



# **MICROCHIP**

---

***Regional Training Centers***

**Designing a WiFi -Based Monitor  
and Control Device**

**COM-WiFi**

# 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



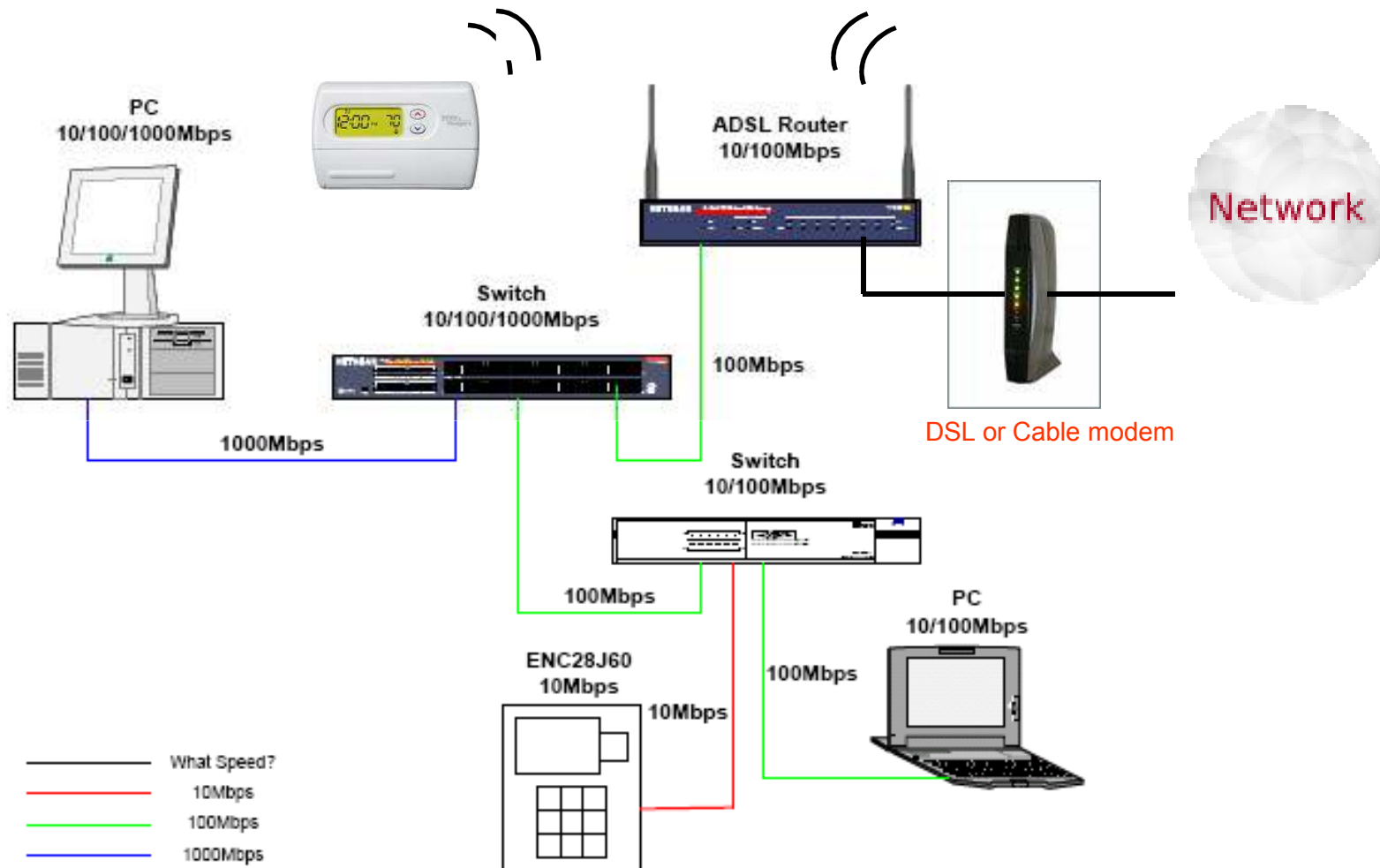
# **MICROCHIP**

---

***Regional Training Centers***

**Embedded WiFi**

# What is Wi-Fi®?



# What is Wi-Fi®?

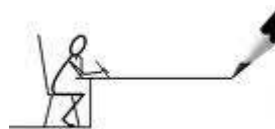
- **Ethernet is the most widely deployed datacom network in the world**
- **Wi-Fi is wireless Ethernet**
  - **Adds mobile internet connectivity**
  - **Removes the wire, but retains the LAN, WAN, WWW connection**

# 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**

# 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

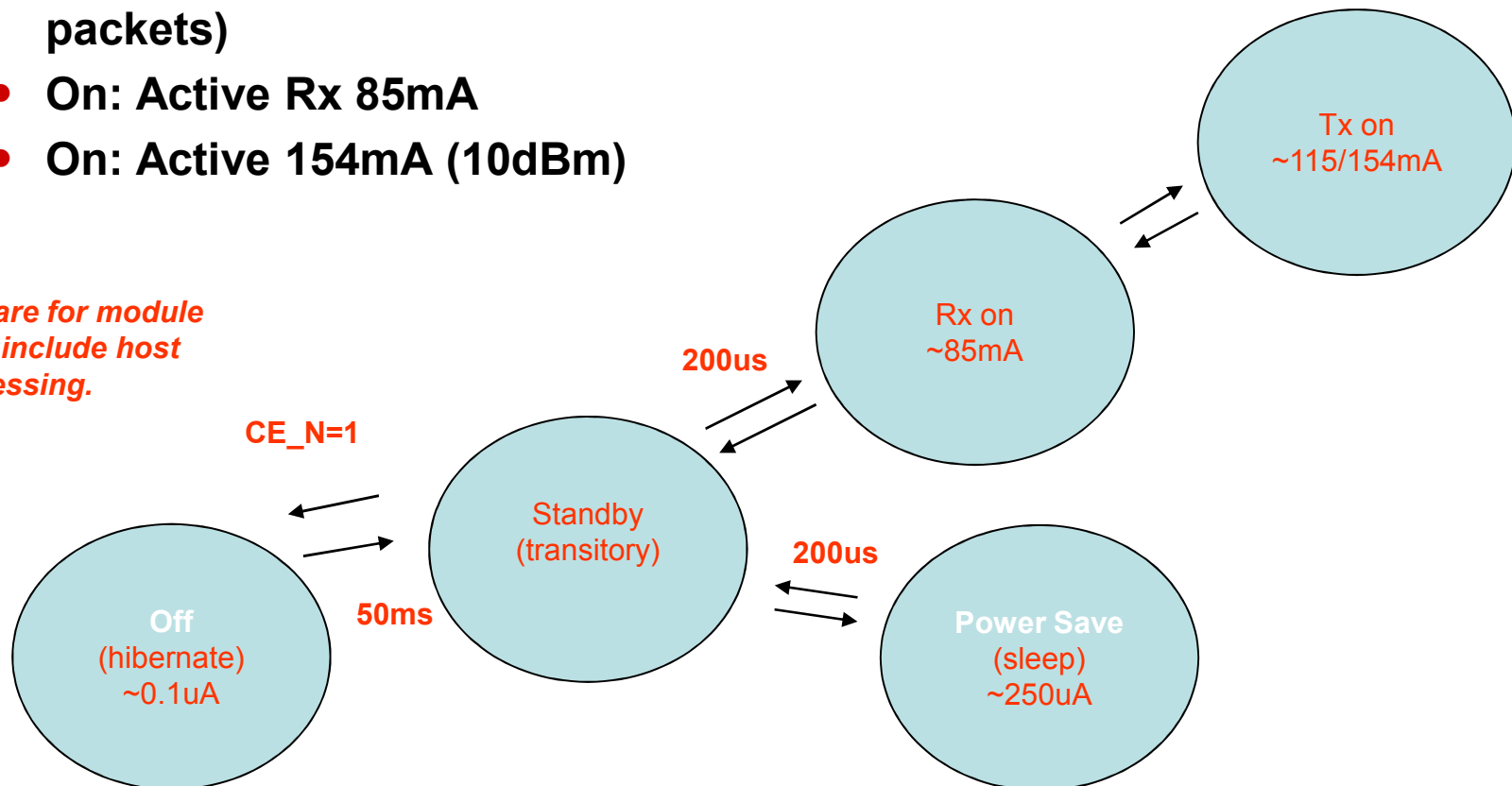
- **Wi-Fi® for low resource embedded applications**
- **Highly integrated, certified module solution**
  - Includes MAC, baseband, RF and PA
  - Easy integration into 8/16/32-bit systems
  - Standards-based, 1 & 2Mb/s, 802.11b
  - Full support of ad-hoc and AP modes
  - FCC, IC certified and ETSI tested
  - 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.*



# Product Fit

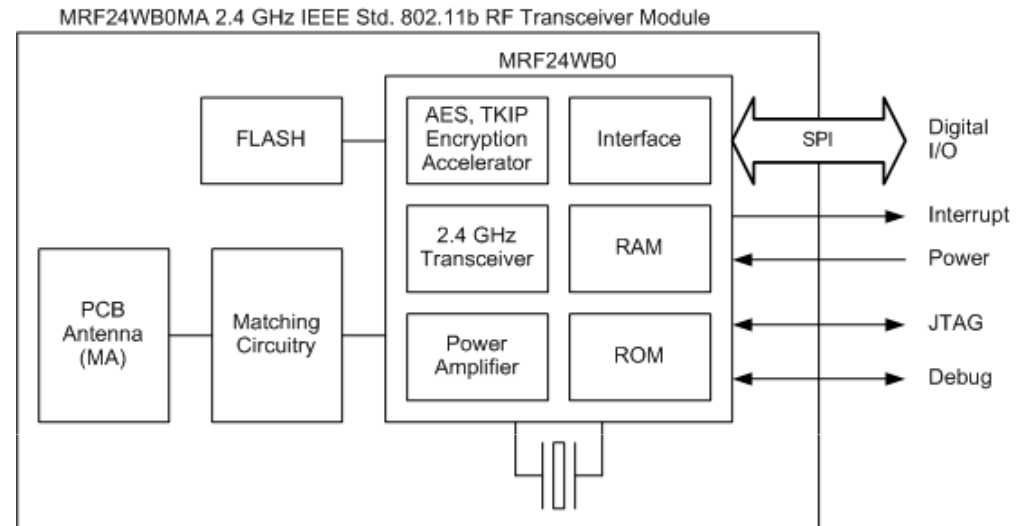
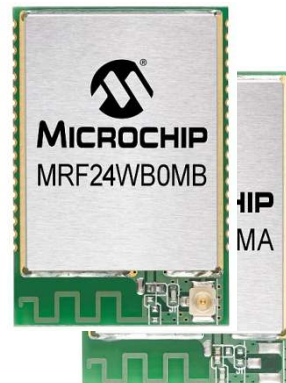
- **Types of products that are ideal**
  - ♦ **Small data quantity to transfer**
  - ♦ **Non-continuous transfer**
  - ♦ **Data is not high bandwidth streaming**



# Microchip Wi-Fi® Modules



MRF24WB0MA/B



AC164136-4

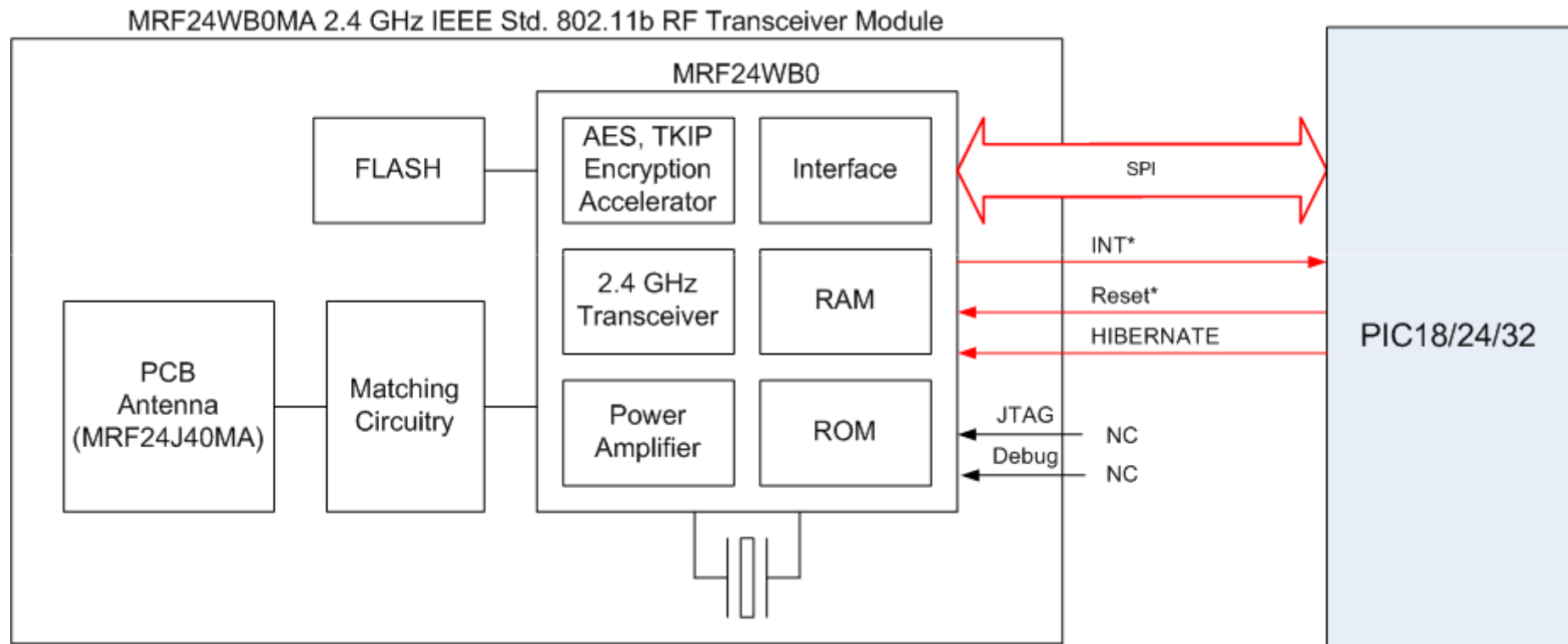


PICtail™ for development

- PICtail connector
- PICtail Plus connector

# Microchip Wi-Fi® Modules

Connection to PIC® MCU



# 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



# Stack Performance

Microcontroller	MIPS	Network Controller	Interface	LAN Transmit Throughput (Kbytes/sec)				Internet Transmit Throughput at 100ms ping (estimated Kbytes/sec) <sup>(1)</sup>				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 <sup>(2)</sup>	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 <sup>(3)</sup>	SPI, 10 MHz	28	75	80	115	2	20	80	115	C18	Debug
PIC24FJ128GA010	16	ENC624J600 <sup>(3)</sup>	SPI, 8 MHz	56	161	175	368	2	20	80	368	C30	s (min size)
dsPIC33FJ256GP710 <sup>(2)</sup>	40	ENC624J600 <sup>(3)</sup>	SPI, 8 MHz	91	276	310	559	2	20	80	559	C30	s (min size)
PIC32MX360F512L	80	ENC624J600 <sup>(3)</sup>	SPI, 13.33 MHz	139	356	412	687	2	20	80	687	C32	s (min size)
PIC32MX795F512L	80	ENC624J600 <sup>(3)</sup>	SPI, 13.33 MHz	152	396	457	801	2	20	80	801	C32	s (min size)
PIC24FJ128GA010	16	ENC624J600 <sup>(3)</sup>	PSP Mode 5, PMP	101	300	339	1043	2	20	80	1043	C30	s (min size)
dsPIC33FJ256GP710 <sup>(2)</sup>	40	ENC624J600 <sup>(3)</sup>	PSP Mode 5, Bitbang	196	659	813	2063	2	20	80	2063	C30	s (min size)
PIC32MX360F512L	80	ENC624J600 <sup>(3)</sup>	PSP Mode 5, PMP	203	786	996	2115	2	20	80	2115	C32	s (min size)
PIC32MX795F512L <sup>(3)</sup>	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 <sup>(2)</sup>	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**

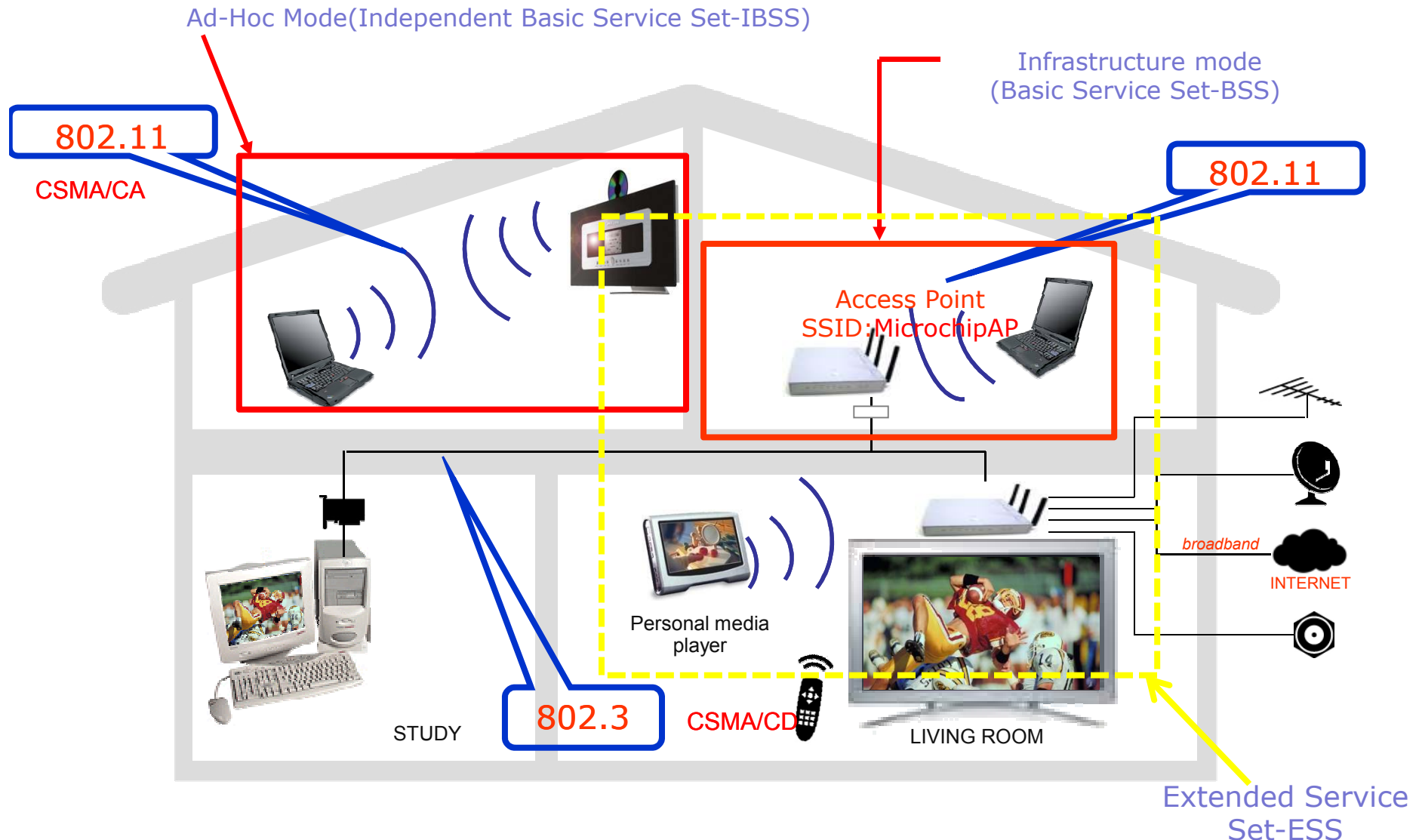
---

***Regional Training Centers***

Overview of 802.11 Networks



# Overview of 802.11 Network



# 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 transmission

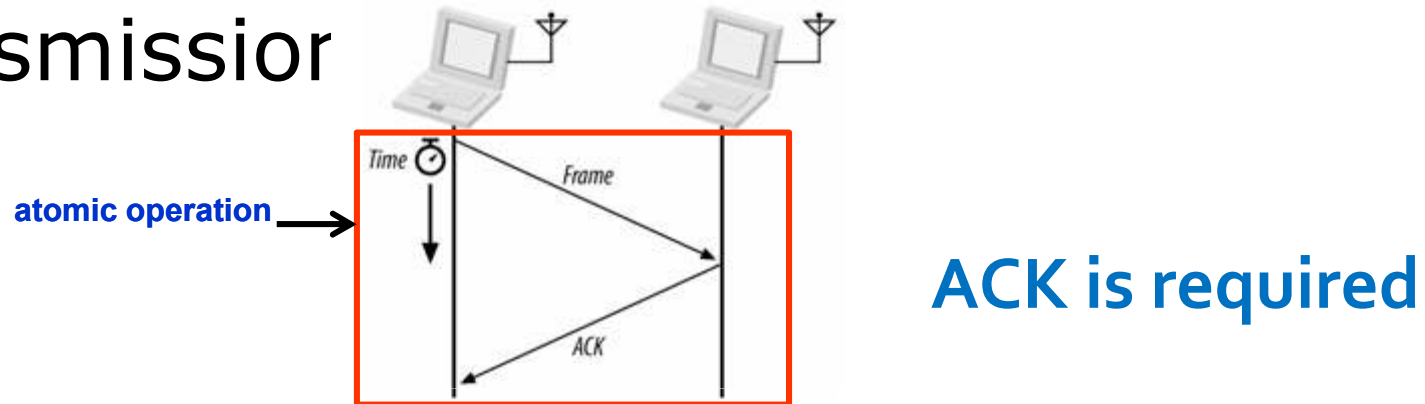


Figure- Positive acknowledgment of data transmissions

- The Hidden Node Problem

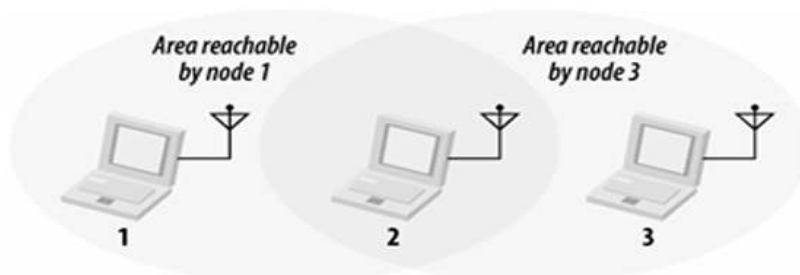


Figure- Nodes 1 and 3 are "hidden"

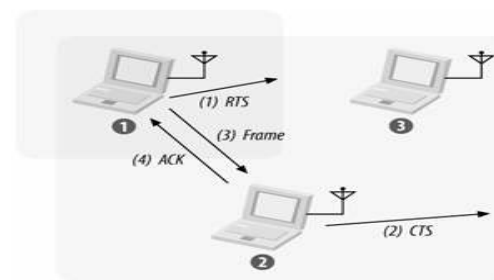
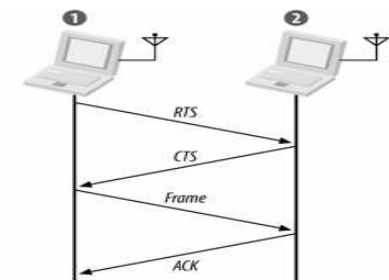
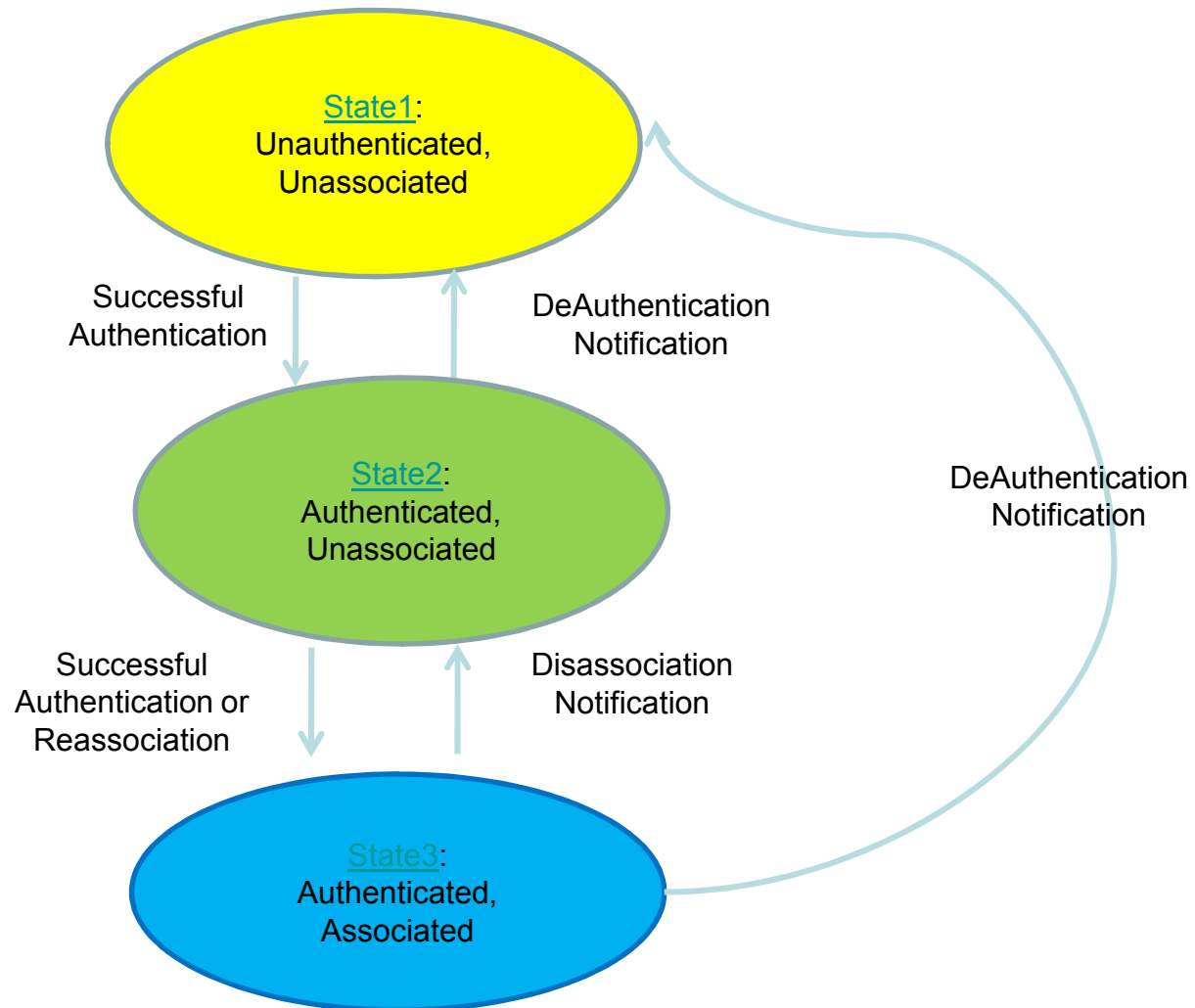


Figure- RTS/CTS clearing



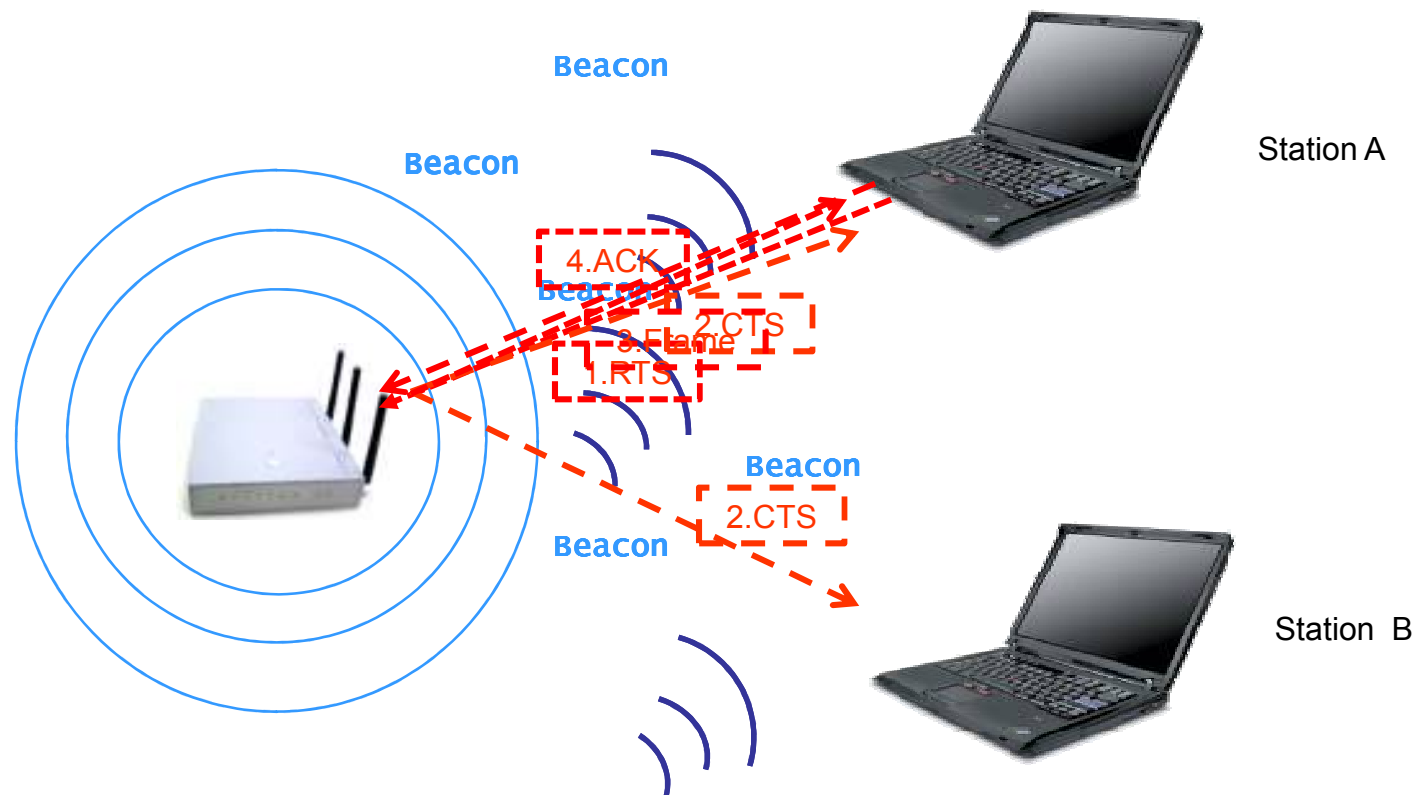
# Frame Transmission & Association States

- Station connect to AP(Access Point)

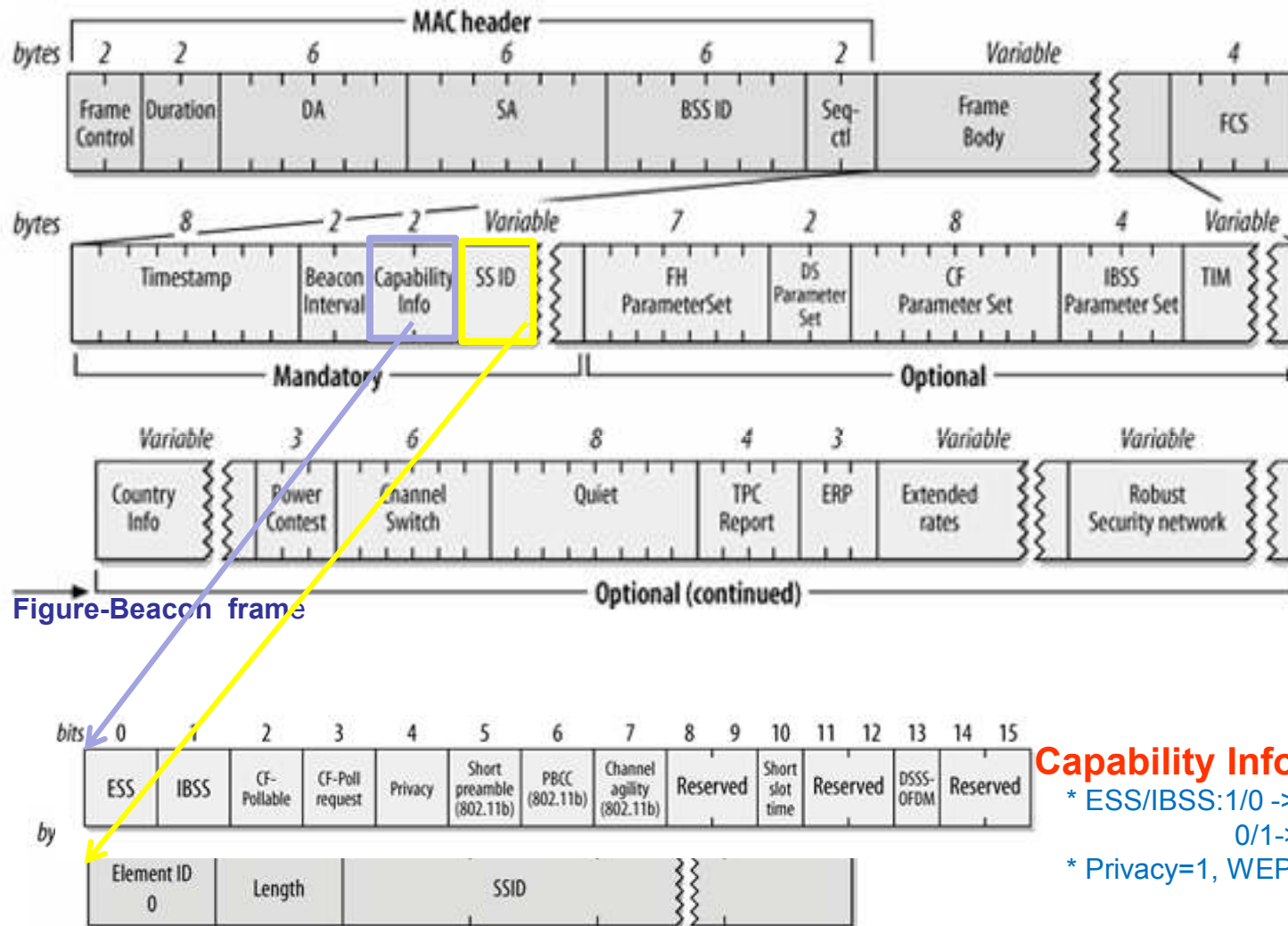


# Frame Transmission & Association States

- State 1( Unauthenticated, Unassociated)



# Beacon-Management Frame



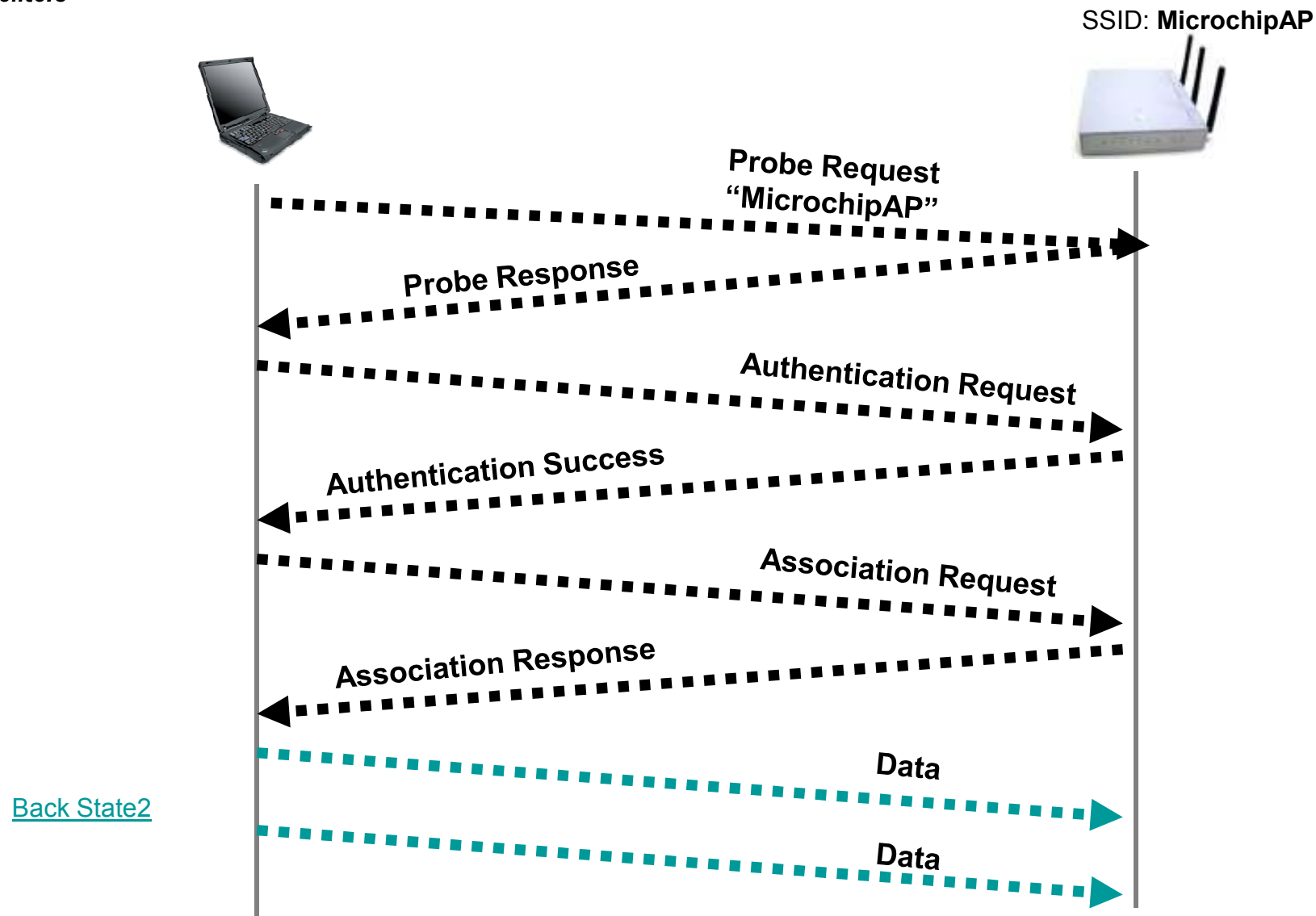
## Capability Information field

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

## Figure-Service Set Identity information element

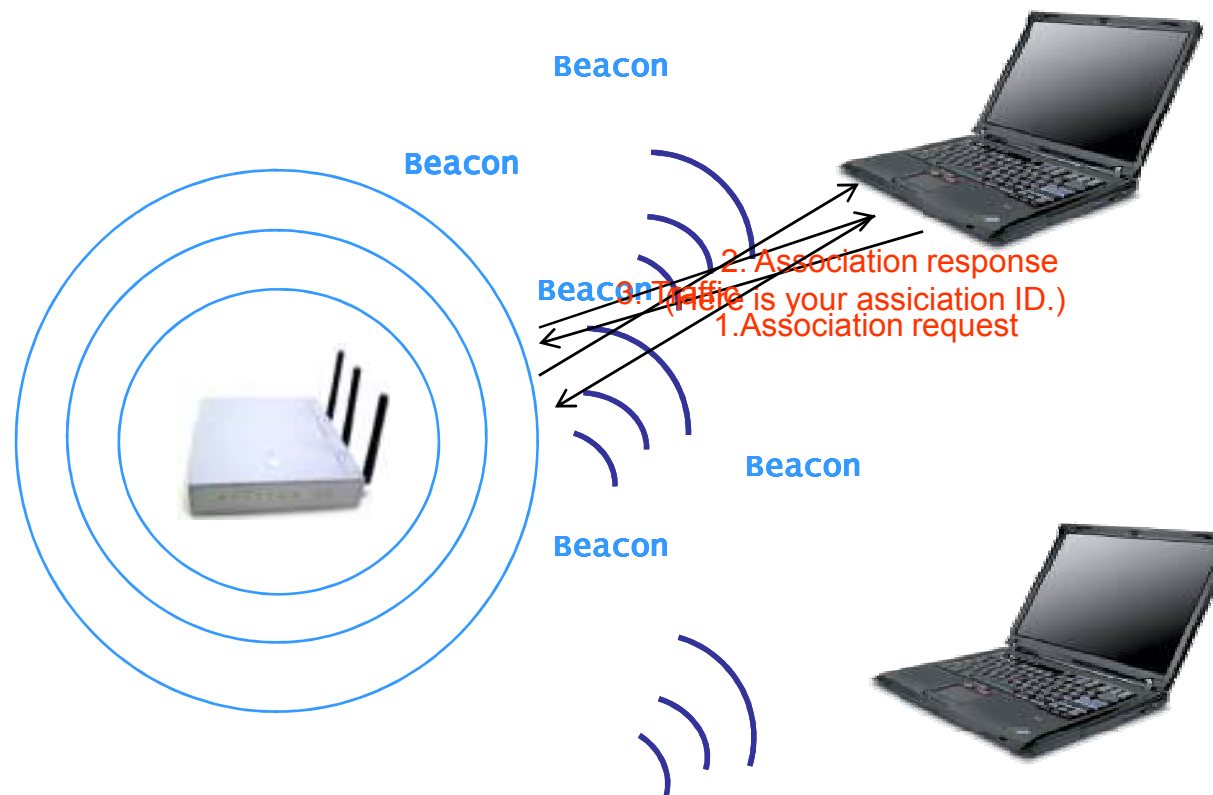
\*Most products require that the string be a garden variety, null-terminated ASCII string. The length of SSID ranges 0 ~ 32 bytes.

# Probe Request/Response (Management Frame)



# Frame Transmission & Association States

- State 2( Authenticated, Unassociated)





# Disassociation and de-authentication (Management Frame)

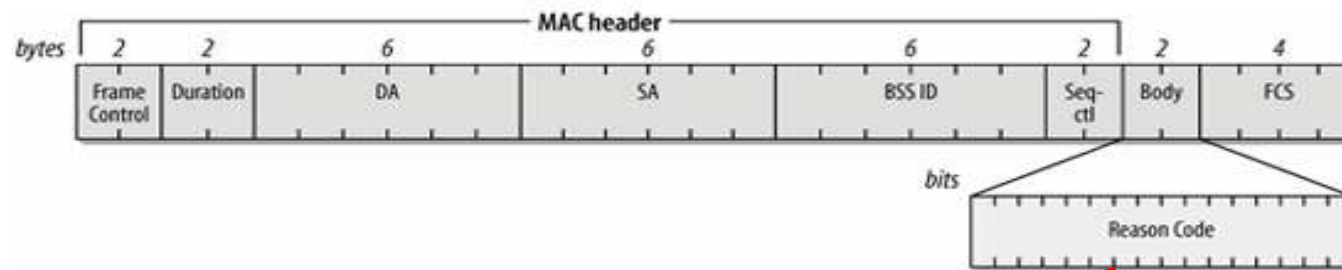


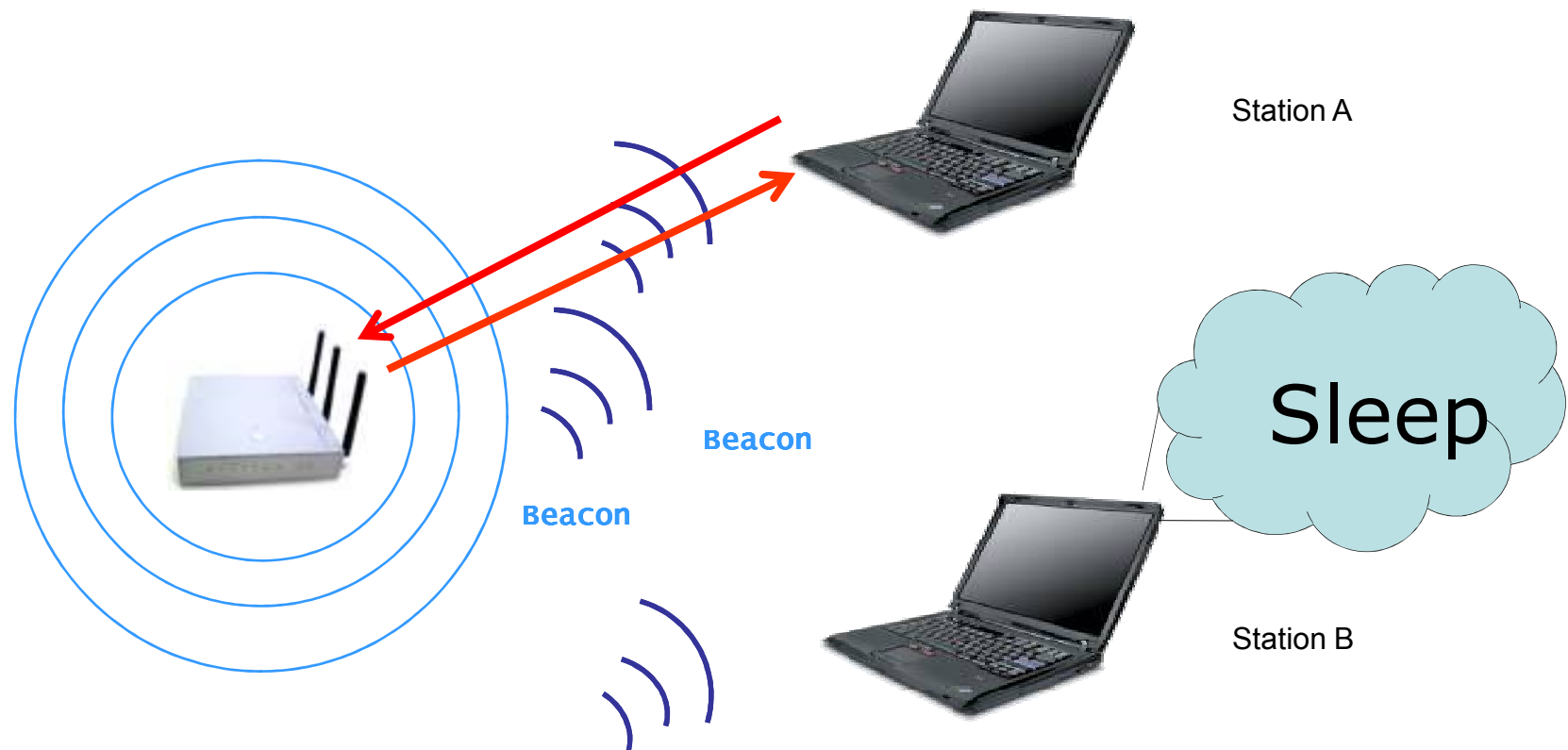
Figure-Disassociation and Deauthentication frames

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

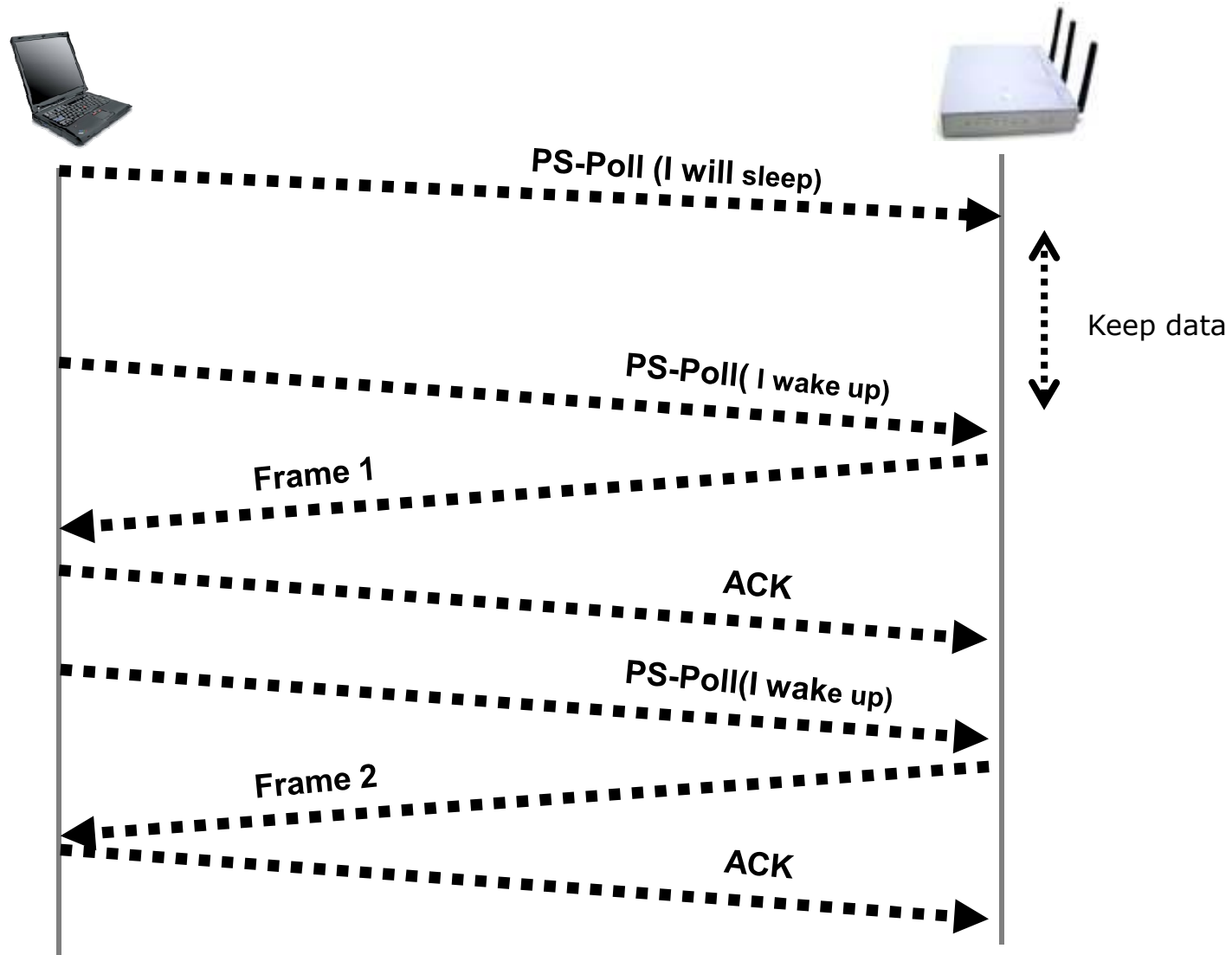
[Back State3](#)

# 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

**LINKSYS**  
A Division of Cisco Systems, Inc. Firmware Version: v9.00.0

**Wireless-G Broadband Router WRT54G**

**Wireless**

Setup | **Wireless** | Security | Access Restrictions | Applications & Gaming | Administration | Status

Basic Wireless Settings | **Wireless Security** | Wireless MAC Filter | Advanced Wireless Settings

**Wireless Security**

Security Mode: **WEP**

Default Transmit Key: 1 2 3 4

WEP Encryption: 64 bits 10 hex digits | 64 bits 10 hex digits | 128 bits 26 hex digits

Passphrase:

Key 1:

Key 2:

Key 3:

Key 4:

**Security Mode:** You may choose from **Disable, WPA Personal, WPA Enterprise, WPA2 Personal, WPA2 Enterprise, RADIUS, WEP**. All devices on your network must use the same security mode in order to communicate. [More...](#)

**CISCO SYSTEMS**

**LINKSYS**  
A Division of Cisco Systems, Inc. Firmware Version: v9.00.0

**Wireless-G Broadband Router WRT54G**

**Wireless**

Setup | **Wireless** | Security | Access Restrictions | Applications & Gaming | Administration | Status

Basic Wireless Settings | **Wireless Security** | Wireless MAC Filter | Advanced Wireless Settings

**Wireless Security**

Security Mode: **WPA Personal**

WPA Algorithms: **TKIP**

WPA Shared Key:

Group Key Renewal: 3600 seconds

**Security Mode:** You may choose from **Disable, WPA Personal, WPA Enterprise, WPA2 Personal, WPA2 Enterprise, RADIUS, WEP**. All devices on your network must use the same security mode in order to communicate. [More...](#)

**CISCO SYSTEMS**

**LINKSYS**  
A Division of Cisco Systems, Inc. Firmware Version: v9.00.0

**Wireless-G Broadband Router WRT54G**

**Wireless**

Setup | **Wireless** | Security | Access Restrictions | Applications & Gaming | Administration | Status

Basic Wireless Settings | **Wireless Security** | Wireless MAC Filter | Advanced Wireless Settings

**Wireless Security**

Security Mode: **WPA2 Personal**

WPA Algorithms: **AES**

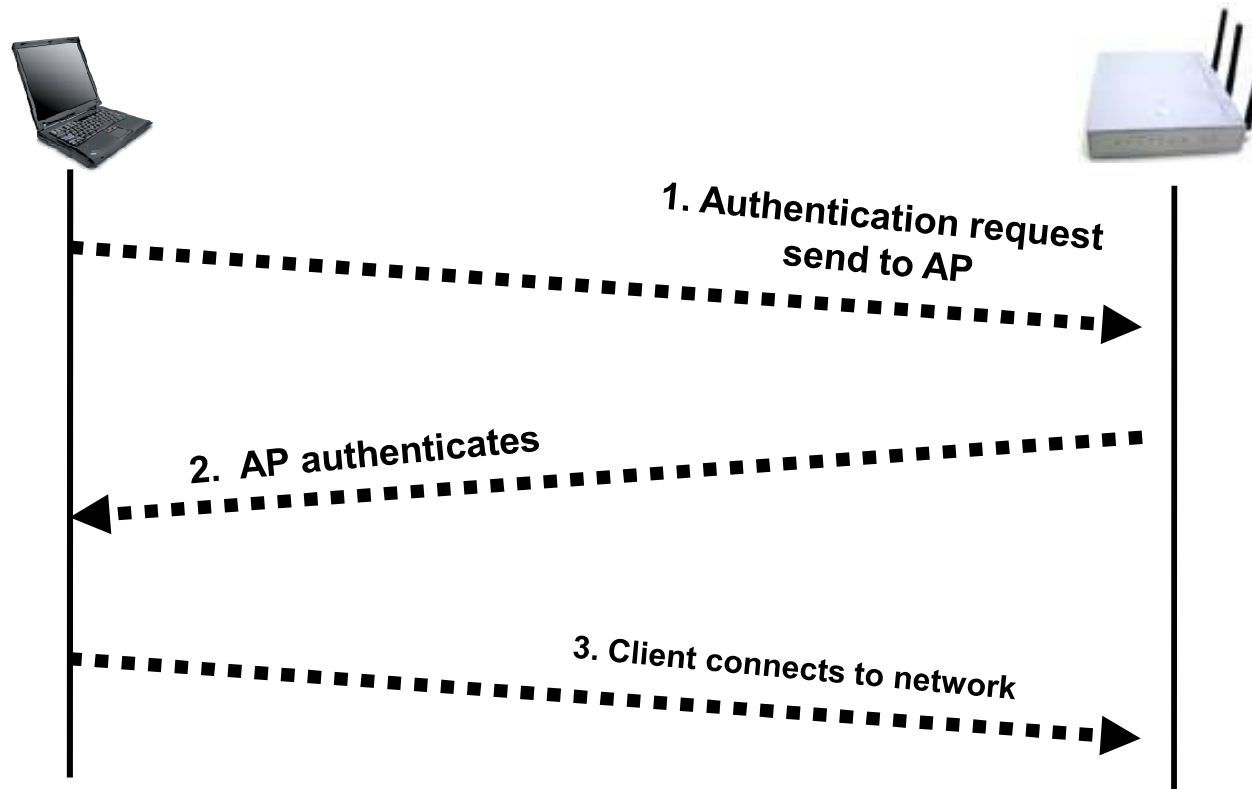
WPA Shared Key: **AES**

Group Key Renewal: 3600 seconds

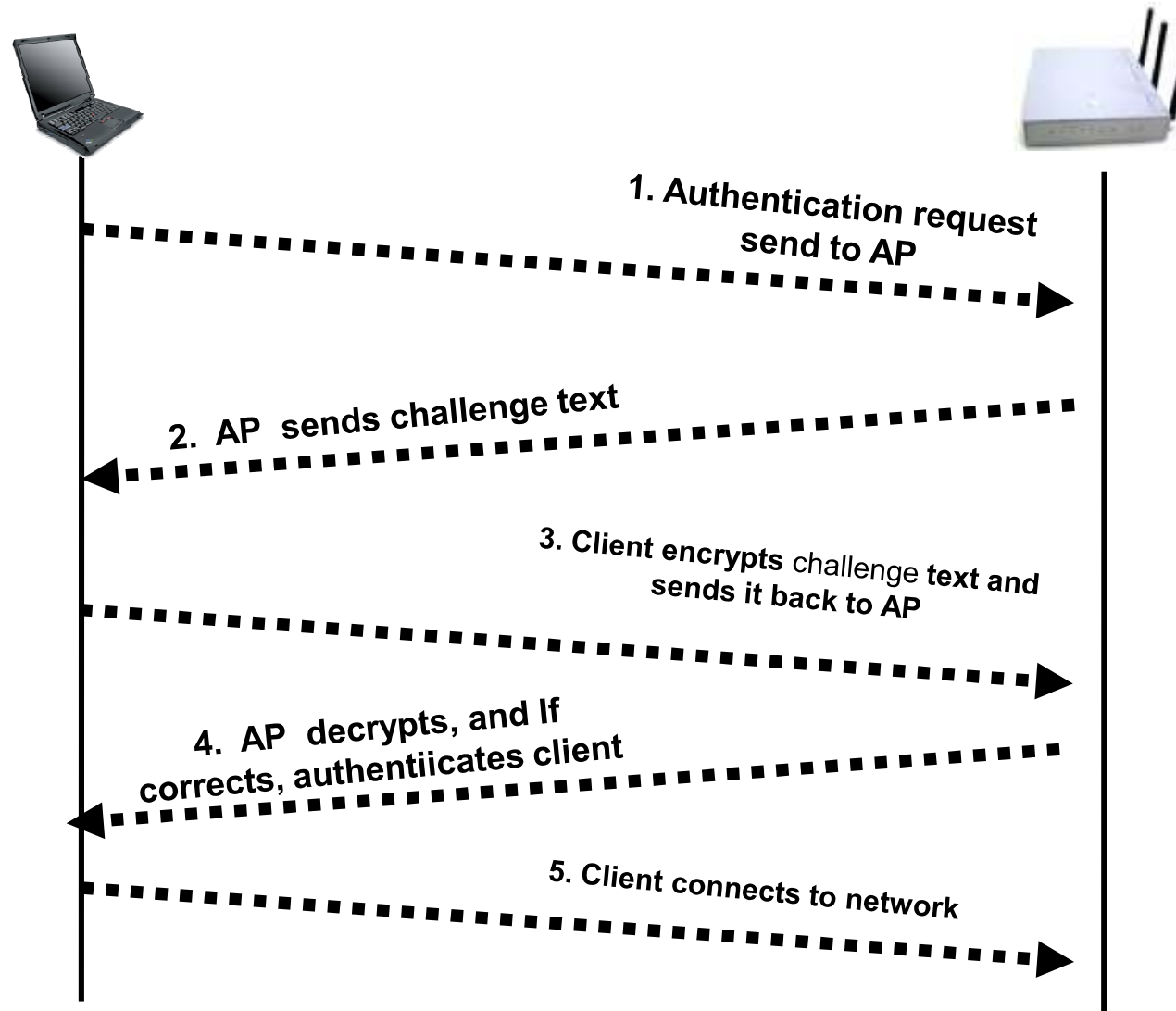
**Security Mode:** You may choose from **Disable, WPA Personal, WPA Enterprise, WPA2 Personal, WPA2 Enterprise, RADIUS, WEP**. All devices on your network must use the same security mode in order to communicate. [More...](#)

**CISCO SYSTEMS**

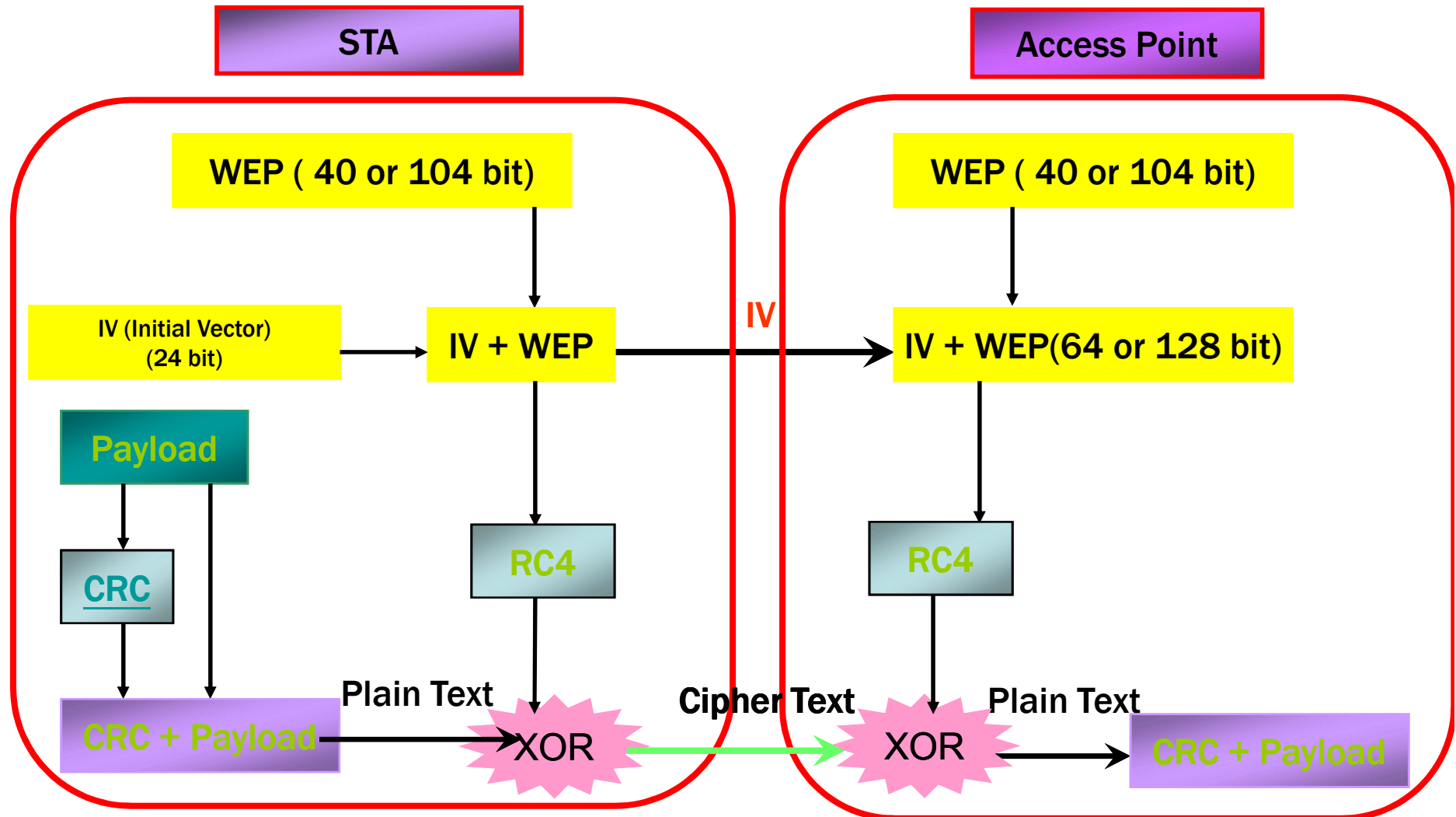
# Open System Authentication



# WEP Shared Key Authentication



# WEP-Encryption/Decryption



RC4([Rivest Cipher](#)), is the most widely-used software stream cipher .

IV: 24 bit initialization vector,  $2^{24} = 16777216$



# WEP Weaknesses

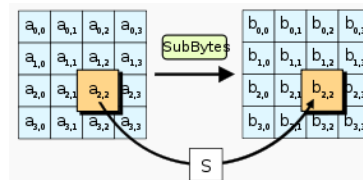
- The IV is too small and in cleartext.
- The IV is static (0 → 16777216)
- The IV makes the key stream vulnerable.
- The IV is a part of the RC4 encryption key
- WEP provides no cryptographic integrity protection

# 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/](http://www.wi-fi.org/knowledge_center/webcast-wpa-061103/)

AES(Advanced Encryption Standard)

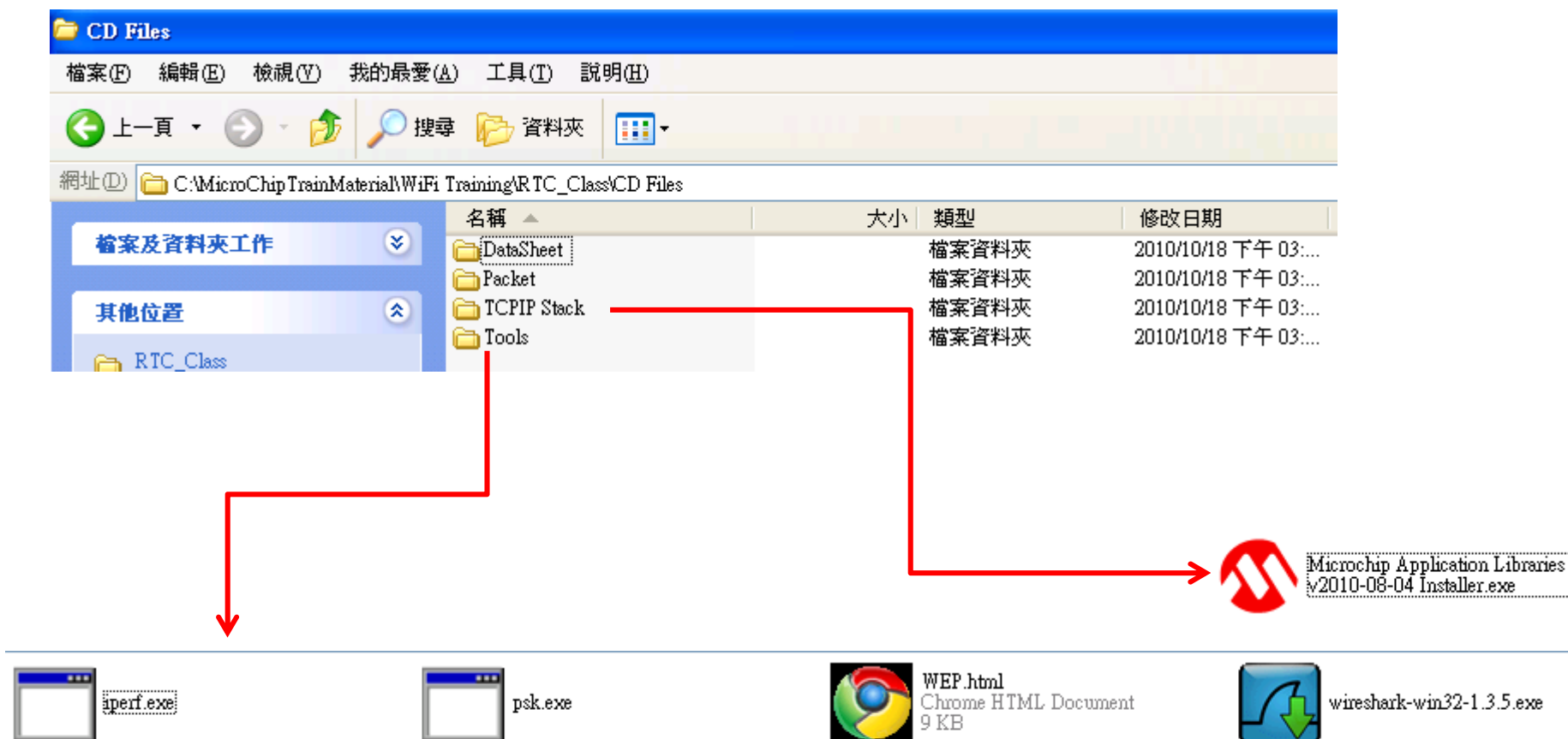


Michael: Message Integrity Code(MIC)

CCM ( Counter-Mode/CBC-MAC)

# Wireless Experiment

# CD Folder



CD Files

檔案(F) 編輯(E) 檢視(V) 我的最愛(A) 工具(T) 說明(H)

← 上一頁 → 搜尋 資料夾

網址(D) C:\MicroChipTrainMaterial\WiFi Training\RTC\_Class\CD Files

名稱	大小	類型	修改日期
DataSheet		檔案資料夾	2010/10/18 下午 03:...
Packet		檔案資料夾	2010/10/18 下午 03:...
TCP/IP Stack		檔案資料夾	2010/10/18 下午 03:...
Tools		檔案資料夾	2010/10/18 下午 03:...

檔案及資料夾工作

其他位置

RTC\_Class

iperf.exe

psk.exe

WEp.html  
Chrome HTML Document  
9 KB

Microchip Application Libraries  
v2010-08-04 Installer.exe

wireshark-win32-1.3.5.exe



# 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)

1. DHCP Discover  
3. DHCP Offer (who use 192.168.2.100)

Packet	Source Physical	Dest. Physical	Size	Bar	Absolute Time	Protocol	Summary
97	00:13:F7:E9:01:44	FF:FF:FF:FF:FF:FF	802.11 Beacon		22:39:02.015901	802.11 Beacon	FC=.....,SN=3205,FW= 0,BI=...
98	00:13:F7:E9:01:42	FF:FF:FF:FF:FF:FF	IP BOOTP		22:39:02.073055	DHCP	R ACK

Packet: 98 [X] ?	
<b>Packet Info</b>	Packet Number=98 Flags=0x00000000 Status=0x00000000 Packet Length=368 Timestamp=22:39:02.073055000 02/26/2009 Data Rate=2 1 .0 Mbps Chan=1 2412
<b>802.11 MAC Header</b>	Version=0 Type=10 Data Subtype=0000 Data Only Duration=0 Microseconds Destination=Ethernet Broadcast BSSID=00:13:F7:E9:01:44 Source=00:13:F7:E9:01:42
<b>802.2:</b>	D=0xAA SNAP S=0xAA SNAP C=0x03 Unnumbered Information
<b>IP:</b>	S=192.168.2.1 D=IP Broadcast
<b>UDP:</b>	Src=bootps Dst=bootpc
<b>BOOTP:</b>	Operation=2 Boot Reply Hardware Address Type=1 Hardware Address Length=6bytes Hops=0 Transaction ID=2917012003 Seconds Since Boot Start=0 BootP 1=0 Boot
<b>DHCP - Dynamic Host Configuration Protocol</b>	
<b>DHCP Magic Cookie:</b>	0x63825363
<b>Message Type Option Code=53</b>	Message Type Option Length=1 Message Type=5 ACK
<b>Server Identifier Option Code=54</b>	Server Identifier Option Length=4 Address=192.168.2.1
<b>IP Address Lease Time Option Code=51</b>	IP Address Lease Time Option Length=4 Value=86400
<b>Subnet Mask Option Code=1</b>	Subnet Mask Option Length=4 Address=255.255.255.0
<b>Routers Option Code=3</b>	Routers Option Length=4 Address=192.168.2.1
<b>Domain Name Servers Option Code=6</b>	Domain Name Servers Option Length=4 Address=192.168.2.1
<b>DHCP Option End Option Code=255</b>	
<b>Data Area:</b>	(26 bytes)
<b>Extra bytes (Padding):</b>	(4 bytes)
<b>FCS:</b>	FCS=0x1CDF4421 Calculated



# **MICROCHIP**

---

***Regional Training Centers***

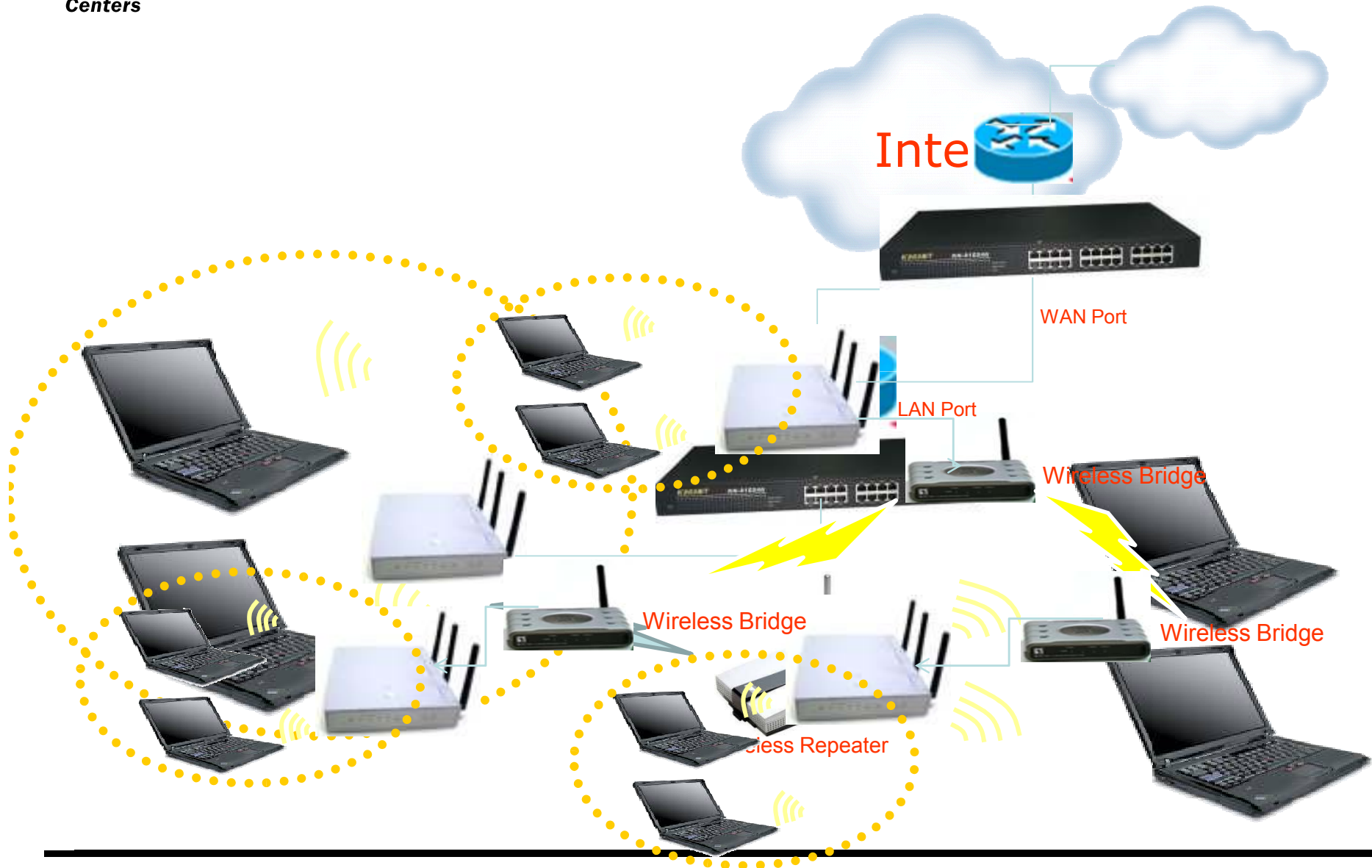
**Wireless Network Deployment &  
Wireless AP Feature**



# Wireless Network Deployment

- Root Mode
- Repeater Mode
- Bridge Mode

# Wireless Network Deployment



# Wireless Network Deployment

一般設定 WPS 無線橋接 無線存取控制 RADIUS設定 專業設定

### 無線網路 - 無線橋接

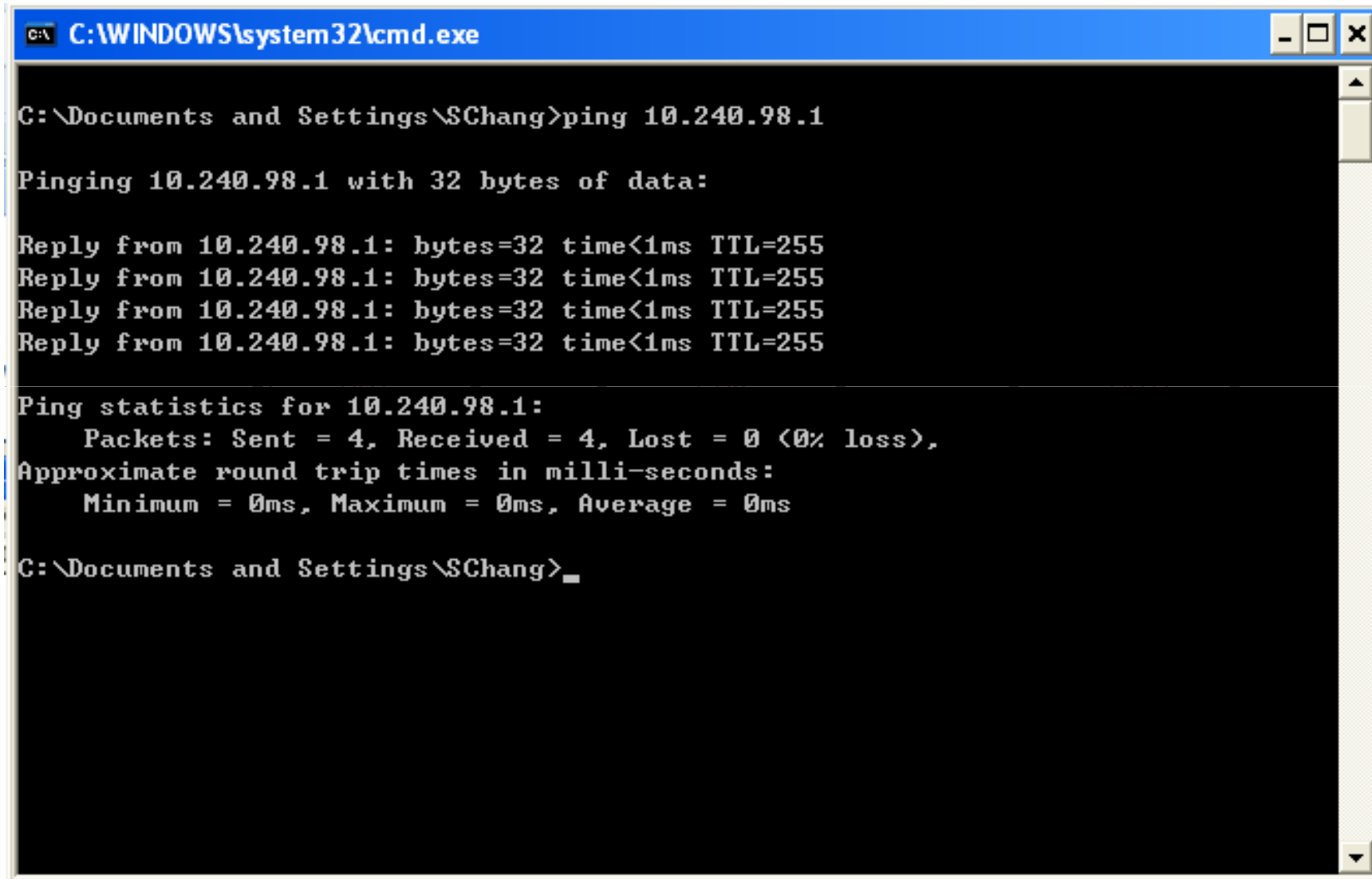
無線橋接（亦稱作「無線分散系統」或WDS）功能讓您可透過無線網路連接多個基地台。啓用此功能時，提醒您注意以下事項：

- 選擇「WDS Only」或「Hybrid」模式，並新增欲橋接的遠端基地台MAC位址。
- 為確保連接無誤，基地台請設定相同的頻道與[安全性加密](#)。
- 若欲橋接不同型號的基地台，本產品僅支援Open system WEP加密方式。

AP模式：	<span>WDS Only</span>
頻道：	<span>1</span>
連接清單中的基地台？	<input checked="" type="radio"/> Yes <input type="radio"/> No
遠端基地台清單	<input type="text"/> <span>新增</span>
	<div>*請輸入完整包含12個16進位制字母的MAC位址，0~9，A~F不包括"："。</div> <div><div></div><div>刪除</div></div>
<span>套用本頁面設定</span>	

# 802.11 Network Deployment

[Back](#)



```
C:\WINDOWS\system32\cmd.exe

C:\Documents and Settings\SChang>ping 10.240.98.1

Pinging 10.240.98.1 with 32 bytes of data:

Reply from 10.240.98.1: bytes=32 time<1ms TTL=255
Reply from 10.240.98.1: bytes=32 time<1ms TTL=255
Reply from 10.240.98.1: bytes=32 time<1ms TTL=255
Reply from 10.240.98.1: bytes=32 time<1ms TTL=255

Ping statistics for 10.240.98.1:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 0ms, Maximum = 0ms, Average = 0ms

C:\Documents and Settings\SChang>
```

# Wireless AP Features

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

# MAC Address Control

一般設定 WPS 無線橋接 **無線存取控制** RADIUS設定 專業設定

### 無線網路 - 無線存取控制

此功能能夠控制WL-520GU無線區域網路中，特定網路卡實體位址（MAC Address）的存取。

MAC存取模式：	<div>未啟用 ▼</div> <div>未啟用 允許模式 拒絕模式</div>	<input type="text"/>	新增
MAC位址：	請輸入完整包含12個16進位制字母的MAC位址，0~9，A~F不包括"："。		
MAC存取控制名單：	<div></div>		刪除

套用本頁面設定

# Bandwidth Management

網際網路設定
**頻寬管理**
通訊埠觸發程式
虛擬伺服器
DMZ
DDNS

### 頻寬管理 - 使用者指定服務

提供高、一般與低三個優先權。例如：您可設定來源IP位址為192.168.1.3的使用者，在FTP服務上，其21連接埠的優先權為最高。

頻寬狀態

偵測總上傳頻寬：Kb/s

手動指定總上傳頻寬： Kb/s

使用者指定規則表

服務名稱	來源 IP 位址	目的地連接埠	優先權	
<input type="text"/>	<input type="text"/>	<input type="text"/>	一般 ▼	新增

目前沒有資料

☐ 長封包分割

套用本頁面設定

# Port Trigger

網際網路設定
頻寬管理
**通訊埠觸發程式**
虛擬伺服器
DMZ
DDNS

### NAT設定 - 通訊埠觸發程式

「通訊埠觸發程式」功能讓您可以開啓特定的TCP或UDP通訊埠來與連接 WL-520GU 的電腦進行通訊。而藉由指定觸發程式通訊埠及內傳通訊埠的方式即可完成此一操作。一旦偵測到觸發程式通訊埠，送往指定內傳通訊埠號的上傳封包便會轉向您的電腦。

#### 觸發程式通訊埠清單

啓用通訊埠觸發程式？ ☐ Yes ☒ No

常見的應用：

說明	觸發程式通訊埠	通訊協定	內傳通訊埠	通訊協定	
<input type="text"/>	<input type="text"/>	TCP	<input type="text"/>	TCP	<input type="button" value="新增"/>

目前沒有資料



# Virtual Server

網路網路設定 頻寬管理 通訊埠觸發程式 虛擬伺服器 DMZ DDNS

### NAT設定 - 虛擬伺服器

要由您所在網路當中的伺服器對網外用戶提供像是WWW、FTP等的網路服務時，您應該先要指定一連結伺服器的本地IP位址。接著，將IP位址及網路通訊協定的類型、通訊埠編號及網路服務名稱新增至下列的清單之中。以此清單為前題，開道器會將網路服務要求由網外用戶轉遞給相對的本地伺服器。

啟用虛擬伺服器？ ☒ Yes ☐ No

內建的伺服器應用： FTP

內建的遊戲應用： 請選擇

服務名稱	通訊埠範圍	本地IP	本地通訊埠	協定	網路服務
FTP Server	20:21	192.168.1.10		TCP	目前沒有資料

目前沒有資料

新增

套用本頁面設定

# DDNS

網際網路設定

頻寬管理

通訊埠觸發程式

虛擬伺服器

DMZ

DDNS

外部網路 - DDNS

動態DNS (DDNS) 讓您即使在沒有靜態IP位址的情況下，仍可將伺服器連同特有名稱一併匯出至網際網路上。目前，在 WL-520GU 之中已有數台嵌入的DDNS客戶端。您可點選下方的「免費試用」，來啟用一個免費試用帳號。

啟用DDNS Client?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
伺服器:	<div>WWW.DYNDNS.ORG</div> <div>免費試用</div>	<div>WWW.DYNDNS.ORG</div> <div>WWW.ASUS.COM</div> <div>WWW.DYNDNS.ORG</div> <div>WWW.DYNDNS.ORG(CUSTOM)</div> <div>WWW.DYNDNS.ORG(STATIC)</div> <div>WWW.TZO.COM</div> <div>WWW.ZONEEDIT.COM</div>
用戶名稱或E-mail帳號:	<input type="text"/>	
密碼或DDNS金鑰:	<input type="password"/>	
主機名稱:	<input type="text"/>	<div>查詢</div>
啟用萬用字元 (wildcard) ?	<input type="radio"/> Yes <input checked="" type="radio"/> No	
手動更新:	<div>更新</div>	
<div>套用本頁面設定</div>		

# FireWall

一般設定
網址過濾
MAC存取控制
封包過濾

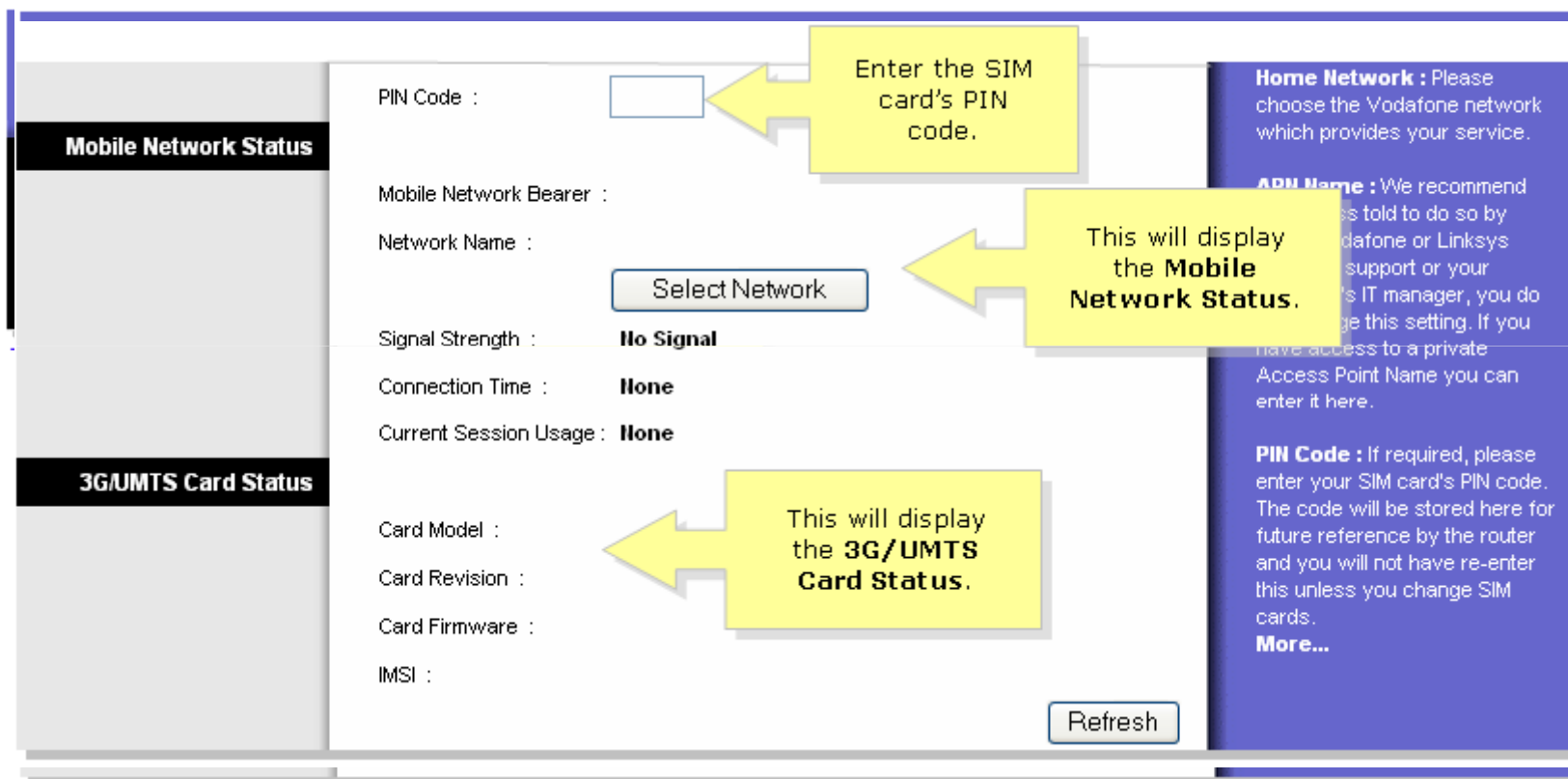
### 防火牆 - 一般設定

「啓用防火牆」功能可為 WL-520GU 及在其之後的裝置提供基本的保護。如果您要過濾出一些特定的封包，請採用在次頁中所述的LAN及WAN過濾功能。

啓用防火牆？	<input checked="" type="radio"/> Yes <input type="radio"/> No
啓動DoS防護？	<input type="radio"/> Yes <input checked="" type="radio"/> No
紀錄的封包類型：	None <input type="button" value="v"/>
從網際網路設定 WL-520GU？	<input type="radio"/> Yes <input checked="" type="radio"/> No
網際網路設定通訊埠：	8080
回應LPR要求？	<input type="radio"/> Yes <input checked="" type="radio"/> No
回應PING要求？	<input type="radio"/> Yes <input checked="" type="radio"/> No

套用本頁面設定

# 3G Broadband Router



The screenshot shows the configuration interface of a 3G Broadband Router. It is divided into two main sections: **Mobile Network Status** and **3G/UMTS Card Status**. The **Mobile Network Status** section includes fields for PIN Code, Mobile Network Bearer, Network Name, Signal Strength (displaying "No Signal"), Connection Time (displaying "None"), and Current Session Usage (displaying "None"). A "Select Network" button is located below the Network Name field. The **3G/UMTS Card Status** section includes fields for Card Model, Card Revision, Card Firmware, and IMSI. A "Refresh" button is located at the bottom right of the interface. Annotations with yellow arrows point to the PIN Code field, the "Select Network" button, and the Card Model field, providing instructions on how to use these features. A blue sidebar on the right contains additional information about Home Network, APN Name, and PIN Code.

**Mobile Network Status**

PIN Code :

Mobile Network Bearer :

Network Name :

Signal Strength : **No Signal**

Connection Time : **None**

Current Session Usage : **None**

**3G/UMTS Card Status**

Card Model :

Card Revision :

Card Firmware :

IMSI :

**Enter the SIM card's PIN code.**

**This will display the Mobile Network Status.**

**This will display the 3G/UMTS Card Status.**

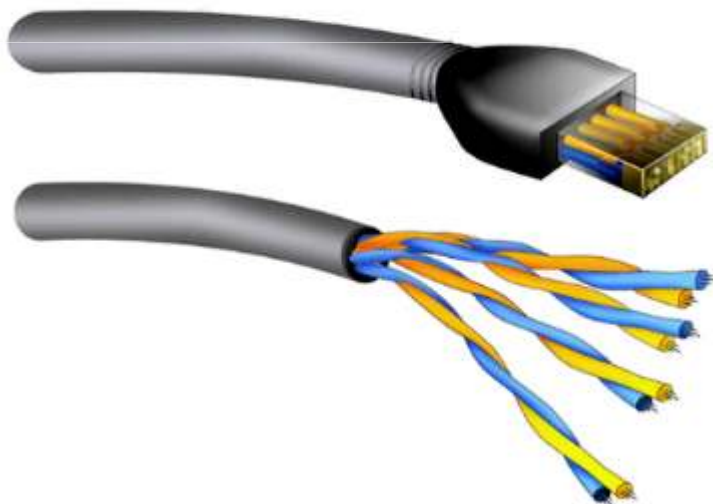
**Home Network :** Please choose the Vodafone network which provides your service.

**APN Name :** We recommend you to do so by Vodafone or Linksys support or your IT manager, you do this setting. If you have access to a private Access Point Name you can enter it here.

**PIN Code :** If required, please enter your SIM card's PIN code. The code will be stored here for future reference by the router and you will not have re-enter this unless you change SIM cards.

**More...**

# 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***

Creating Wi-Fi Enabled Applications

# 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**



Explorer 16  
Development Board

DV240001



PICDEM.net™ 2 Development Board

DM163024



AC164136-4



# 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

```

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_CLEANING
#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 addr:
// #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\TCP\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

WF\_Config.h

```
C:\MAL\May2010\TCP/IP 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"

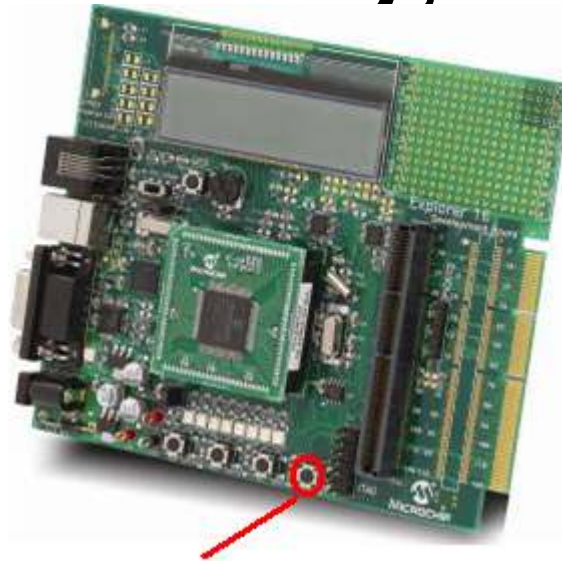
//-----
// Default WEP keys used in WF_SECURITY_WEP_40 and WF_SECURITY_WEP_104 security mode
//-----
#define MY_DEFAULT_WEP_PHRASE                "WEP Phrase"

// string 4 40-bit WEP keys -- corresponding to passphrase of "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

# Configure Infrastructure

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

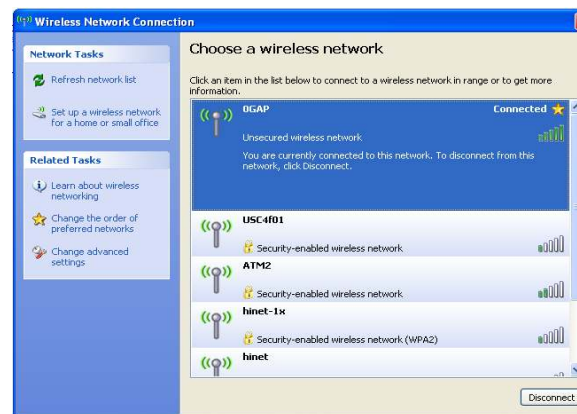


# Test Procedure(ping command)

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



- Verify connection from PC/access point



# Test Procedure(ping command)

- **Ping Command**

```
C:\WINDOWS\system32\cmd.exe
Ethernet adapter Wireless Network Connection:

    Connection-specific DNS Suffix  . : guest.zerogwireless.com
    IP Address. . . . . : 192.168.1.102
    Subnet Mask . . . . . : 255.255.255.0
    Default Gateway . . . . . : 192.168.1.1

Ethernet adapter Local Area Connection:

    Media State . . . . . : Media disconnected

C:\Documents and Settings\SChang>ping 192.168.1.101

Pinging 192.168.1.101 with 32 bytes of data:

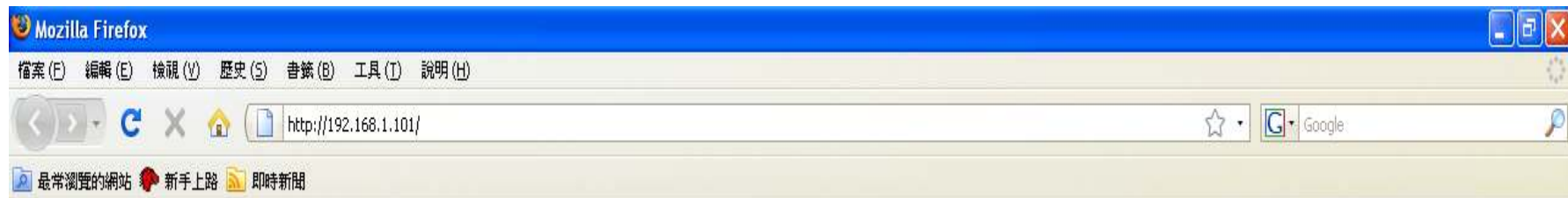
Reply from 192.168.1.101: bytes=32 time=79ms TTL=100
Reply from 192.168.1.101: bytes=32 time=21ms TTL=100
Reply from 192.168.1.101: bytes=32 time=28ms TTL=100
Reply from 192.168.1.101: bytes=32 time=20ms TTL=100

Ping statistics for 192.168.1.101:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 20ms, Maximum = 79ms, Average = 37ms
```



# Test Procedure(webserver)

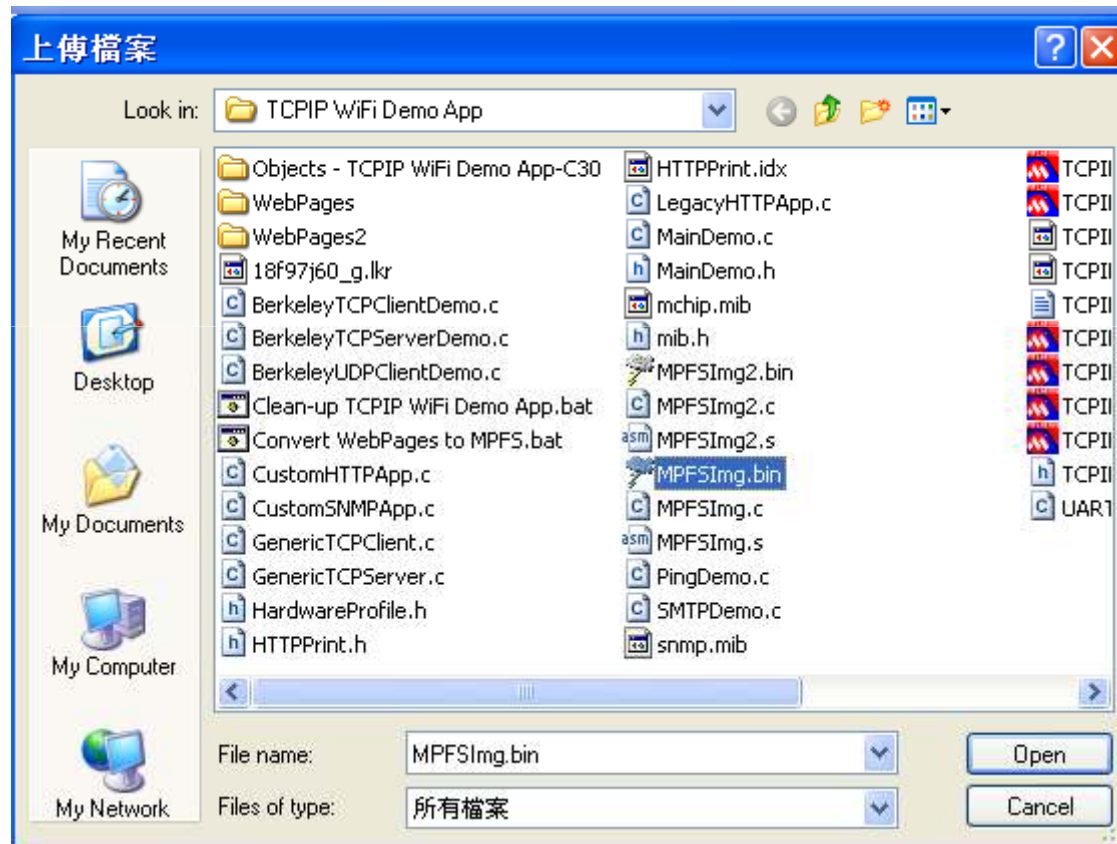
- Brower on Explorer 16 Board



404: File not found  
Use [MPFS Upload](#) to program web pages

# Test Procedure(webserver)

- Upload MPFSImg.bin

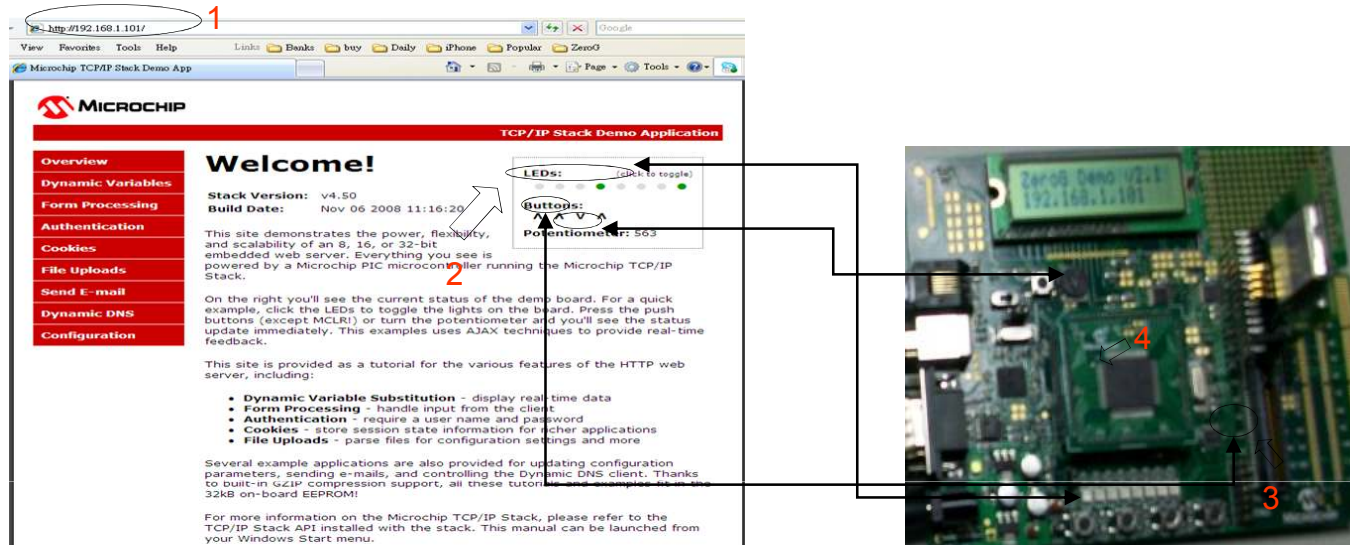


# Test Procedure(WebServer)

- 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 Explorer 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

# Handon



# **MICROCHIP**

---

***Regional Training Centers***

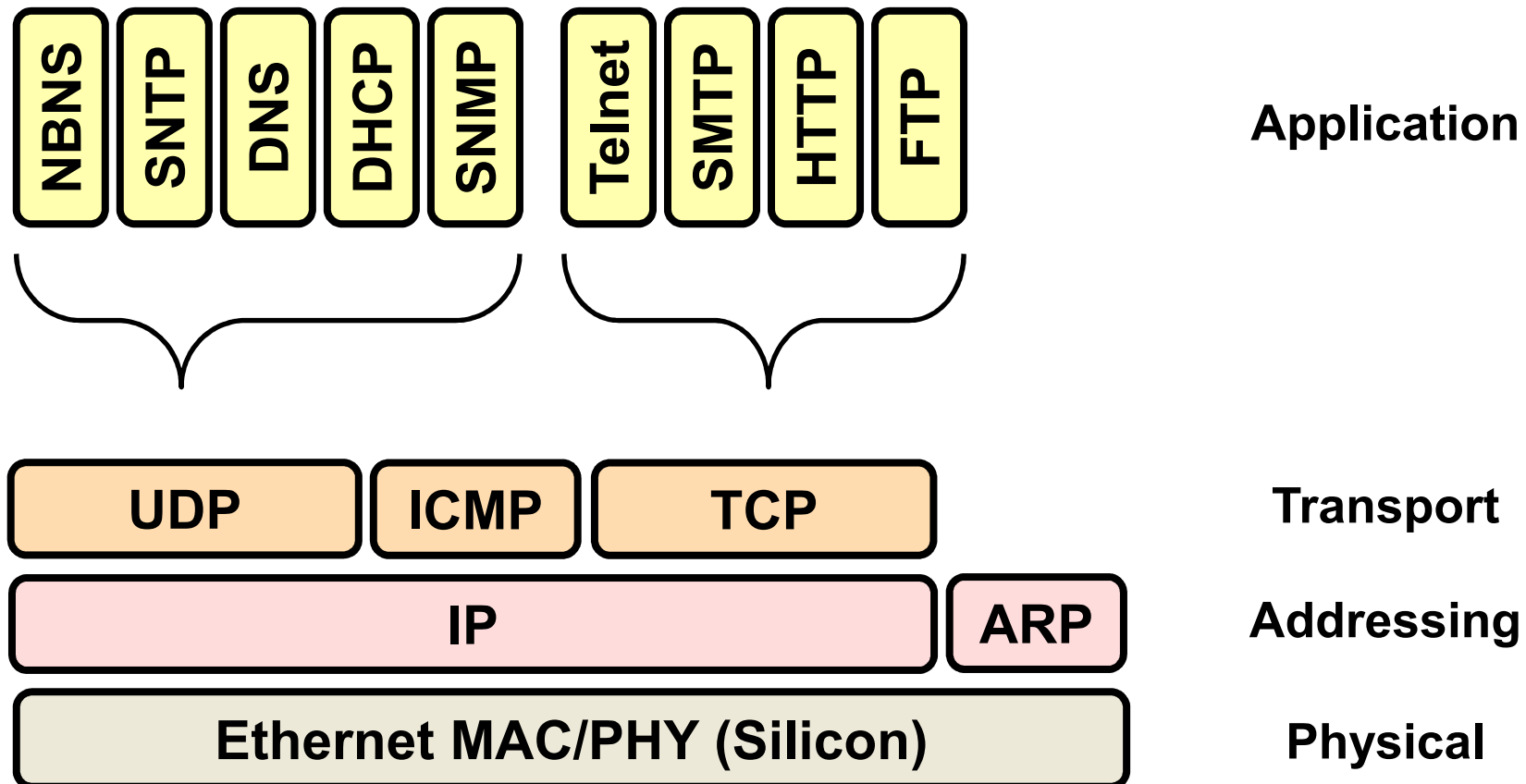
Software Architecture

# TCP/IP Stack



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

# What's Included?





# 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

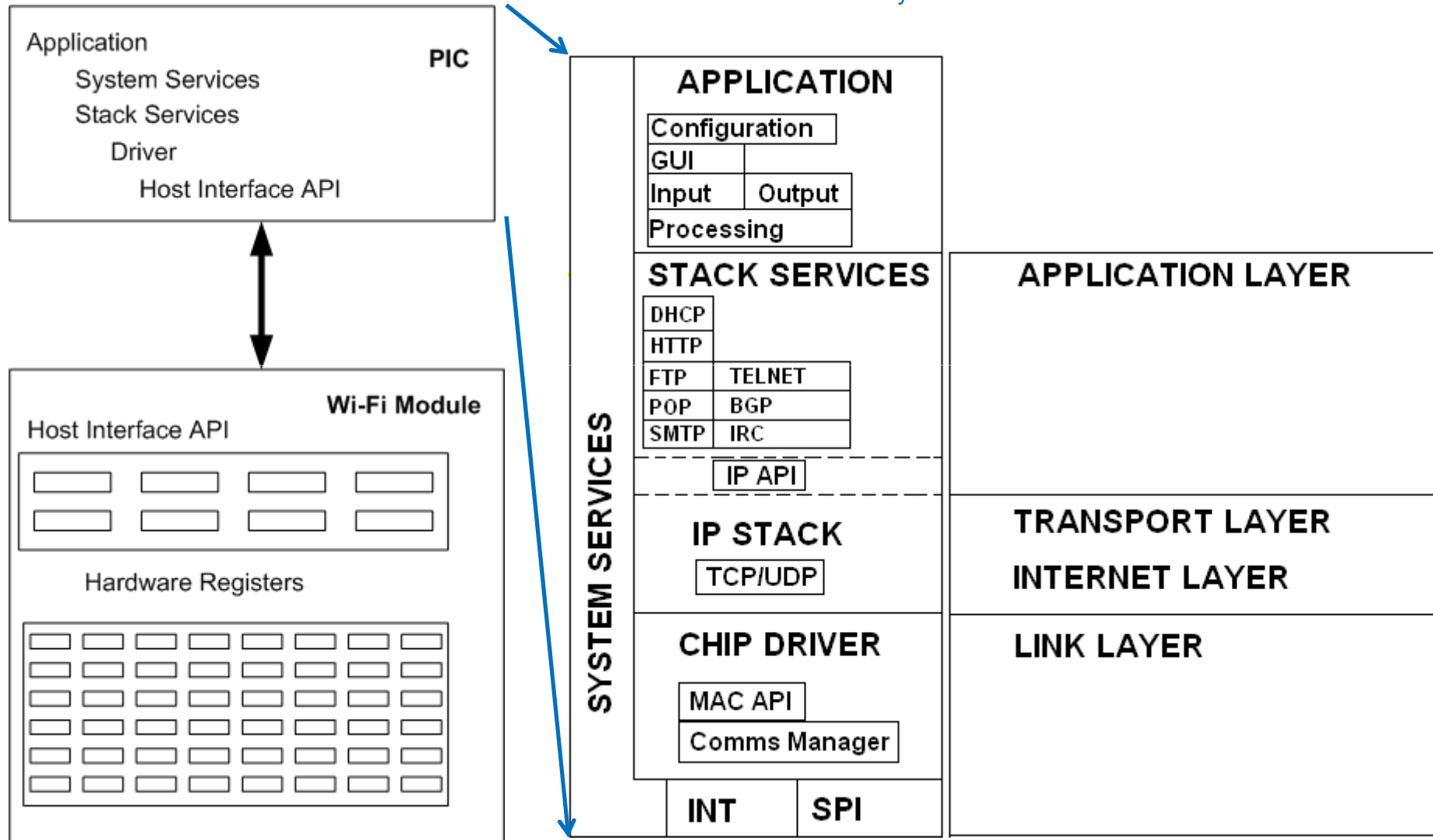
**DDNS**

## **Dynamic Domain Name Service Client**

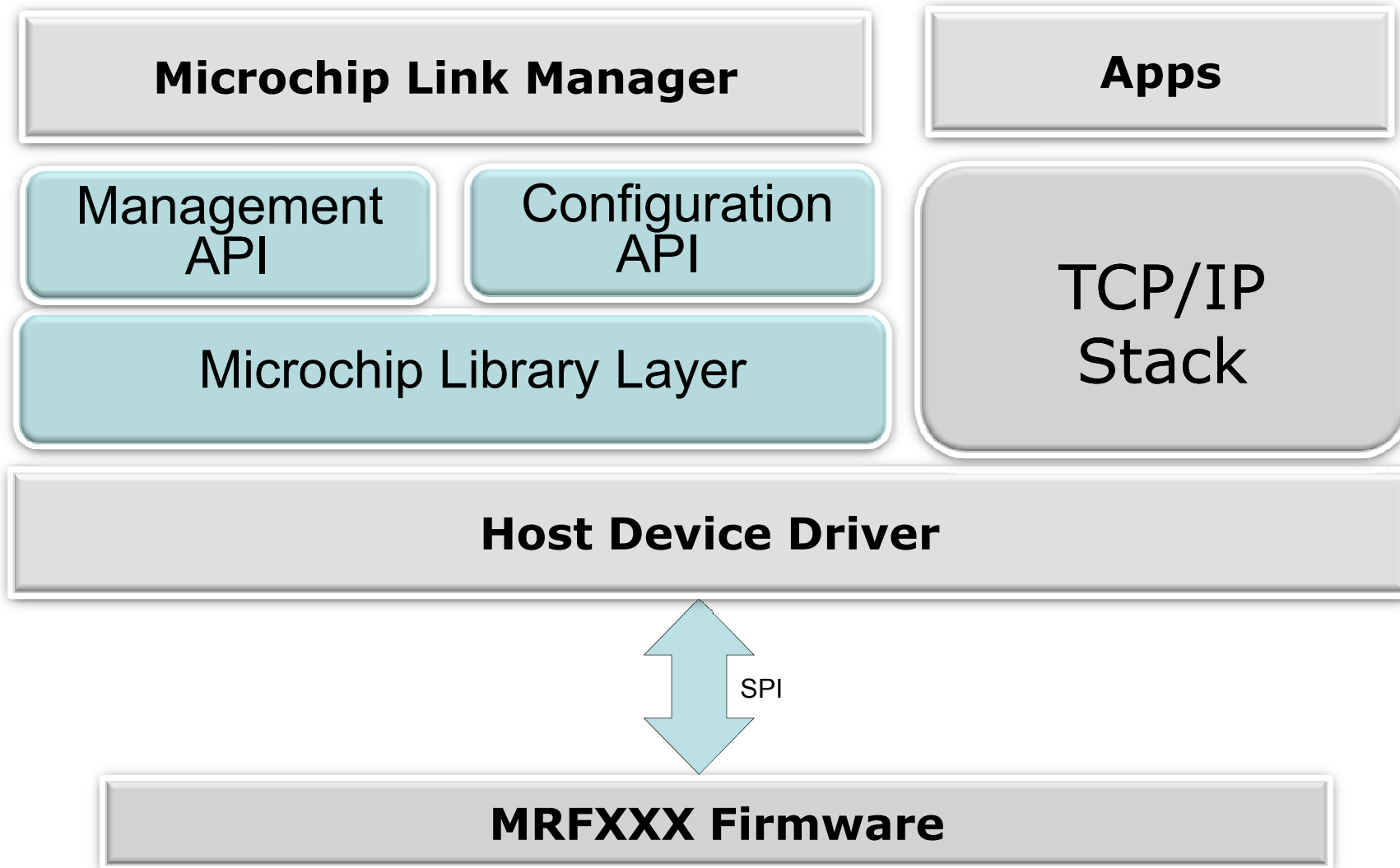
Associate host names with dynamic addresses

# Microchip 802.11 Solution

System & Software Architecture

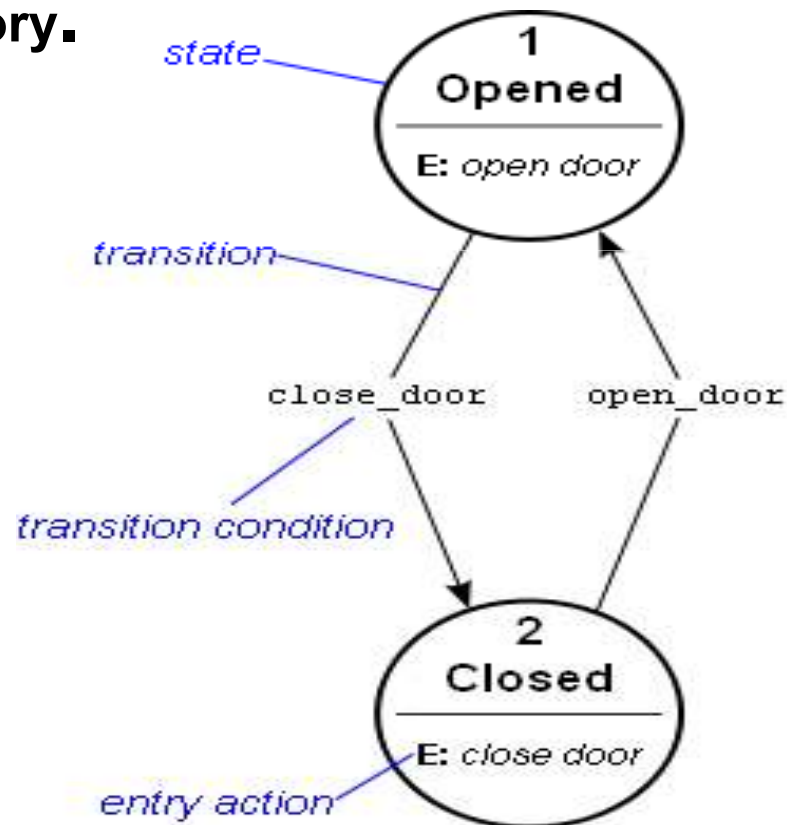


# System Block Diagram

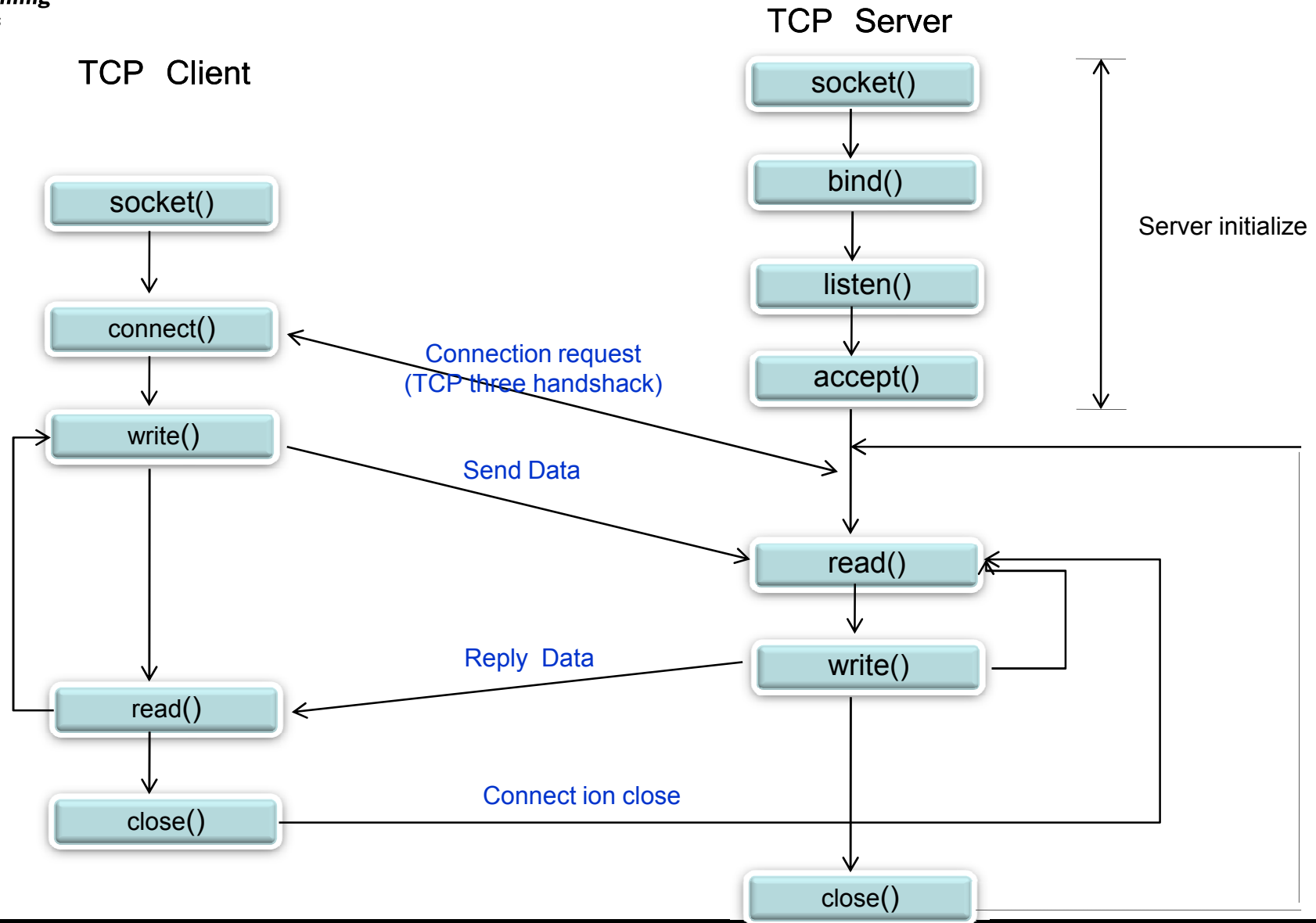


# FSM(Finite-state machine)

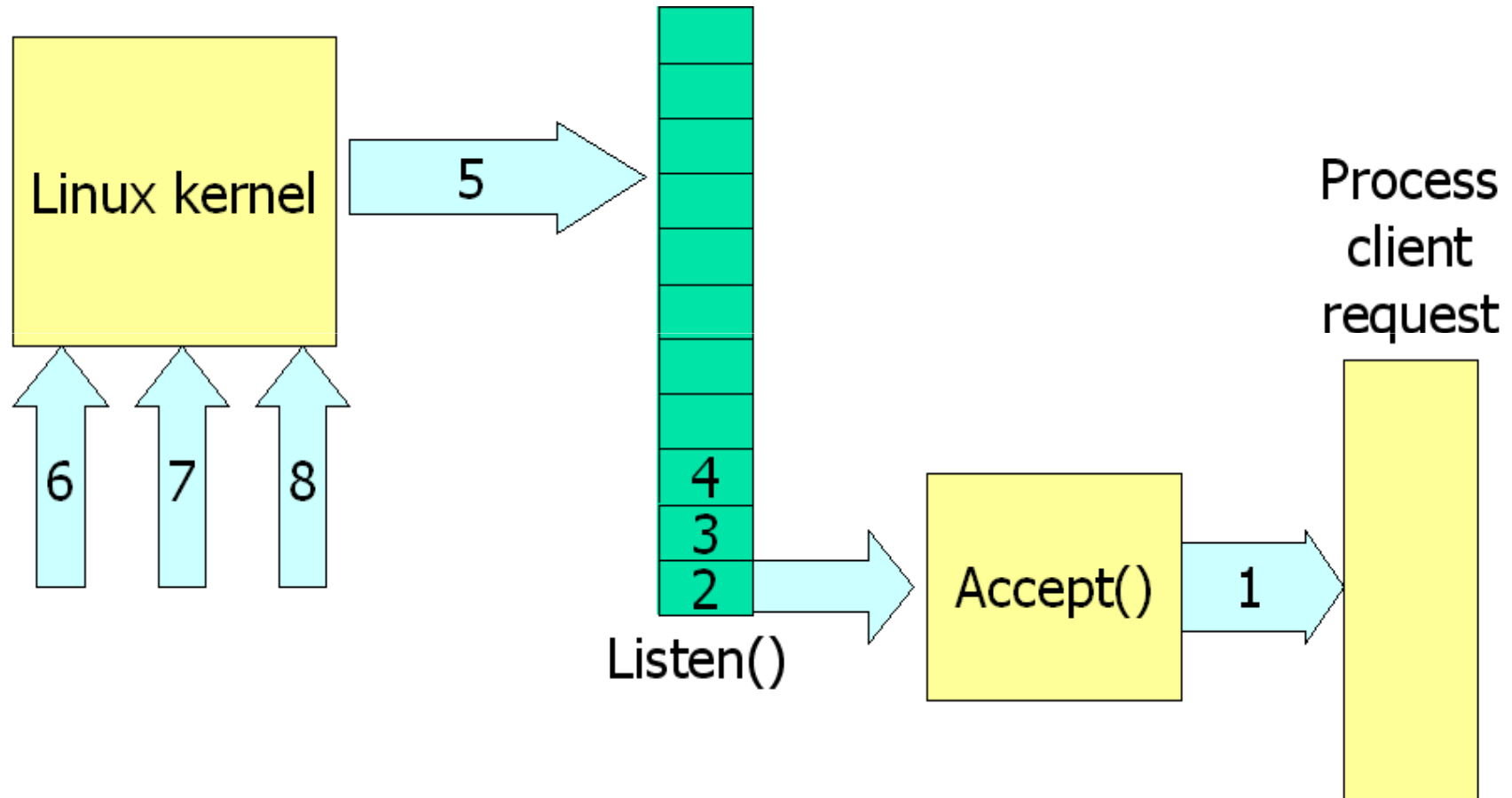
- FSM is a model of behavior composed of a finite number of states, transitions between those states, and actions. A finite state machine is an abstract model of a machine with a Primitive internal memory.



# TCP Client-Server



# Linux TCP/IP



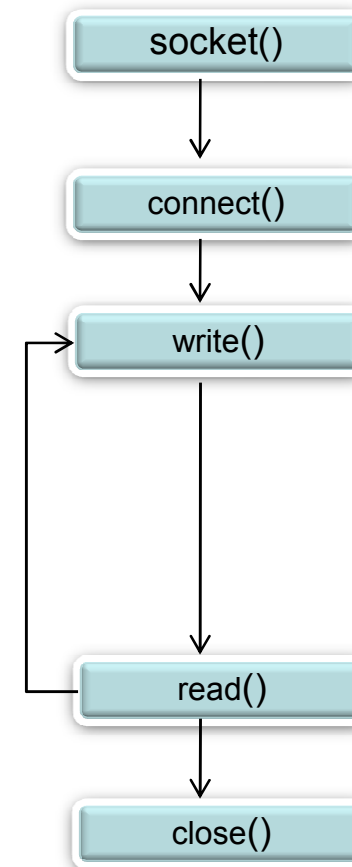
# Linux Client source code

```
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 TCP/IP Stack





Regional Training  
Center

# 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 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 BARDClose_CLOSE:
    closesocket(bsdClientSocket);
    BARDCloseClientState = BARDClose_DONE;

case BARDClose_DONE:
    ZGConsoleReleaseConsoleMsg();
    BARDCloseClientState = BARDClose_START;
    break;

default:
    return;
}

}
```



# **MICROCHIP**

---

***Regional Training Centers***

WiFi Programming

# 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**

# 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



# Add Code on WF\_Config.c

```

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

```



# **MICROCHIP**

---

## ***Regional Training Centers***

# **Thank You**

*Note: The Microchip name and logo, dsPIC, MPLAB and PIC are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries.  
MiWi, PICDEM and PICtail are trademarks of Microchip Technology Inc. in the U.S.A. and other countries.  
All other trademarks mentioned herein are property of their respective companies.*