The Power of Three

- ZigBee/802.15.4
  - Redes de sensores de bajo consumo
    - gran cantidad de estaciones, baja velocidad, bajo throughput, Poisson
- Bluetooth
  - periféricos de PC y celulares
- Wi-Fi
  - Multimedia, pocas estaciones, alta velocidad, alto throughput, streaming
  - Reutilización de infraestructura
  - Acceso desde PCs y/o Internet (sin gateway ad hoc)

Qu'est-ce que c'est la Wi-Fi ?

- Wi-Fi Alliance → 

 "Wi-Fi CERTIFIED"
- IEEE
  - 802.11
    - Recommendation (recomendación, "el standard")
      - 802.11-1997 (obsoleta)
      - 802.11-2007
      - 802.11-2012
    - Ammendments (enmiendas)
      - 802.11a
      - 802.11b
      - 802.11g
      - 802.11n

Quid hōc ad hoc ?

- Ad hoc
  - Hosts que se comunican entre sí por su cuenta
- Infrastructure
  - Red administrada
  - Access points (puntos de acceso).
    - SSID (Service Set IDentifier)

Seguridad

- 802.11b
  - WEP (Wired Equivalent Privacy)
    - clave de 40 ó 104 bits
- 802.11b/g y posteriores
    - frase ASCII para recordar genera clave mediante proceso computationally intensive
  - WPA/TKIP (Wi-Fi Protected Access using Temporal Key Integrity Protocol)
    - firmware upgrade a WEP antes de 802.11i
  - WPA/CCMP (Counter Mode with Cipher Block Chaining Message Authentication Code Protocol)
    - solución intermedia de la que se recomienda alejarse (no-standard)
  - "WPA2" (WPA2/CCMP o WPA2/AES)
    - 802.11i

Seguridad orientada a

- Personal
  - PSK (Pre-Shared Key)
    - la clave se utiliza para el cifrado (WEP)
    - la clave genera claves transitorias (WPA)
- Enterprise
  - Autenticación
    - EAP (Extensible Authentication Protocol)
      - EAP-TLS y familia
    - Servidor de autenticación (RADIUS)
- Home Network, easier for you! buy my device!
  - WPS (Wi-Fi Protected Setup) → Desactivalo YA!

Identificación

- SSID (Service Set IDentifier)
  - 32 bytes
  - Difundido en forma periódica por los access points
    - Es posible suprimir este broadcast periódico.
      - Sin embargo, en el momento en que un dispositivo se incorpora a la red, el SSID es transmitido sin cifrar

Wi-Fi no es la panacea

- No está por encima de las leyes físicas que se aplican a los sistemas de comunicaciones inalámbricas
  - Sí provee algunos mecanismos para resolver problemas conocidos
- No puede hacer nada que TCP/IP no haga, dado que la utilizamos como layer-2
  - Es un mero reemplazo de Ethernet/802.3
- El acceso al medio está sujeto a colisiones y por lo tanto no es determinístico

Especificaciones técnicas

- Ingreso a la red
  - scanning + sync
    - pasivo (esperar beacon)
    - activo (pedir)
  - autenticación
  - asociación
- CSMA/CA
  - DCF
  - PCF

Ingreso a la red: peor caso en una red 802.11b de alto tráfico

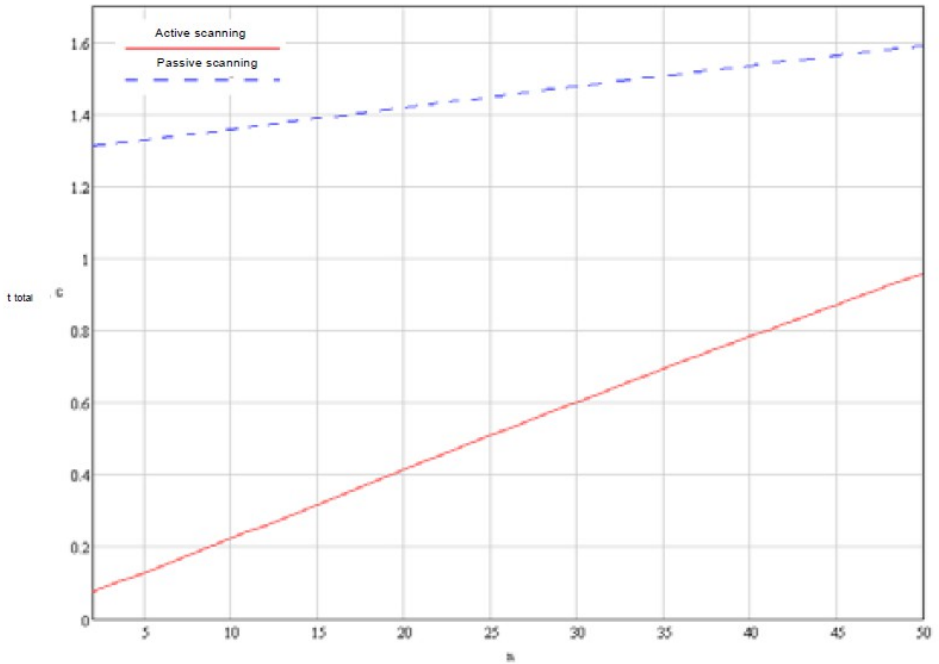


Fig. 1 Dependence of connection establishment time from quantity of stations in a service zone

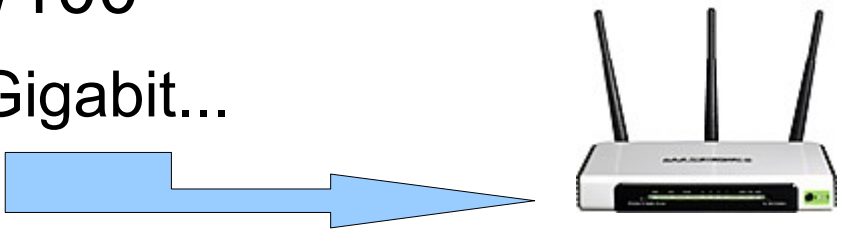
Especificaciones técnicas

- 802.11a
  - OFDM, 5GHz, 20MHz/ch 54Mbps
- 802.11b
  - DSSS, 2,4GHz, 22MHz/ch 11Mbps
- 802.11g
  - OFDM, 2,4GHz, 20MHz/ch 54Mbps
- 802.11n
  - OFDM, 2,4GHz + 5GHz, 20/40MHz/ch 54/150Mbps/link
  - MIMO

Full HD, Full LED, " Full 'n' " ?

- Marketing vs. vida real
- MIMO (Multiple Input Multiple Output)
  - SDM (Spatial Division Multiplexing)
    - 1 x 150 = 150
    - 2 x 150 = 300
    - 4 x 150 = 600

- Principio de Dirichlet
  - 300Mbps no caben en 10/100
    - si la interfaz de red no es Gigabit...



¿Existe un límite para esto?

- 802.11ac (DRAFT 2012)
  - 5Ghz, 800+Mbps
  - "The next generation of Wi-Fi"
    - ¡Compralo ya, no esperes a que sea obsoleto!
- 802.11ad
  - 7Gbps
    - haciéndote comprar todo de nuevo en 2014

Opciones Embedded

- Hardware Wi-Fi on chip
  - Rabbit 5000/6000
    - full host TCP/IP
- Chipset + diseño ad hoc
  - riesgos por mercado informático
- Módulo serie/SPI
  - sólo Wi-Fi (PHY + layer-2)
  - TCP/IP off-load (en el módulo)
    - XBee Wi-Fi
      - comunicación mediante un socket

Rabbit 5000/6000

- Hardware Wi-Fi on chip
- Dynamic C
  - full host TCP/IP
  - no-intrusivo
    - (el usuario manda, no el stack)

```
main()
{
    sock_init();
    while(1){
        tcp_tick();
        // mi código
    }
}
```

Web server en Dynamic C

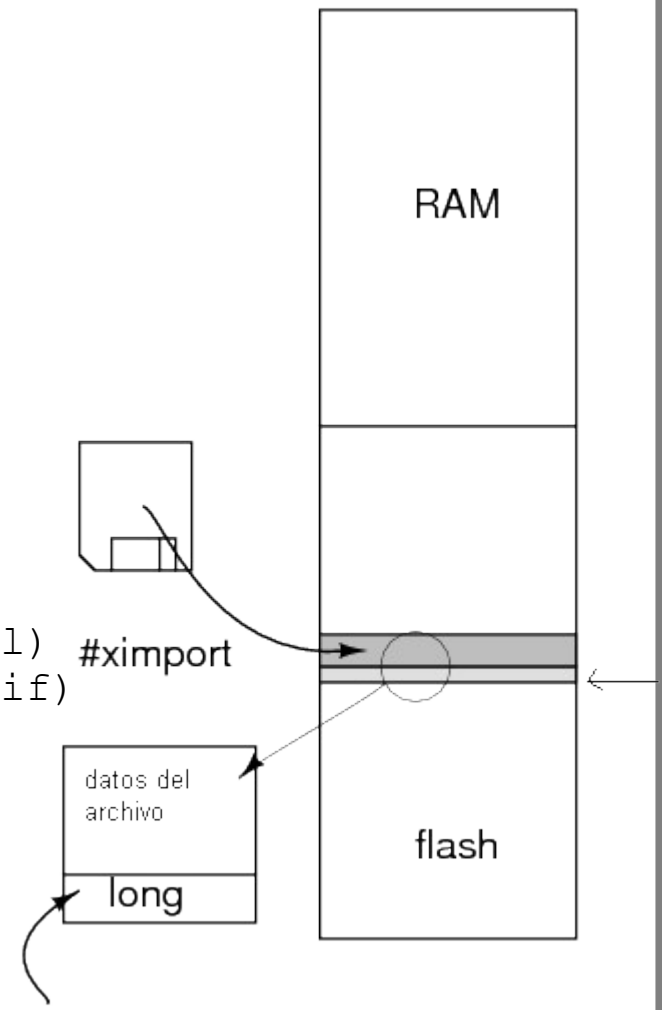
```
#ximport "index.html"      index_html
#ximport "rabbit1.gif"     rabbit1_gif

#use "dcrtcp.lib"
#use "http.lib"


SSPEC_MIMETABLE_START
    SSPEC_MIME(".html", "text/html")
    SSPEC_MIME(".gif", "image/gif")
SSPEC_MIMETABLE_END


SSPEC_RESOURCETABLE_START
    SSPEC_RESOURCE_XMEMFILE("/", index_html)
    SSPEC_RESOURCE_XMEMFILE("/index.html", index_html)
    SSPEC_RESOURCE_XMEMFILE("/rabbit.gif", rabbit1_gif)
SSPEC_RESOURCETABLE_END

main()
{
    sock_init();
    http_init();
    while(1){
        http_handler();
    }
}
```



Ethernet o Wi-Fi en Dynamic C

```
#define TCPCONFIG 0
#define USE_ETHERNET      1

#define MY_IP_ADDRESS   "192.168.1.54"
#define MY_NETMASK      "255.255.255.0"

#define MY_GATEWAY    "192.168.1.1"
```

```
#define TCPCONFIG 1
#define USE_WIFI      1
         #define IFC_WIFI_SSID         "Cika"
         #define IFC_WIFI_MODE         IFPARAM_WIFI_INFRASTRUCTURE
         #define IFC_WIFI_REGION       IFPARAM_WIFI_REGION_AMERICAS
         #define IFC_WIFI_ENCRYPTION   IFPARAM_WIFI_ENCR_NONE

#define MY_IP_ADDRESS "192.168.1.54"
#define  _PRIMARY_STATIC_IP MY_IP_ADDRESS
#define MY_NETMASK      "255.255.255.0"
#define  _PRIMARY_NETMASK MY_NETMASK

#define MY_GATEWAY    "192.168.1.1"
```

Wi-Fi en Dynamic C

```
#define USE_WIFI    1
#define WIFI_USE_WPA
#define WIFI_AES_ENABLED

const unsigned int enc[]={
    IFPARAM_WIFI_ENCR_NONE,
    IFPARAM_WIFI_ENCR_TKIP,
    IFPARAM_WIFI_ENCR_CCMP
};
const unsigned int auth[]={
    IFPARAM_WIFI_AUTH_OPEN,
    IFPARAM_WIFI_AUTH_WPA_PSK,
};

    ifconfig(IF_WIFI0,
        IFS_IPADDR, mysrc_ip,
        IFS_NETMASK, mysrc_mask,
        IFS_ROUTER_SET, mydef_gwy,
        IFS_NAMESERVER_SET, mydns,
        IFS_WIFI_SSID, strlen(myssid), myssid,
        IFS_WIFI_ENCRYPTION, enc[myenc],
        IFS_WIFI_AUTHENTICATION, auth[myauth],
        IFS_WIFI_WPA_PSK_HEXSTR,myhexkey,
        IFS_END);
```

DHCP en Dynamic C

- Indicar uso siempre

```
#define TCPCONFIG 5
```

- Configuración dinámica
  - mediante ifconfig()

```
ifconfig(
   ...
   IFS_DHCP, 1
   ...
   );
```

Resolución por DNS en Dynamic C

- blocking

```
resolve("");
```

- non-blocking

```
handle = resolve_name_start(name);
retval = resolve_name_check(handle,&resolved_ip);
```

  - RESOLVE_AGAIN
  - RESOLVE_SUCCESS
  - RESOLVE_FAILED
  - ...

UDP en Dynamic C

- Abrir el socket, podemos indicar
  - en qué puerto vamos a transmitir y/o recibir
  - si escuchamos a cualquiera que nos hable en ese puerto, al primero que lo haga, o solamente a una dirección en particular
  - si le hablamos a alguien en particular o mandamos broadcasts dentro de nuestra subred
  - También es posible trabajar sobre multicast, pero resulta algo más complejo.

UDP en Dynamic C

- Abrir el socket

```
udp_open(&s,local port,remote IP, remote port, NULL );
```

- local port: donde escucho
- remote IP
  - IP: quien me habla
  - 0: el primero que me habla
  - -1: todos (transmito broadcasts además)
- remote port:
  - 0: el del primero que me habla
  - si remote IP == -1
    - ignorado en recepción
    - usado para transmisión
- NULL: si quiero definir un data handler, puntero al mismo
- “El primero que me habla”: el socket se arma al recibir el primer datagrama

UDP en Dynamic C

- Enviar un datagrama

```
udp_send(&s,buffer,longitud);
```

- Recibir un datagrama

```
if((len=udp_recv(&s,buffer,maxlen))>=0) // maxlen= tamaño del buffer
    // tengo un datagrama de len bytes disponible en buffer
```

- Ignorar el checksum

```
sock_mode(&s,UDP_MODE_NOCHK);
```

UDP en Dynamic C

```
main()
{
udp_Socket s;
static char buffer[1024];
int i,j;

    sock_init();
    while (ifpending(IF_DEFAULT) == IF_COMING_UP) tcp_tick(NULL);
    if(udp_open(&s,LOCAL_PORT,DEST_IP,DEST_PORT,NULL)==0) exit(1);
    while(1){
        tcp_tick(&s);
        costate {
            while((i=udp_send(&s,MY_STRING,strlen(MY_STRING)))==-2)
                yield; // no resolvió la MAC
            if(i==-1)
                waitfor(DelaySec(1)); // falla, reintentar luego
        }
        costate {
            while((j=udp_recv(&s,buffer,1024))<0)
                yield; // no hay mensajes
            if(j>0){
                // procesar mensaje
            }
        }
    }
    sock_close(&s);
}
```

TCP en Dynamic C

- Abrir el socket
  - como cliente, inicia conexión

    ```
    tcp_open(&socket,0L,DEST_IP,DEST_PORT,NULL)
    ```
  - como servidor, espera conexión

    ```
    tcp_listen(&socket,MY_PORT,0,0,NULL,0)
    ```

- Esperar establecimiento de la conexión

```
(sock_established(&socket)) || (sock_bytesready(&socket) > 0)
```

TCP en Dynamic C

- Configurar

```
sock_mode(&socket,TCP_MODE_BINARY);
```

- Transferir datos
  - Enviar

```
bytes_written=sock_fastwrite(&socket,buffer,len);
```

  - Recibir

```
if((bytes_read=sock_fastread(&socket,buffer,bufsize))!=0)
```

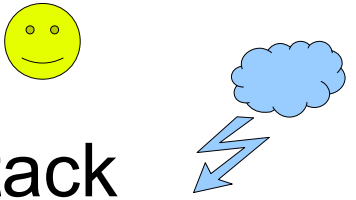
- Cerrar conexión

```
sock_close(&socket);
```

Fin de la sección Rabbit

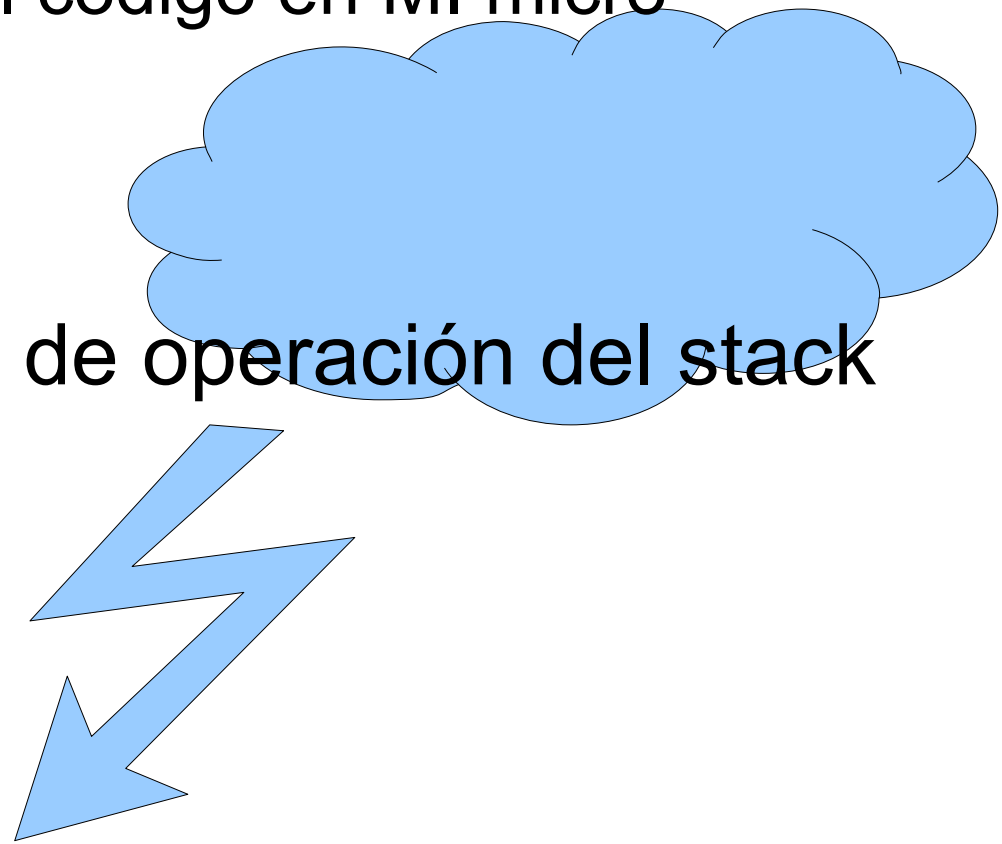
XBee Wi-Fi

- Comunicación serie
  - UART/SPI
  - comandos AT
    - operación transparente
    - sólo un socket
  - modo "API"
    - control por mensaje
- TCP/IP off-load
  - puedo usar un micro más chico
  - no tengo todo el control sobre el stack



¿De qué no dispongo?

- Web server, mail y/o demás aplicaciones
  - a menos que escriba el código en MI micro
- Resolución por DNS
  - ídem
- Control de parámetros de operación del stack TCP/IP



Wi-Fi en XBee

- SSID

  `ATIDCika`

- Encryption & Co.

  `ATEE=`

  - `0: open`
  - `1: WPA/TKIP`
  - `2: WPA2/CCMP`
  - `3: WEP`

  `ATKYnuncasabrascualesmiclave`

DHCP (o no) en XBee

- Usar DHCP

```
ATMA=0
```

- Configuración estática

```
ATMA=1
ATMY192.168.1.54
ATMK255.255.255.0
ATGW192.168.1.1
```

- Recordemos que no hay resolución por DNS

Comunicación en el "Modo transparente"

- Configuración

```
ATDL192.168.1.123    <-- dirección de destino
ATDE1234             <-- destination port
ATCO1234             <-- source port
ATIP=
```

  - 0: UDP
  - 1: TCP

- Comunicación
  - Lo que envío por el puerto serie se envía a destino
    - a menos que haya ingresado a modo comando (+++) para configurar
  - Lo que recibo sale por el puerto serie
    - no sé quién me lo envía (a menos que figure DENTRO del mensaje...)

Comunicación en el "Modo 'API'"

- Tramas "API"

`<0x7E><LEN: 2 bytes><INFO: len bytes><CHECKSUM>`

`INFO: <ID><DATA>`

- Configuración
  - Tramas "Comandos AT"
- Comunicación
  - Tramas "Enviar mensaje"
  - Tramas "Mensaje recibido"

Comunicación en el "Modo 'API'"

- Comunicación
  - Tramas "Enviar mensaje"

```
struct {
    unsigned char frameid;
    unsigned char addr[4];
    unsigned char dport[2];
    unsigned char sport[2];
    unsigned char proto;
    unsigned char options;
    unsigned char data[];
};
```

Comunicación en el "Modo 'API'"

- Comunicación
  - Tramas "Mensaje recibido"

```
struct {
    unsigned char addr[4];
    unsigned char dport[2];
    unsigned char sport[2];
    unsigned char proto;
    unsigned char status;
    unsigned char data[];
};
```