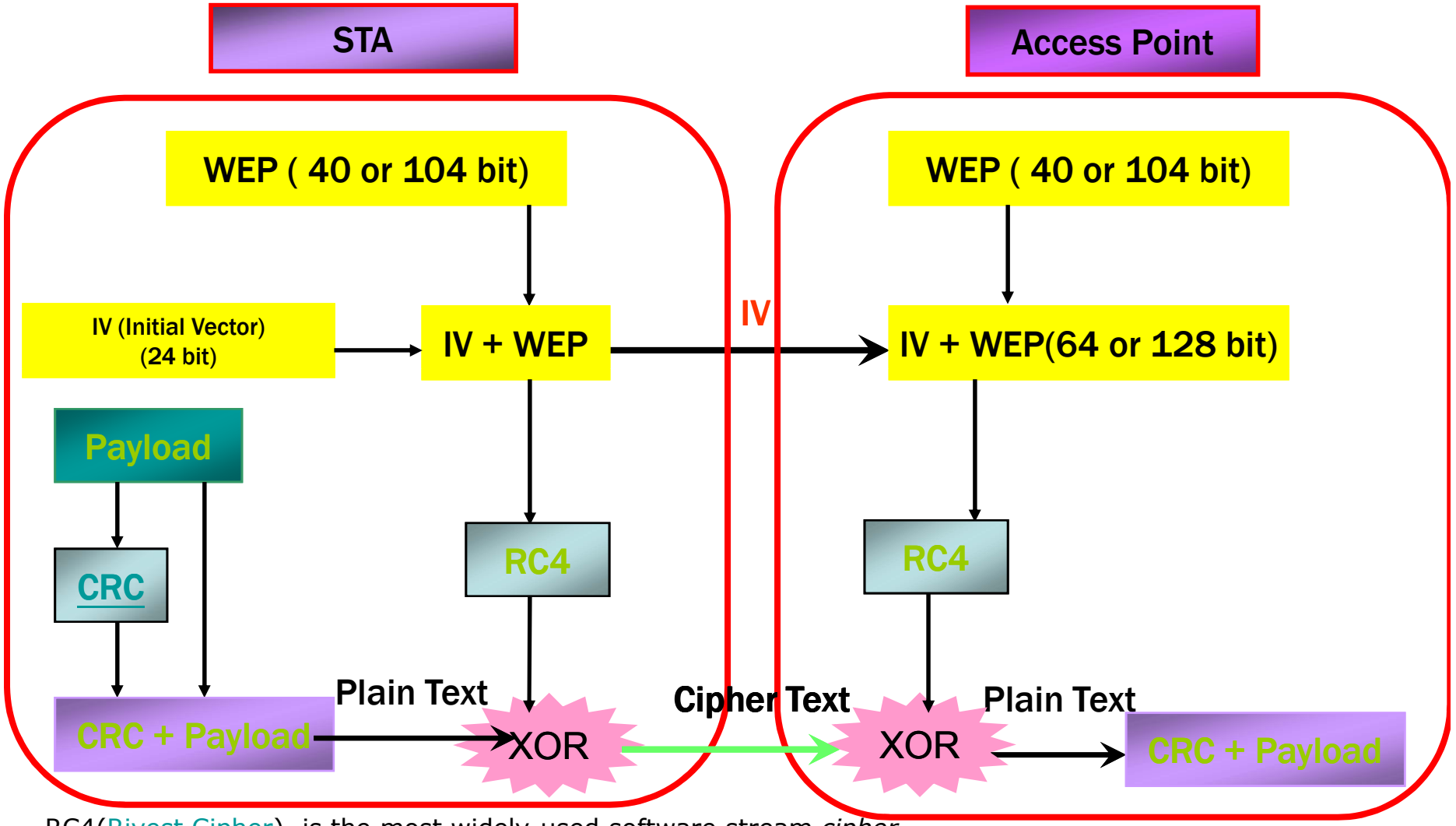


MICROCHIP
Regional Training Centers

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

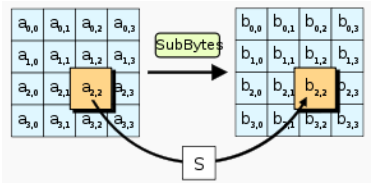
- The IV is too small and in cleartext.
- The IV is static (0→ 16777216)
- The IV makes the key stream vulnerable.
- The IV is a part of the RC4 encryption key
- WEP provides no cryptographic integrity protection

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)



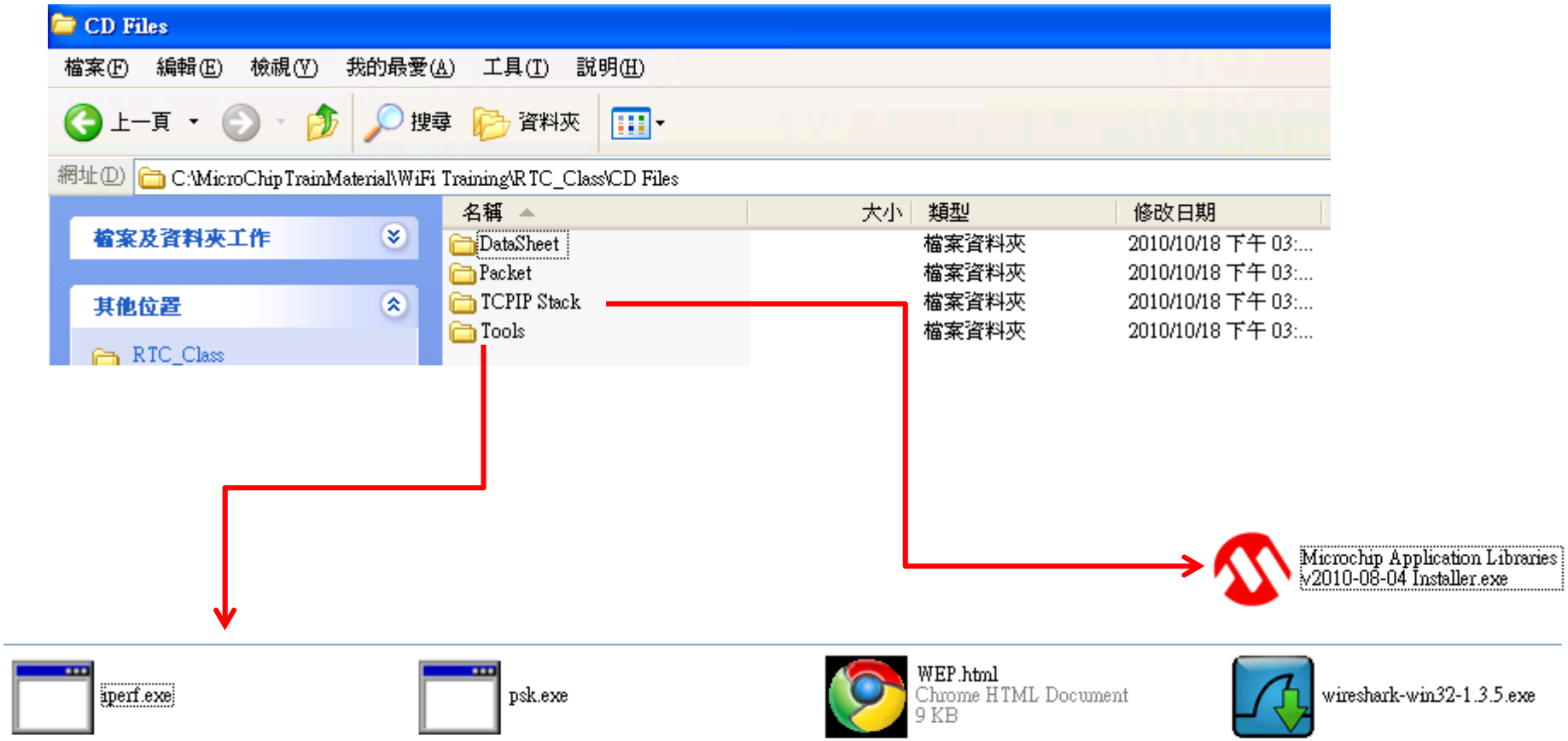
Michael: Message Integrity Code(MIC)

CCM（Counter-Mode/CBC-MAC)

MICROCHIP
Regional Training Centers

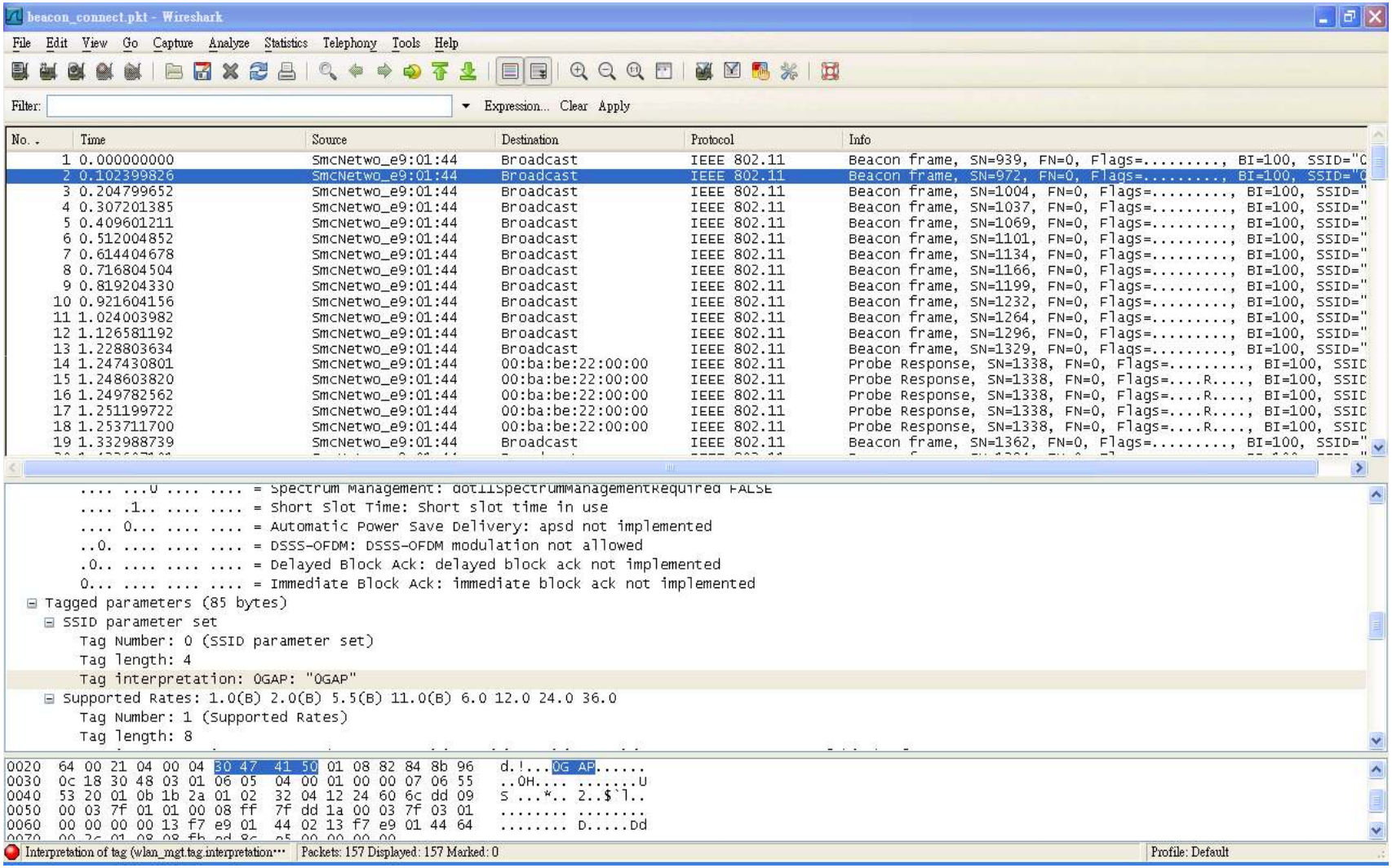
Wireless Experiment

CD Folder



MICROCHIP
Regional Training Centers

WireShark



Test Environment

Web Server

00:1e:c0:00:1f:ff

00:16:ea:c5:c1:14

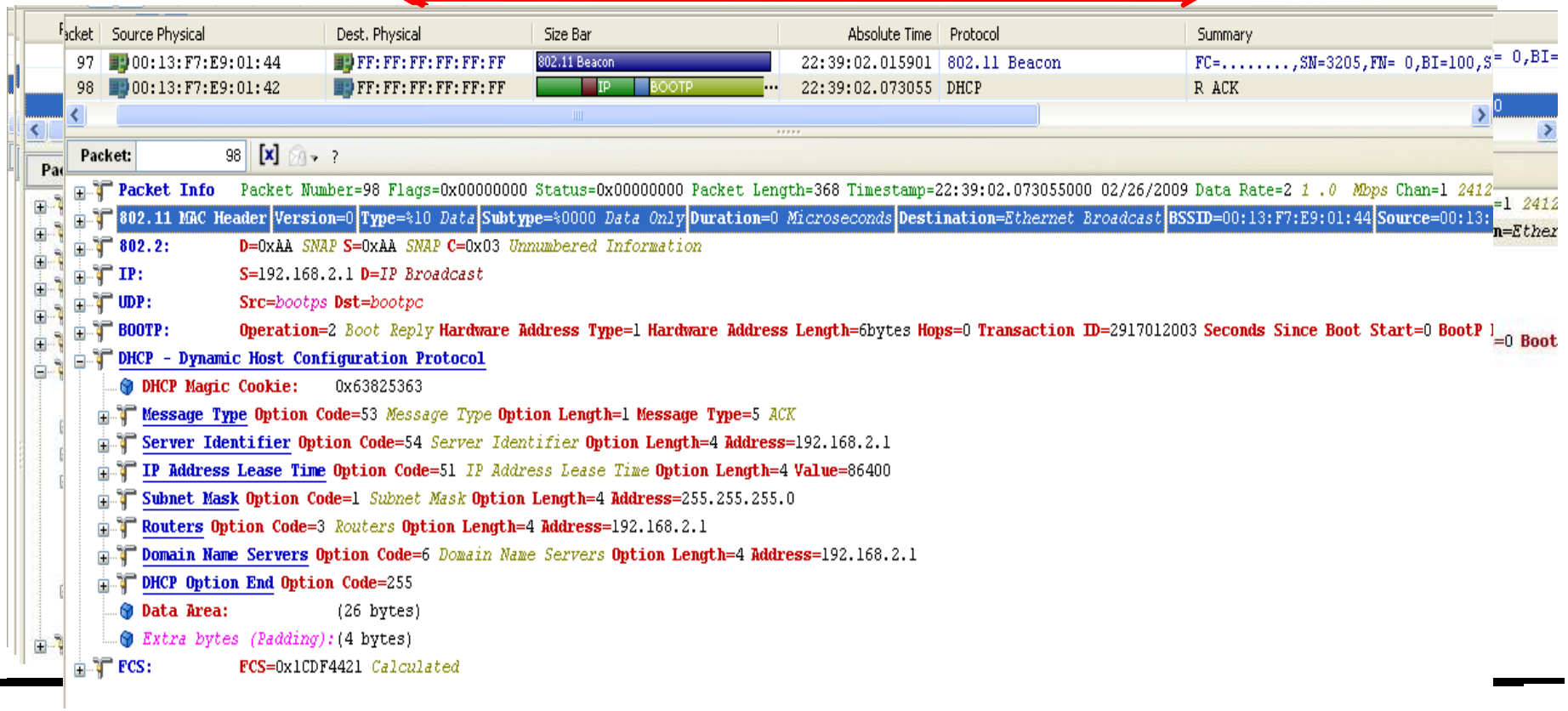
00:1a:70:d4:e8:52

The DHCP lease process

Explorer 16+ZeroG PicTail
(00:BA:BE:22:00:00)

Wireless AP(00:13:F7:E9:01:44)

5. DHCP Request
6. DHCP Reply server send ARP Packet (who use 192.168.2.100)



MICROCHIP

Regional Training Centers

Wireless Network Deployment & Wireless AP Feature

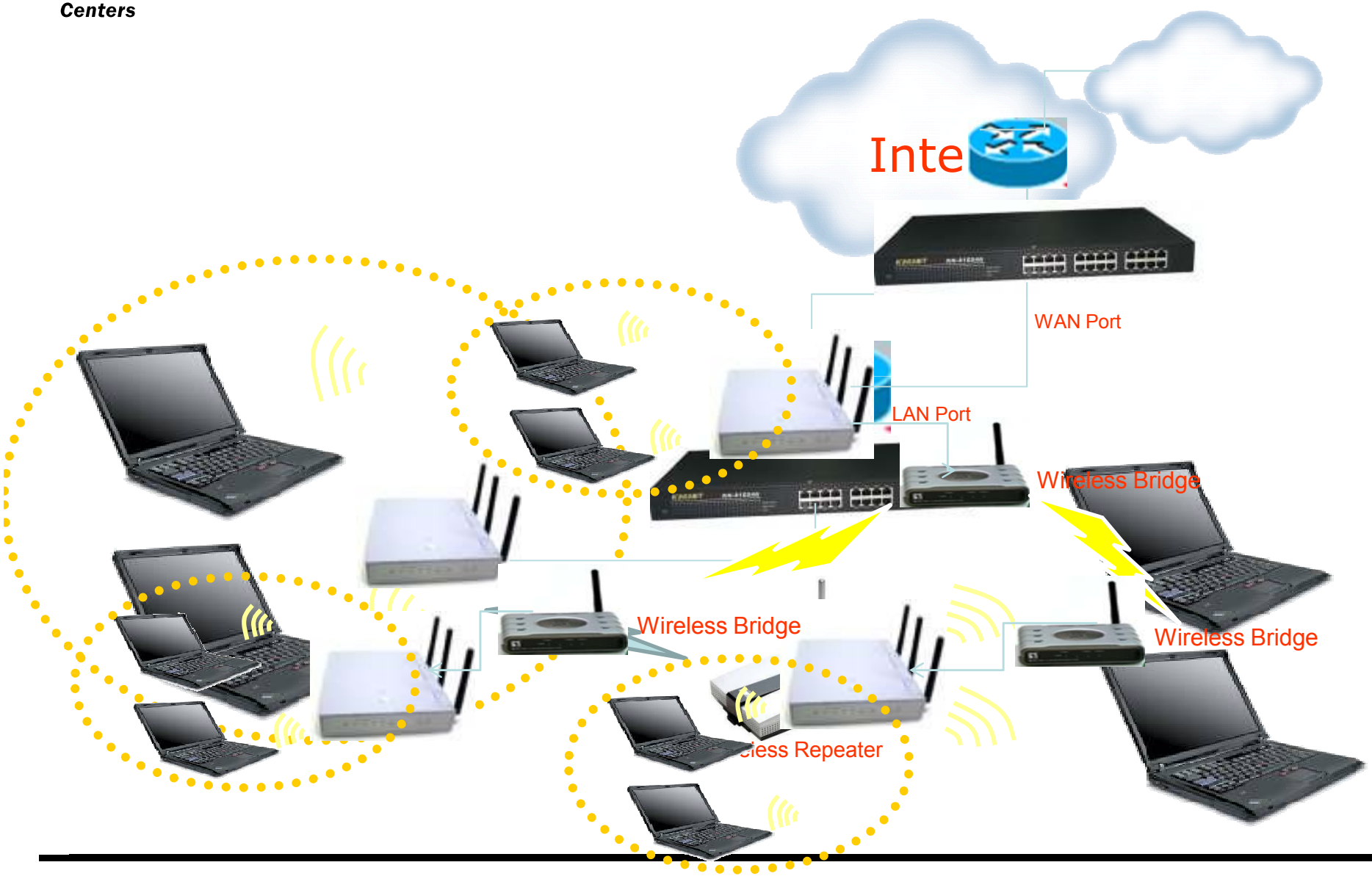
MICROCHIP
Regional Training Centers

Wireless Network Deployment

- Root Mode
- Repeater Mode
- Bridge Mode

MICROCHIP
Regional Training Centers

Wireless Network Deployment



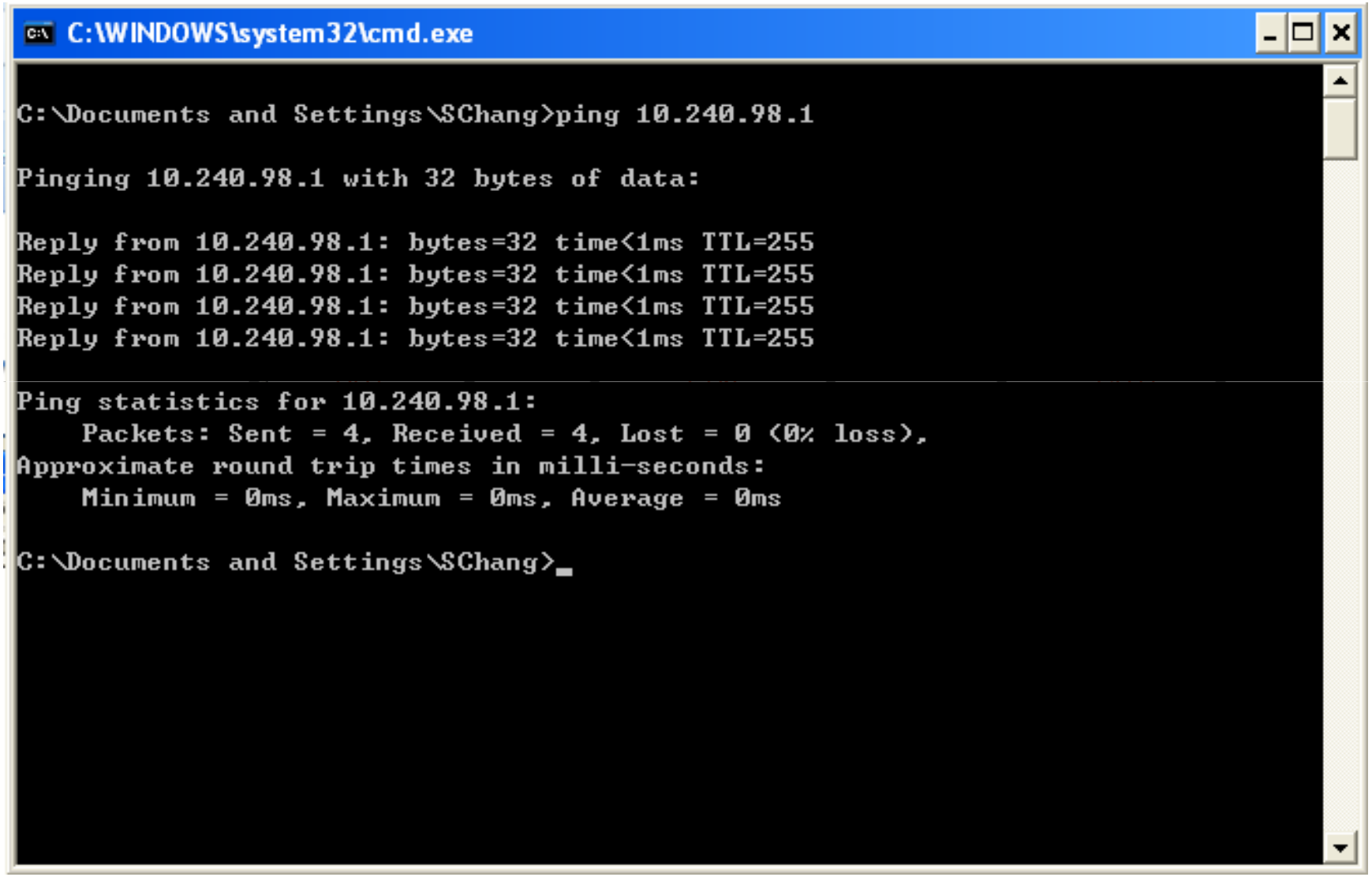
MICROCHIP
Regional Training Centers

Wireless Network Deployment

MICROCHIP
Regional Training Centers

802.11 Network Deployment

Back

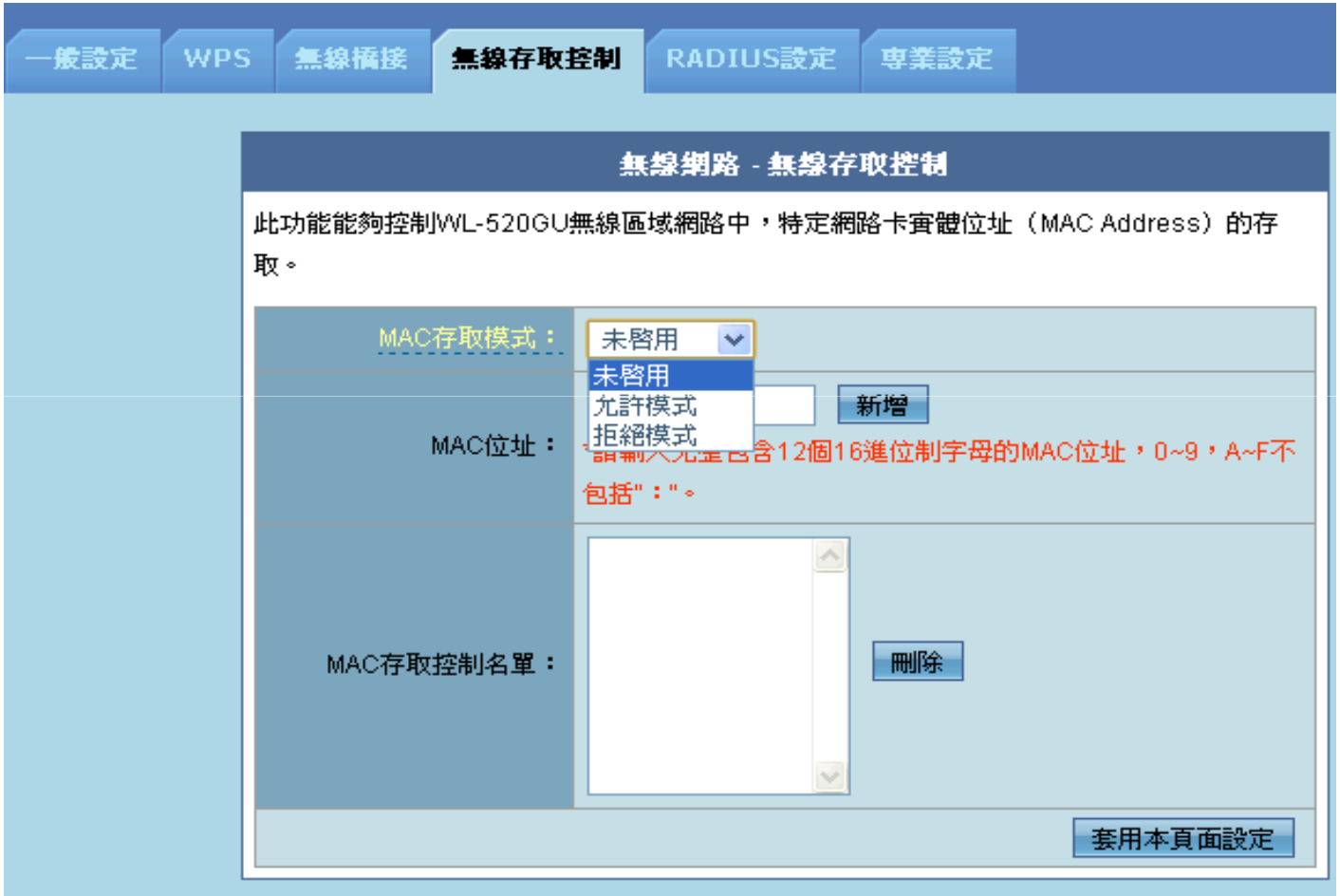


MICROCHIP
Regional Training Centers

Wireless AP Features

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

MAC Address Control



MICROCHIP
Regional Training Centers

Bandwidth Management

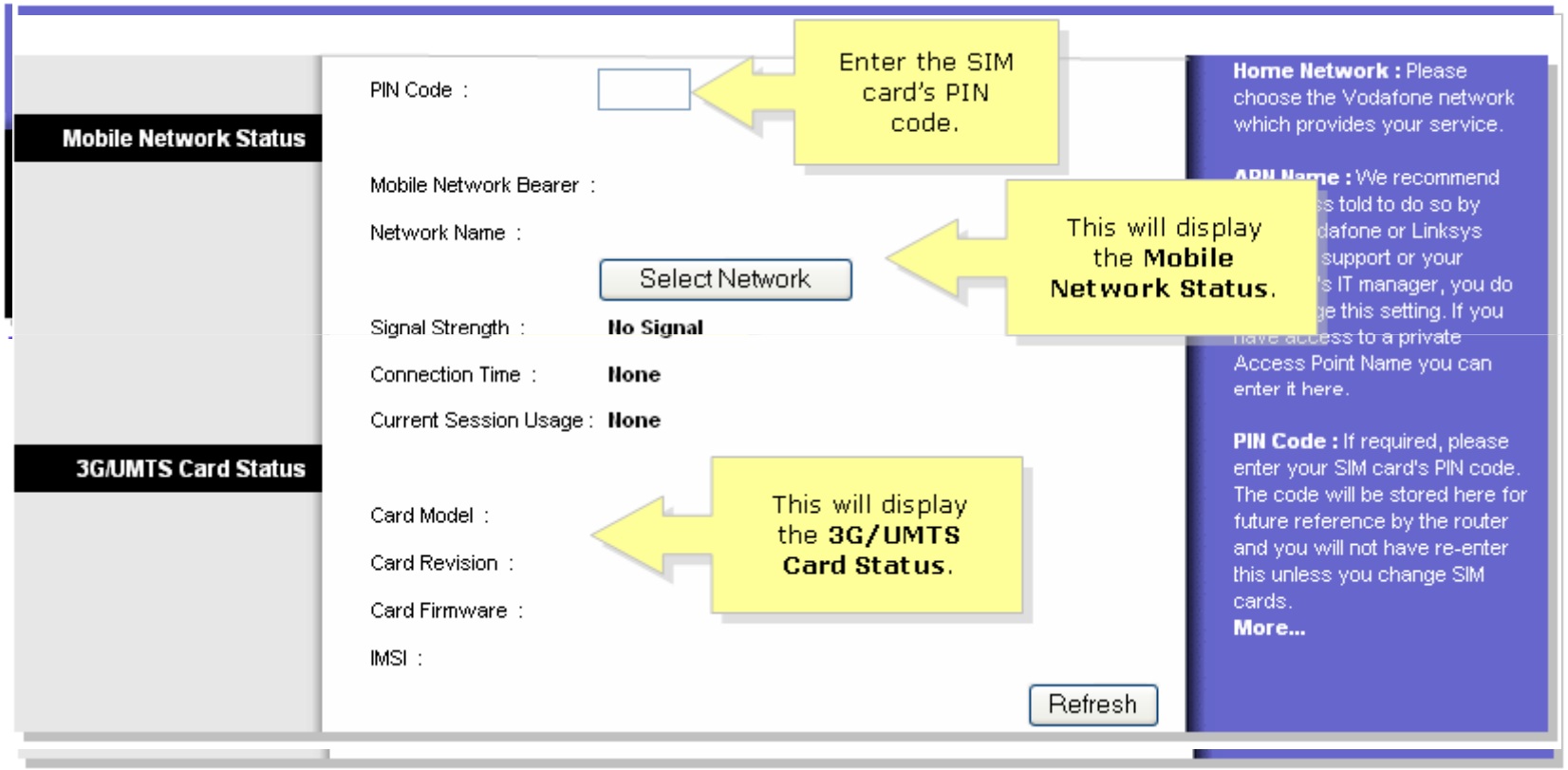
Port Trigger

Virtual Server

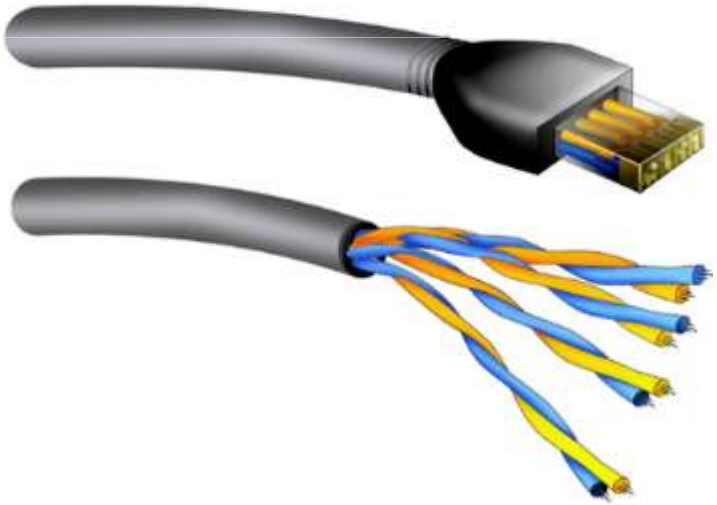
DDNS

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 -

MICROCHIP
Regional Training Centers

Wireless AP Setting (Handon)

MICROCHIP

Regional Training Centers

Creating Wi-Fi Enabled Applications

MICROCHIP
Regional Training Centers

Adding WiFi® to Microchip

- Start w/Microchip TCPIP Stack v5.25
- Comes with several demonstrations
  - TCPIP Wi-Fi Demo App
  - TCPIP WiFi Console Demo App
  - TCPIP WiFi EasyConfig Demo App
- Provides an out-of-box demo to start
- Options for your application

DV240001

DM163024

AC164136-4

MICROCHIP
Regional Training Centers

Product Configuration

- The TCPIP Stack provides:
  - Default values in code
  - Configuration application wizard
  - “C” functions to allow configuration changes
  - Run time configuration change capability – example application

Stack Configurations

TCPIPConfig.h

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

```
#define STACK_USE_UART                  // Application demo using UART for IP address display and stack con
//#define STACK_USE_UART2TCP_BRIDGE     // UART to TCP Bridge application example
//#define STACK_USE_IP_GLEANING
#define STACK_USE_ICMP_SERVER           // Ping query and response capability
//#define STACK_USE_ICMP_CLIENT         // Ping transmission capability
//#define STACK_USE_HTTP_SERVER         // Old HTTP server
#define STACK_USE_HTTP2_SERVER          // New HTTP server with POST, Cookies, Authentication, etc.
//#define STACK_USE_SSL_SERVER          // SSL server socket support (Requires SW300052)
//#define STACK_USE_SSL_CLIENT          // SSL client socket support (Requires SW300052)
#define STACK_USE_AUTO_IP               // Dynamic link-layer IP address automatic configuration protocol
#define STACK_USE_DHCP_CLIENT           // Dynamic Host Configuration Protocol client for obtaining IP addre
//#define STACK_USE_DHCP_SERVER         // Single host DHCP server
```

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

```
#define MY_DEFAULT_HOST_NAME            "MCHPBOARD"

#define MY_DEFAULT_MAC_BYTE1            (0x00)  // Use the default of
#define MY_DEFAULT_MAC_BYTE2            (0x04)  // 00-04-A3-00-00-00 if using
#define MY_DEFAULT_MAC_BYTE3            (0xA3)  // an ENCX24J600 or MRF24WB0M
#define MY_DEFAULT_MAC_BYTE4            (0x00)  // and wish to use the internal
#define MY_DEFAULT_MAC_BYTE5            (0x00)  // factory programmed MAC
#define MY_DEFAULT_MAC_BYTE6            (0x00)  // address instead.

#define MY_DEFAULT_IP_ADDR_BYTE1        (169ul)
#define MY_DEFAULT_IP_ADDR_BYTE2        (254ul)
#define MY_DEFAULT_IP_ADDR_BYTE3        (1ul)
#define MY_DEFAULT_IP_ADDR_BYTE4        (1ul)

#define MY_DEFAULT_MASK_BYTE1           (255ul)
#define MY_DEFAULT_MASK_BYTE2           (255ul)
#define MY_DEFAULT_MASK_BYTE3           (0ul)
#define MY_DEFAULT_MASK_BYTE4           (0ul)
```

Change MAC

NOTE: 00:04:A3:00:00:00

WiFi® Configurations

WF_Config.h

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

```
/*-------------------------------------------*/
/* Default settings for Connection Management */
/*-------------------------------------------*/
#define MY_DEFAULT_SSID_NAME                "MicrochipDemoAP"

#define MY_DEFAULT_NETWORK_TYPE             WF_INFRASTRUCTURE   /* WF_INFRASTRUCTURE or WF_ADHOC     */

#define MY_DEFAULT_SCAN_TYPE                WF_ACTIVE_SCAN      /* WF_ACTIVE_SCAN or WF_PASSIVE_SCAN */

#define MY_DEFAULT_CHANNEL_LIST             {1,6,11}            /* use {} to scan all channels       */

#define MY_DEFAULT_LIST_RETRY_COUNT         (3)

#define MY_DEFAULT_EVENT_NOTIFICATION_LIST  (WF_NOTIFY_CONNECTION_ATTEMPT_SUCCESSFUL  |          \
                                             WF_NOTIFY_CONNECTION_ATTEMPT_FAILED      |          \
                                             WF_NOTIFY_CONNECTION_TEMPORARILY_LOST    |          \
                                             WF_NOTIFY_CONNECTION_PERMANENTLY_LOST    |          \
                                             WF_NOTIFY_CONNECTION_REESTABLISHED)

#define MY_DEFAULT_PS_POLL                  WF_DISABLED         /* WF_DISABLED or WF_ENABLED */

#define MY_DEFAULT_WIFI_SECURITY_MODE       WF_SECURITY_OPEN

//#define USE_MRF24W_HOST_BUFFER

//#define STACK_USE_EZ_CONFIG
//#define EZ_CONFIG_SCAN
//#define EZ_CONFIG_STALL
//#define EZ_CONFIG_STORE
```

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 MRF24WBOM
//   WF_SECURITY_WPA_WITH_PASS_PHRASE      : WPA-PSK Personal where passphrase is given to MRF24WBOM and it c
//   WF_SECURITY_WPA2_WITH_KEY             : WPA2-PSK Personal where binary key is given to MRF24WBOM
//   WF_SECURITY_WPA2_WITH_PASS_PHRASE     : WPA2-PSK Personal where passphrase is given to MRF24WBOM 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

WF_Config.h

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

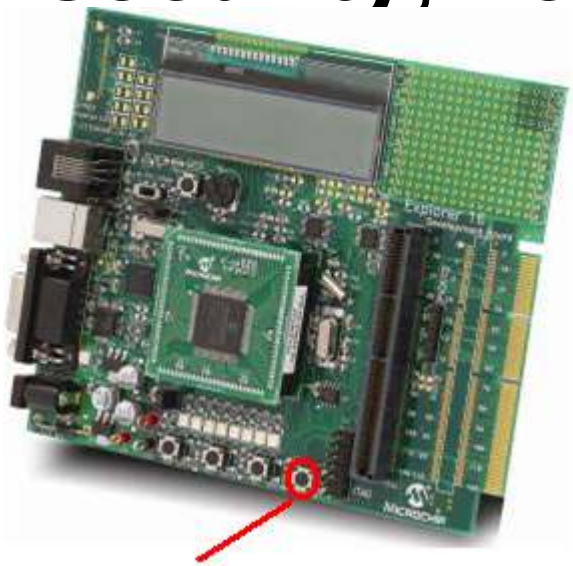
```
// Default pass phrase used for WF_SECURITY_WPA_WITH_PASS_PHRASE and
// WF_SECURITY_WPA2_WITH_PASS_PHRASE security modes
#define MY_DEFAULT_PSK_PHRASE                "Microchip 802.11 Secret PSK Password"


// If using security mode of WF_SECURITY_WPA_WITH_KEY or WF_SECURITY_WPA2_WITH_KEY, then this section
// must be set to  match the key for MY_DEFAULT_SSID_NAME and MY_DEFAULT_PSK_PHRASE
// combination.  The values below are derived from the SSID "MicrochipDemoAP" and the pass phrase
// "Microchip 802.11 Secret PSK Password".
// The tool at http://www.wireshark.org/tools/wpa-psk.html can be used to generate this field.
#define MY_DEFAULT_PSK "\
\x86\xC5\x1D\x71\xD9\x1A\xAA\x49\
\x40\xC8\x88\xC6\xE9\x7A\x4A\xD5\
\xE5\x6D\xDA\x44\x8E\xFB\x9C\x0A\
\xE1\x47\x81\x52\x31\x1C\x13\x7C"


//------------------------------------------------------------------------------
// Default WEP keys used in WF_SECURITY_WEP_40  and WF_SECURITY_WEP_104 security mode
//------------------------------------------------------------------------------
#define MY_DEFAULT_WEP_PHRASE            "WEP Phrase"

// string 4 40-bit WEP keys -- corresponding to passphraseof "WEP Phrase"
#define MY_DEFAULT_WEP_KEYS_40 "\
\x5a\xfb\x6c\x8e\x77\
\xc1\x04\x49\xfd\x4e\
\x43\x18\x2b\x33\x88\
\xb0\x73\x69\xf4\x78"
```

Security Details

MICROCHIP
Regional Training Centers

Configure Infrastructure

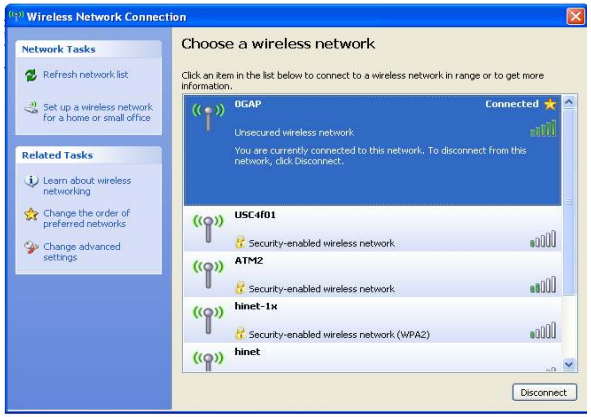
- Demo Applications are pre-configured
- To change defaults, erase EEPROM
- To change: security, network



Test Procedure(ping command)

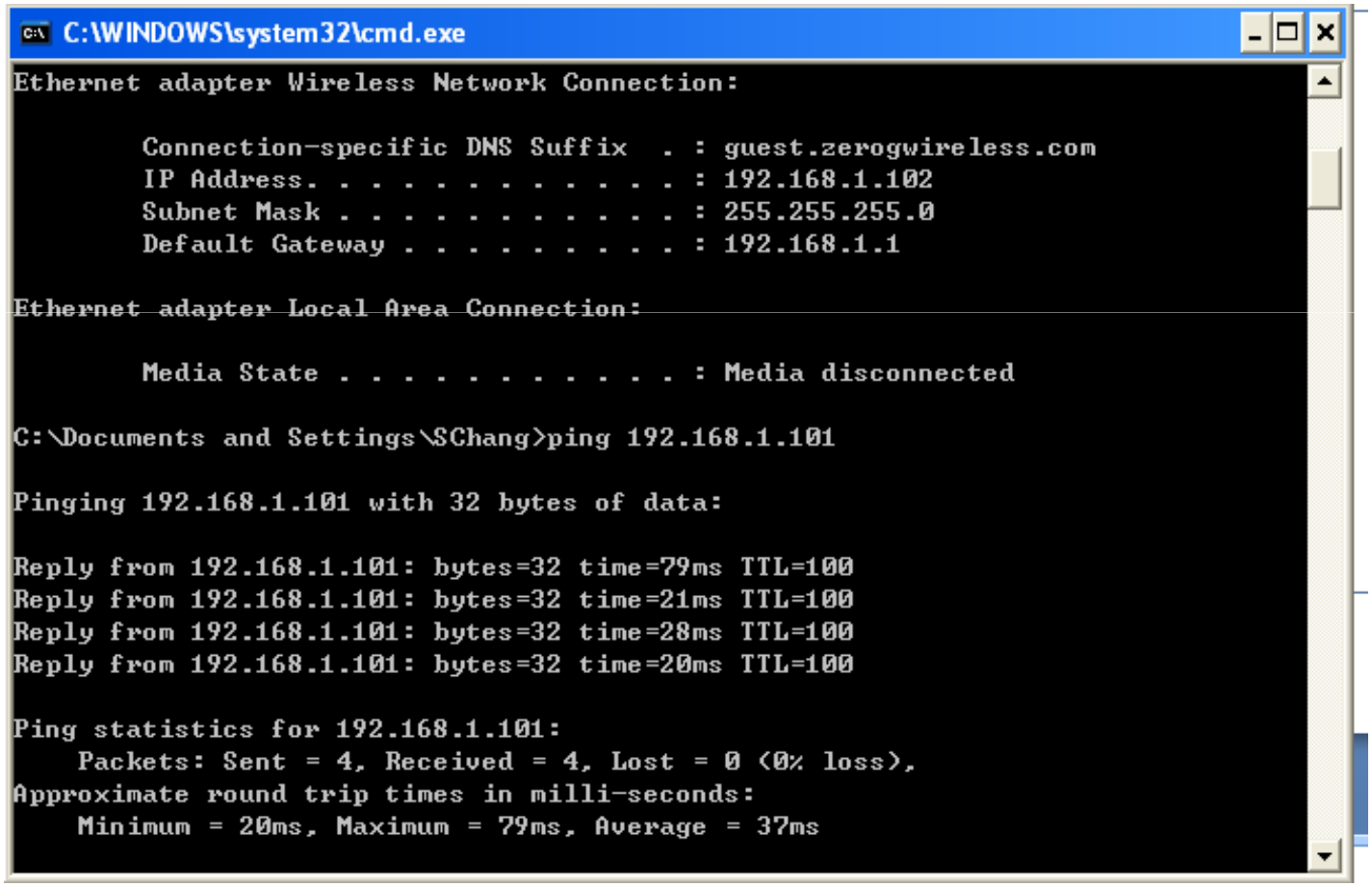
- See the IP Address on **Explorer 16** Board

- Verify connection from PC/access point



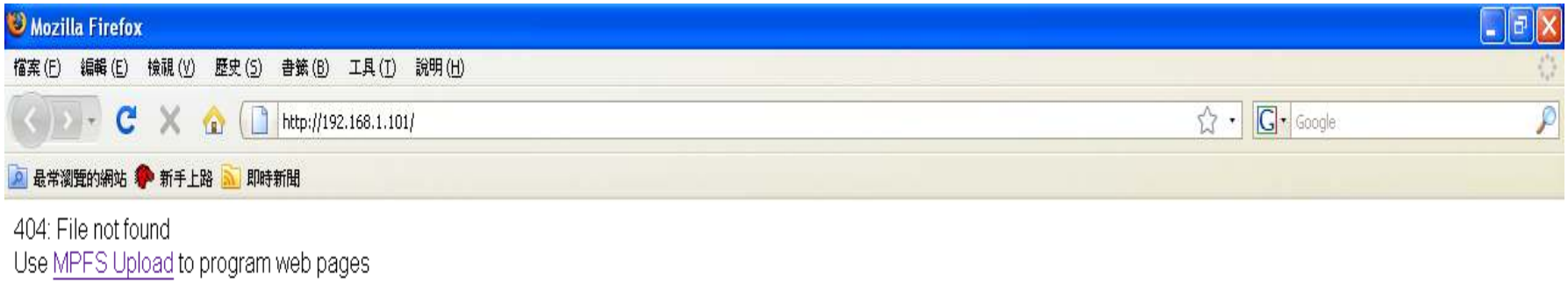
Test Procedure(ping command)

- **Ping Command**



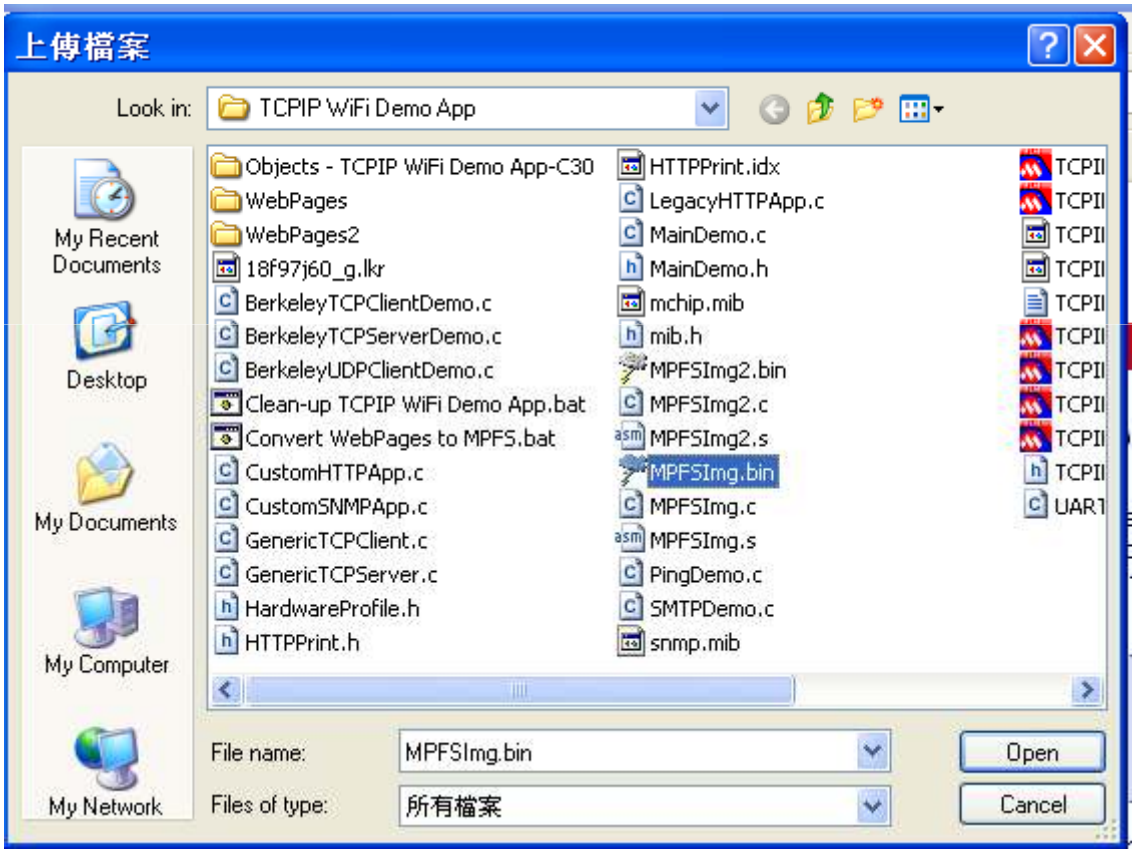
Test Procedure(webserver)

- Brower on Explorer 16 Board



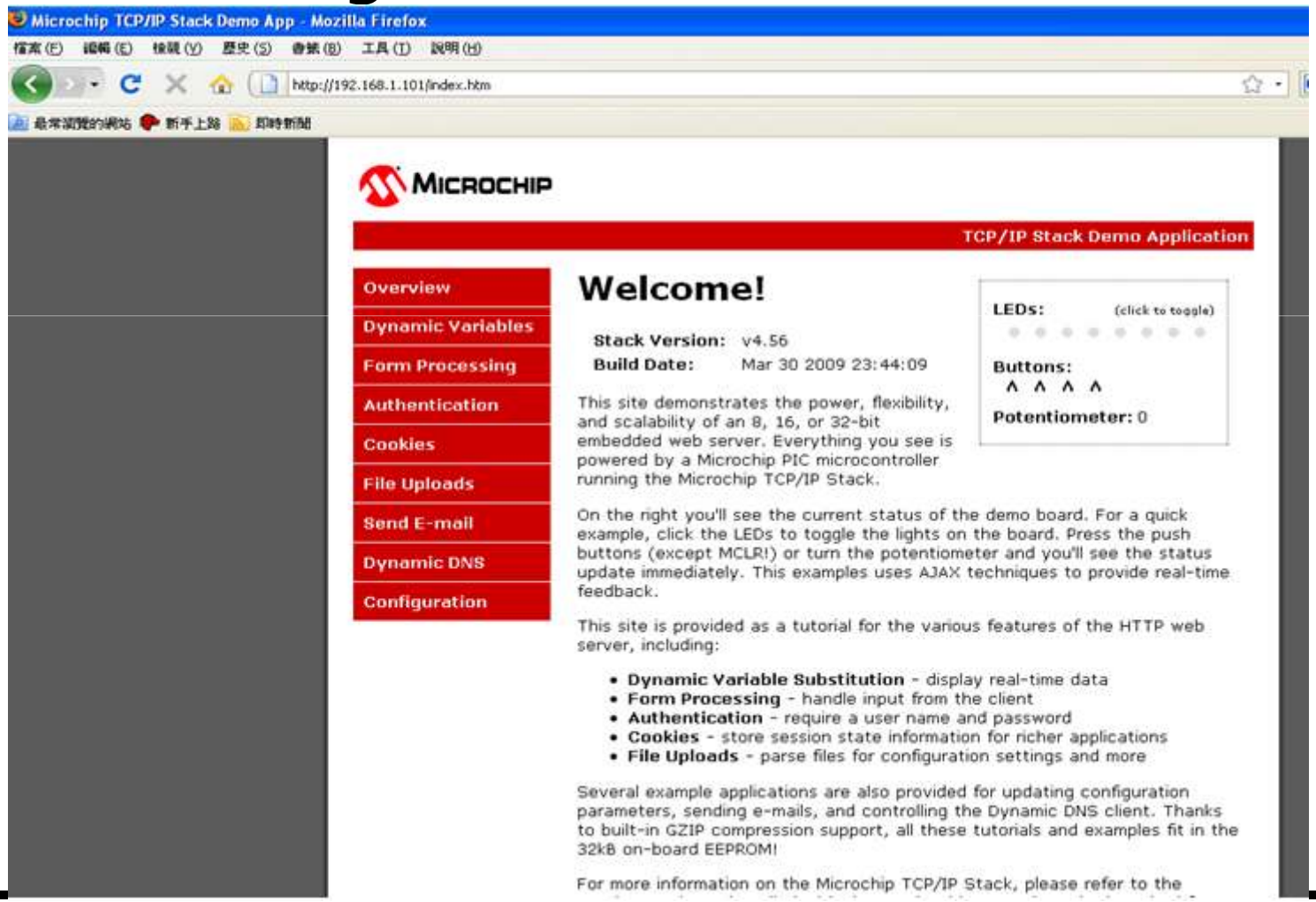
Test Procedure(webserver)

- Upload MPFSImg.bin

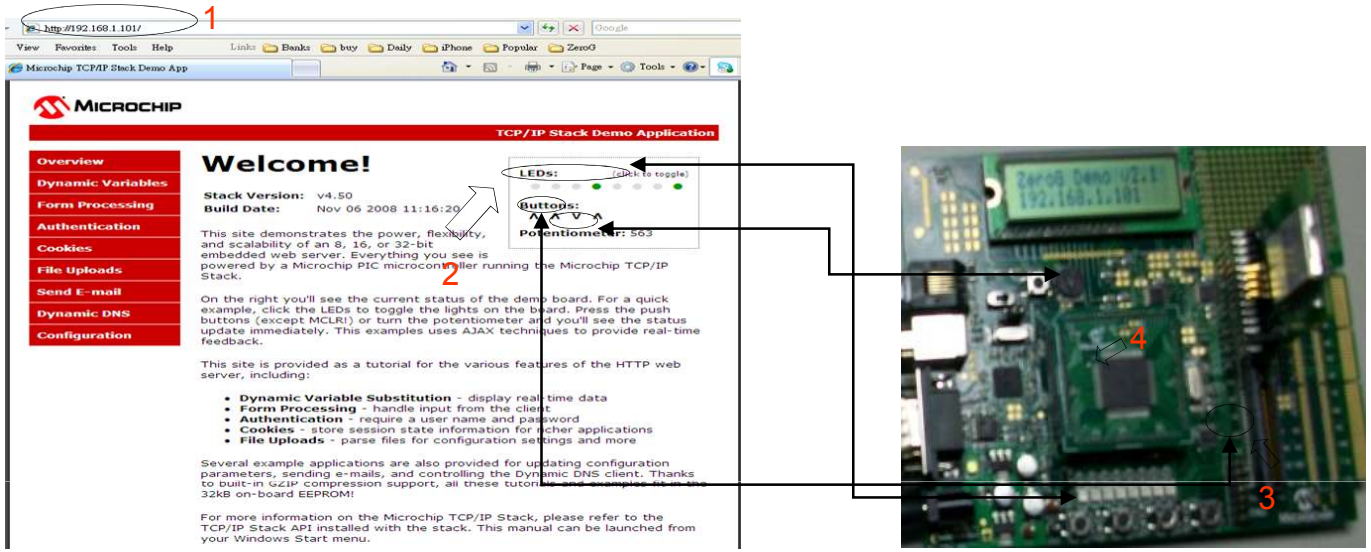


Test Procedure(WebServer)

- Browser again



Web Page Control



1. Input 192.168.1.10 in a browser to display a Microchip web server page
2. Pointing the radio buttons on the web page to change LED lights on the Exploer 16/Picdem.net2 board
3. Click the button to change up/down arrow direction on the web page (Do not click any other buttons)
4. Turning the knob to change number display on the web page

MICROCHIP
Regional Training Centers

Handon

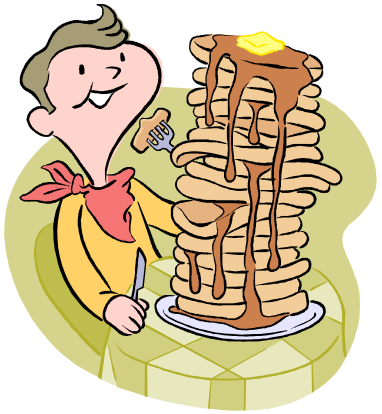
MICROCHIP

Regional Training Centers

Software Architecture

MICROCHIP
Regional Training Centers

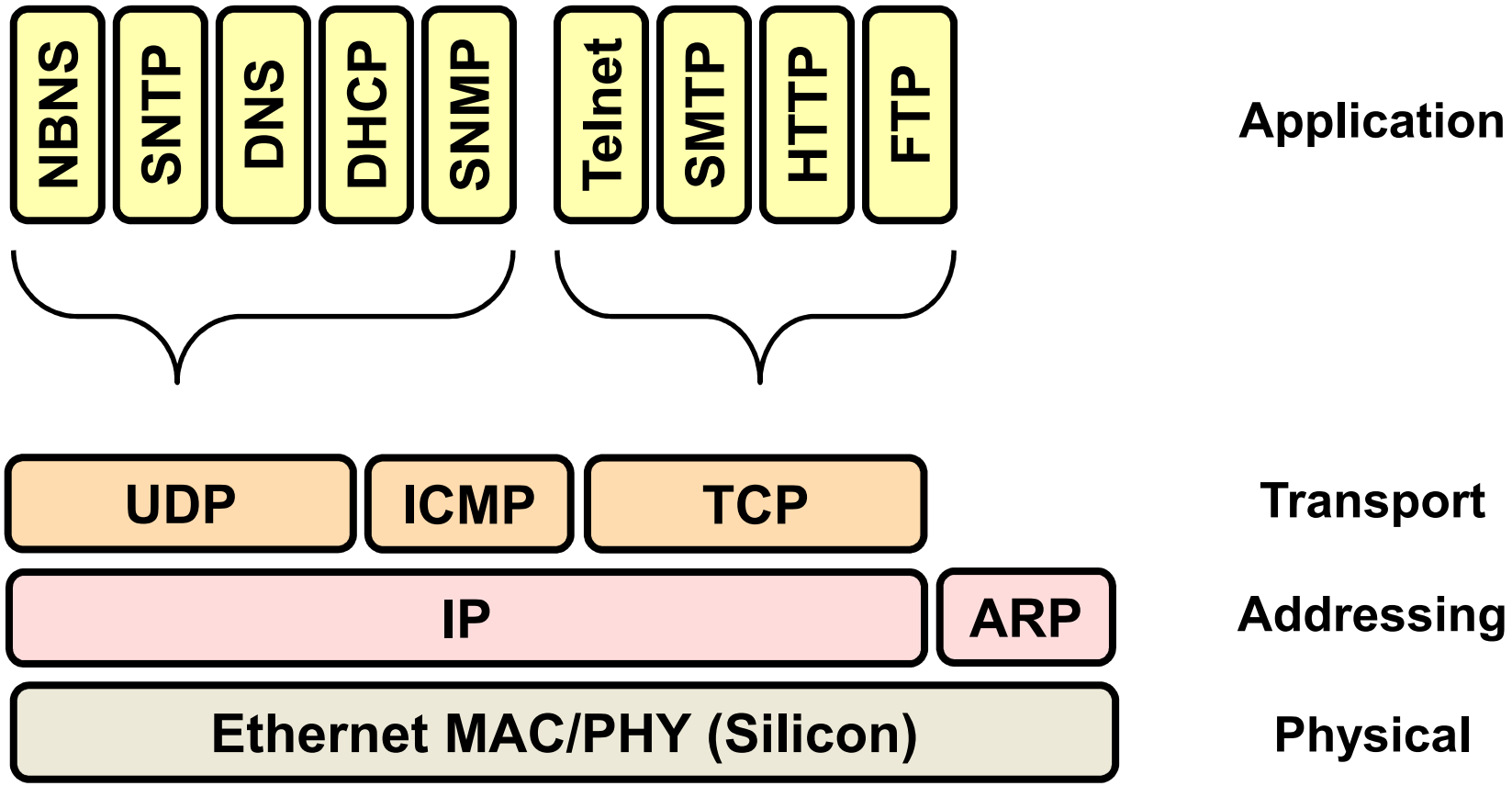
TCP/IP Stack



- C source code provided
  - No-fee license agreement
  - Use Microchip PIC® MCU or dsPIC® DSC
  - Only driver for ENC28J60 can be ported
  - Download: www.microchip.com/tcpip
- PIC18, PIC24, dsPIC DSC, PIC32
- RTOS Independent & Modular
- Supports multiple connections
- Example projects
- Standard Microchip technical support

MICROCHIP
Regional Training Centers

What's Included?



MICROCHIP
Regional Training Centers

What's Included?

HTTP

Web Server
Serve web pages and process web form input

SMTP

E-mail Client
Send e-mail or SMS messages

Telnet

Command-Line Interface Server
Allow simple text-based monitoring and control

SNMP

Simple Network Management Protocol Server
Aggregate enterprise monitor/control capabilities

TCP/UDP

Generic Data Transports
Retrieve data from servers, or serve data to clients

SSL

Secure Socket Layer
Encrypt communications over untrusted networks

Bootload

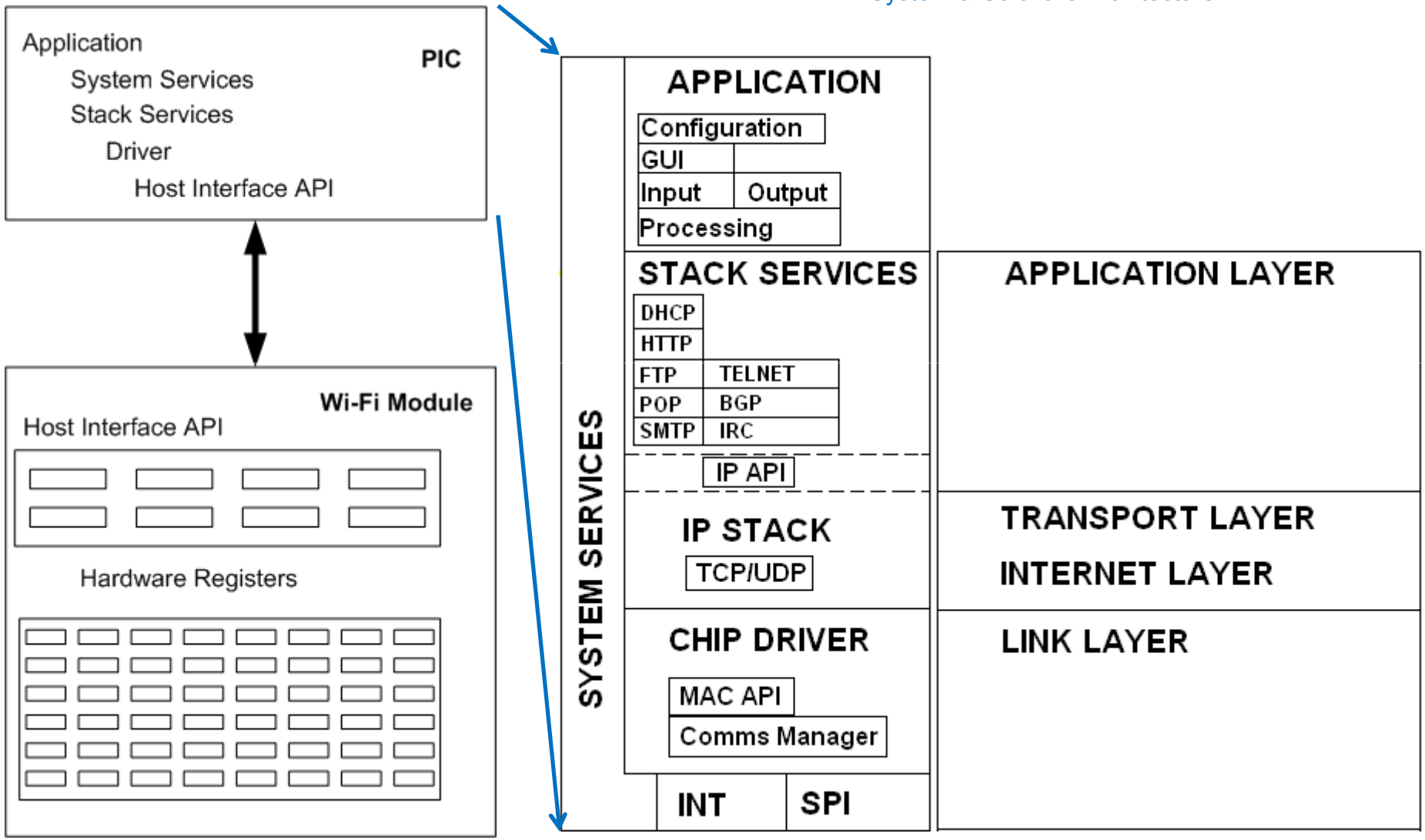
TFTP Bootloader
Update embedded firmware remotely

DDNS

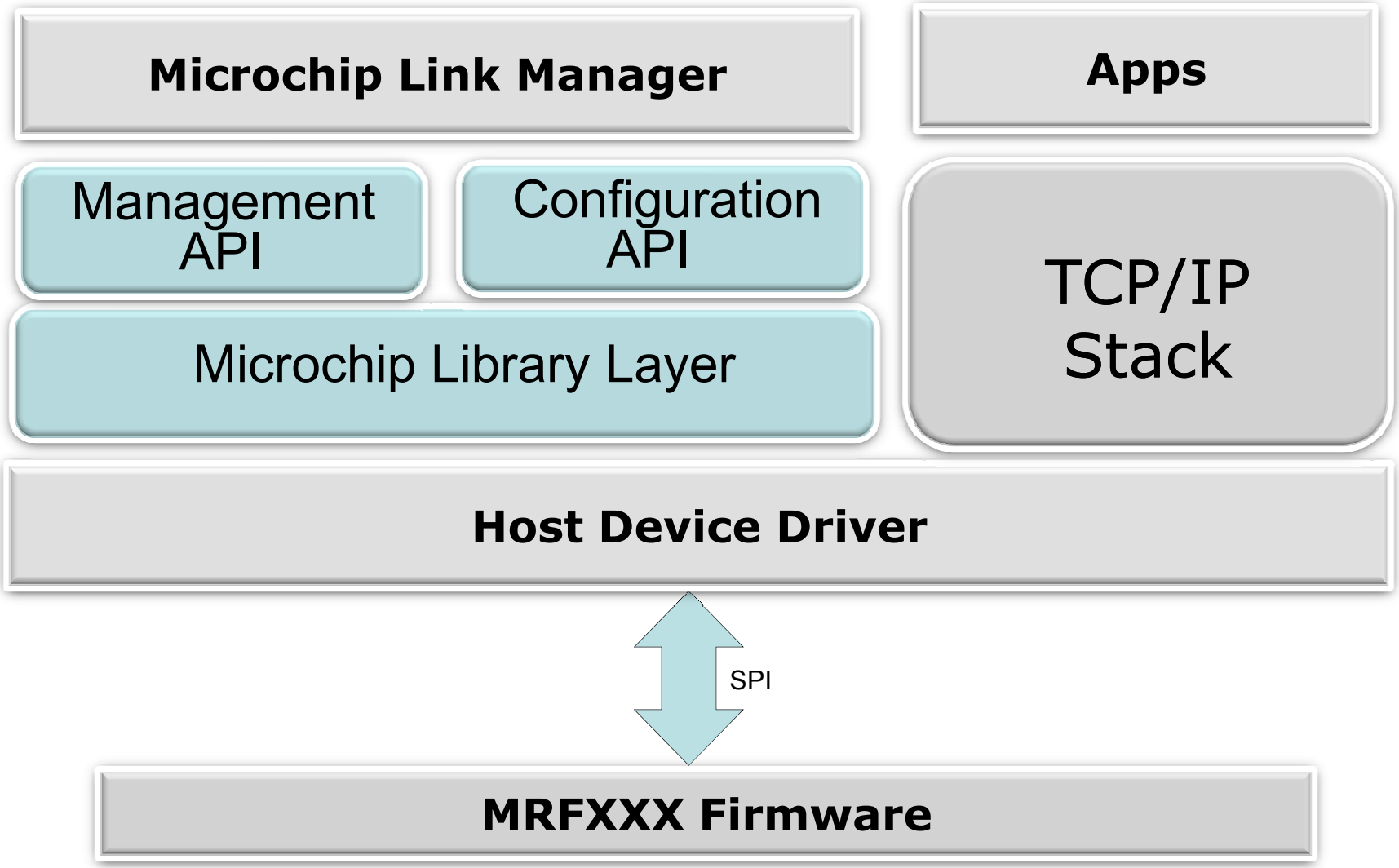
Dynamic Domain Name Service Client
Associate host names with dynamic addresses

Microchip 802.11 Solution

System & Software Architecture

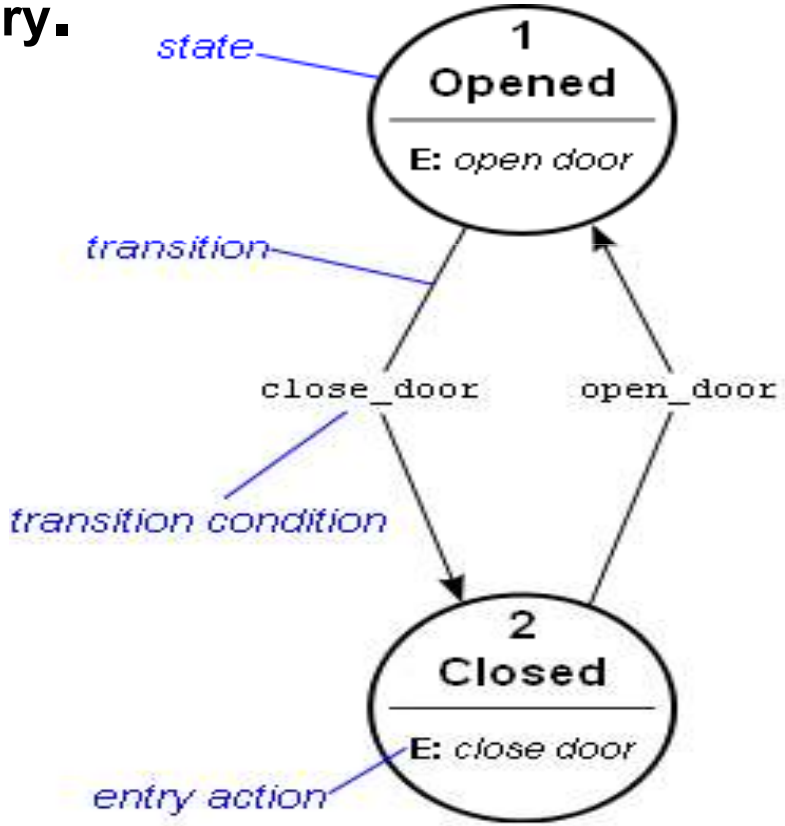


System Block Diagram

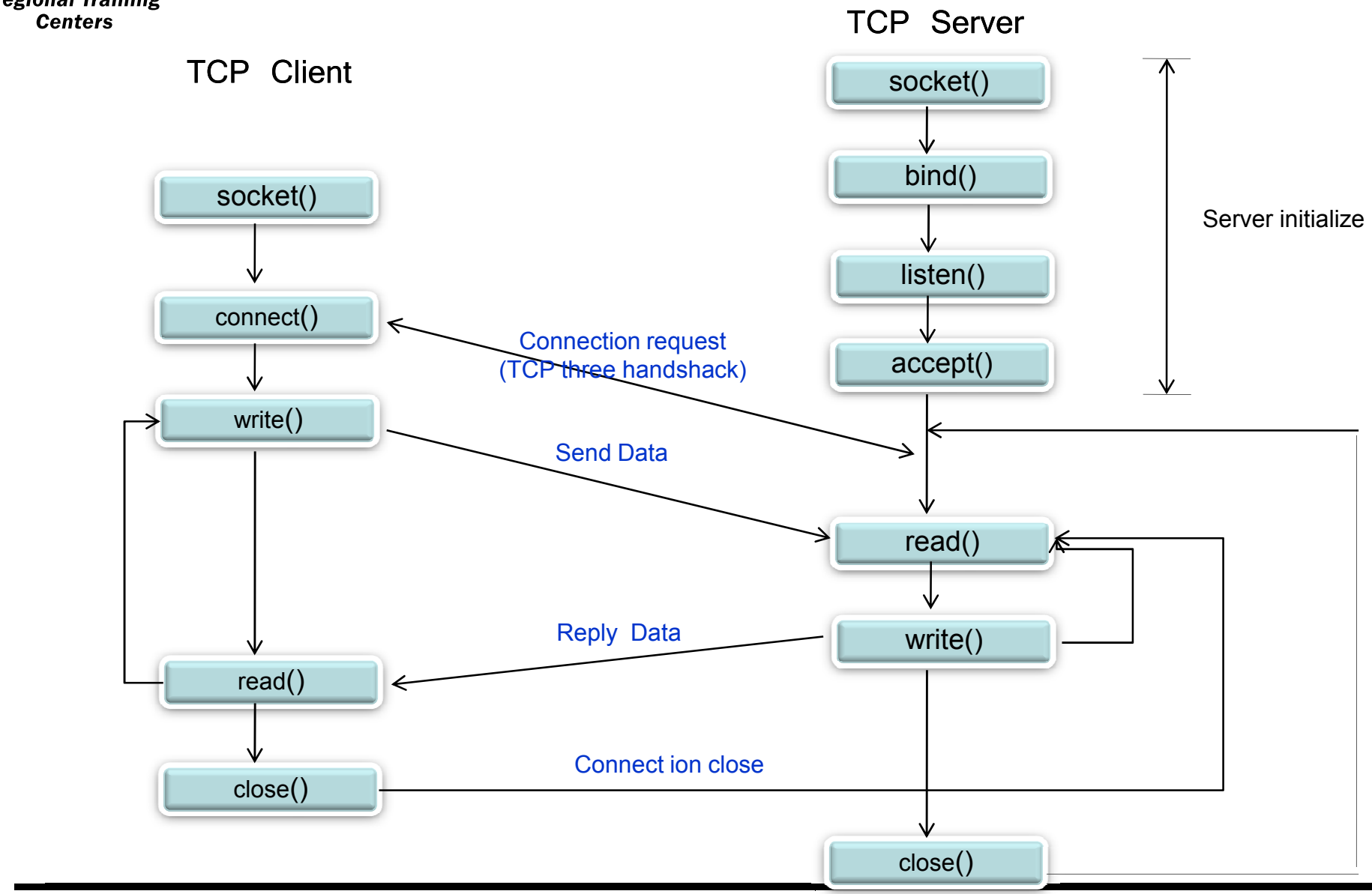


FSM(Finite-state machine)

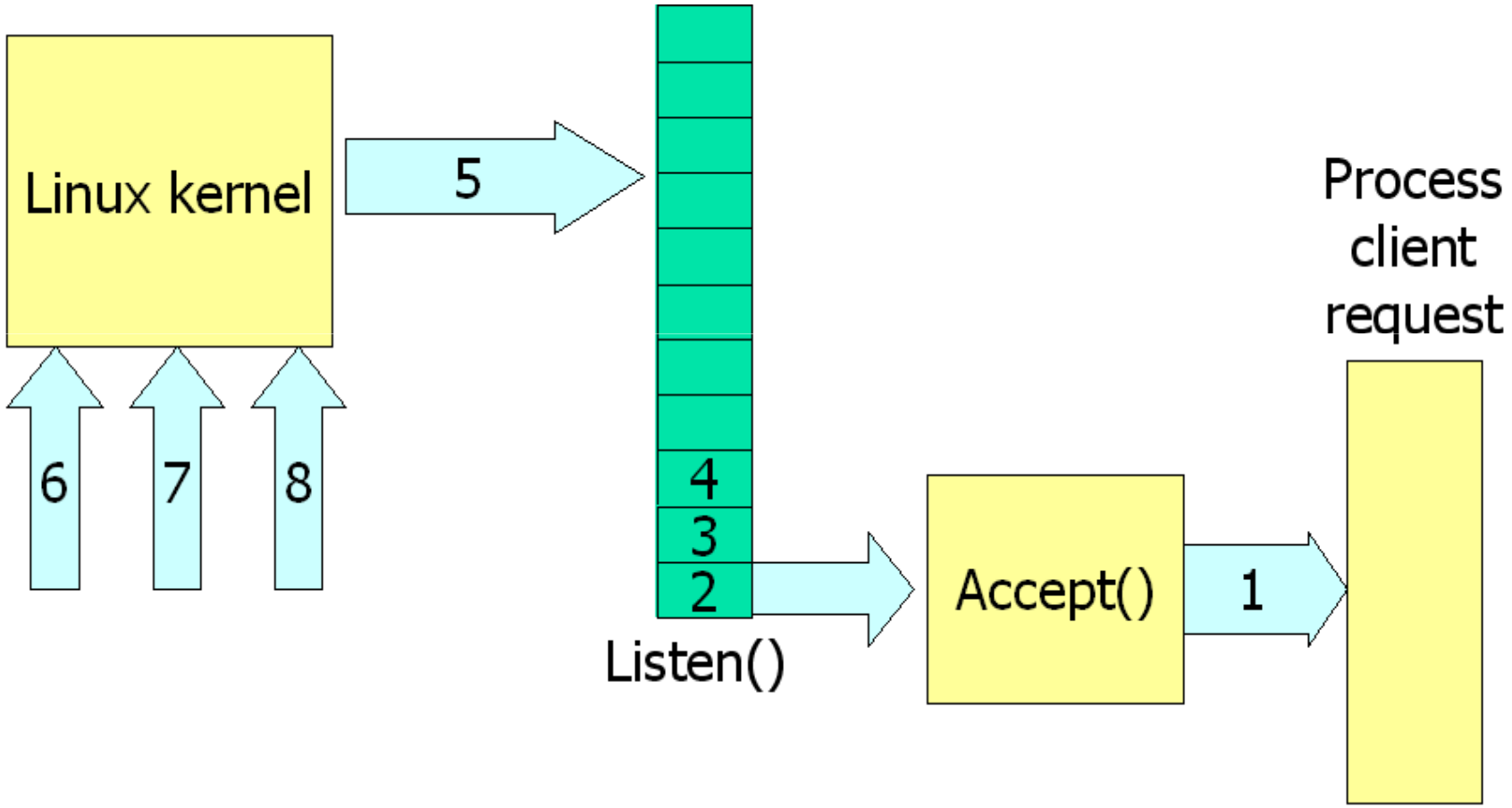
- **FSM is a model of behavior composed of a finite number of states, transitions between those states, and actions. A finite state machine is an abstract model of a machine with a Primitive internal memory.**



TCP Client-Server



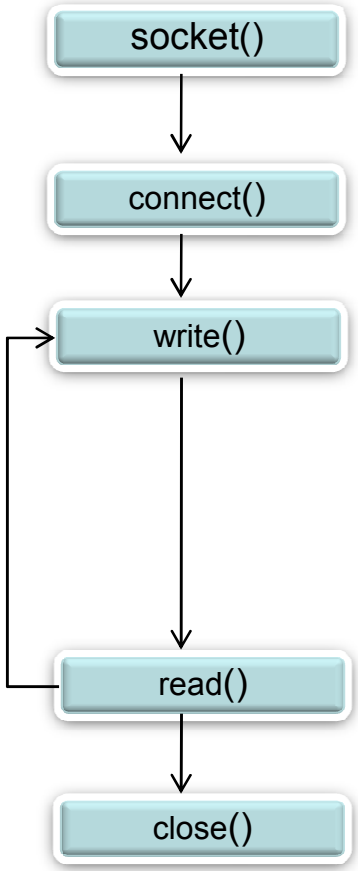
Linux TCP/IP



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);
        // get the server address
        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;
}
```

TCP Client



MICROCHIP
Regional Training Centers

Microchip TCP/IP Stack

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();
        ConsoleProcess();
        PingAppCall();
        BarCodeReaderDemo();
        ProcessIO();
     }
}
```

MICROCHIP
Regional Training Centers

MicroChip 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 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 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);
        //Timeout=2000;
        BARCodeClientState = BARDCode_OPERATION;
        break;
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;
```

MicroChip source code(4)

```
    case BARDCode_CLOSE:
        closesocket(bsdClientSocket);
        BARCodeClientState = BARDCode_DONE;

    case BARDCode_DONE:
        ZGConsoleReleaseConsoleMsg();
        BARCodeClientState = BARDCode_START;
        break;

    default:
        return;
    }

    }
```

MICROCHIP

Regional Training Centers

WiFi Programming

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: Enable PING function
- Lab 5: SendData to WebServer

MICROCHIP
Regional Training Centers

Hint

- Disable eeprom issue on InitAppconfig()
- Show error message

  WF_Config.h→WF_USE_DATA_TX_RX_FUNCTIONS
- WPA/WPA2→ SSID+Key→Generate key
- PING

#define STACK_USE_ICMP_CLIENT

- SendData

#define STACK_USE_BERKELEY_API

    {TCP_PURPOSE_BERKELEY_CLIENT, TCP_ETH_RAM, 125, 100}, //spencer

MICROCHIP
Regional Training Centers

Add Code on WF_Config.c

```
/*-------------------------------------*/
    case WF_EVENT_CONNECTION_FAILED:
    /*-------------------------------------*/
      /* eventInfo will contain value from tWFConnectionFailureReasons */
      #if defined(STACK_USE_UART)
      putrsUART("Event: Connection Failed  -- eventInfo = ");
      sprintf(buf, "%d \r\n", eventInfo);
      putsUART(buf);
                    switch (eventInfo) //add by spencer
                    {
                     case WF_JOIN_FAILURE:
                                  putrsUART("eventMessage: WF_JOIN_FAILURE \r\n");
                                  break;
                     case WF_AUTHENTICATION_FAILURE:
                                  putrsUART("eventMessage: WF_AUTHENTICATION_FAILURE \r\n");
                                  break;
                     case WF_ASSOCIATION_FAILURE:
                                  putrsUART("eventMessage: WF_ASSOCIATION_FAILURE \r\n");
                                  break;
                     case WF_WEP_HANDSHAKE_FAILURE:
                                  putrsUART("eventMessage: WF_WEP_HANDSHAKE_FAILURE \r\n");
                                  break;
                     case WF_PSK_CALCULATION_FAILURE:
                                  putrsUART("eventMessage: WF_PSK_CALCULATION_FAILURE \r\n");
                                  break;
                     case WF_PSK_HANDSHAKE_FAILURE:
                                  putrsUART("eventMessage: WF_PSK_HANDSHAKE_FAILURE \r\n");
                                  break;
                     case WF_ADHOC_JOIN_FAILURE:
                                  putrsUART("eventMessage: WF_ADHOC_JOIN_FAILURE \r\n");
                                  break;
                     case WF_SECURITY_MISMATCH_FAILURE:
                                  putrsUART("eventMessage: WF_SECURITY_MISMATCH_FAILURE \r\n");
                                  break;
                     case WF_NO_SUITABLE_AP_FOUND_FAILURE:
                                  putrsUART("eventMessage: WF_NO_SUITABLE_AP_FOUND_FAILURE \r\n");
                        break;
                     case WF_RETRY_FOREVER_NOT_SUPPORTED_FAILURE:
                                  putrsUART("eventMessage: WF_RETRY_FOREVER_NOT_SUPPORTED_FAILURE \r\n");
                                  break;
                    }
      #endif
      break;
```

MICROCHIP

Regional Training Centers

Thank You