Tomás Fernández Pena
José Carlos Cabaleiro Domínguez
Oscar García Lorenzo

Grado en Robótic

Universidade de Santiago de Compostela

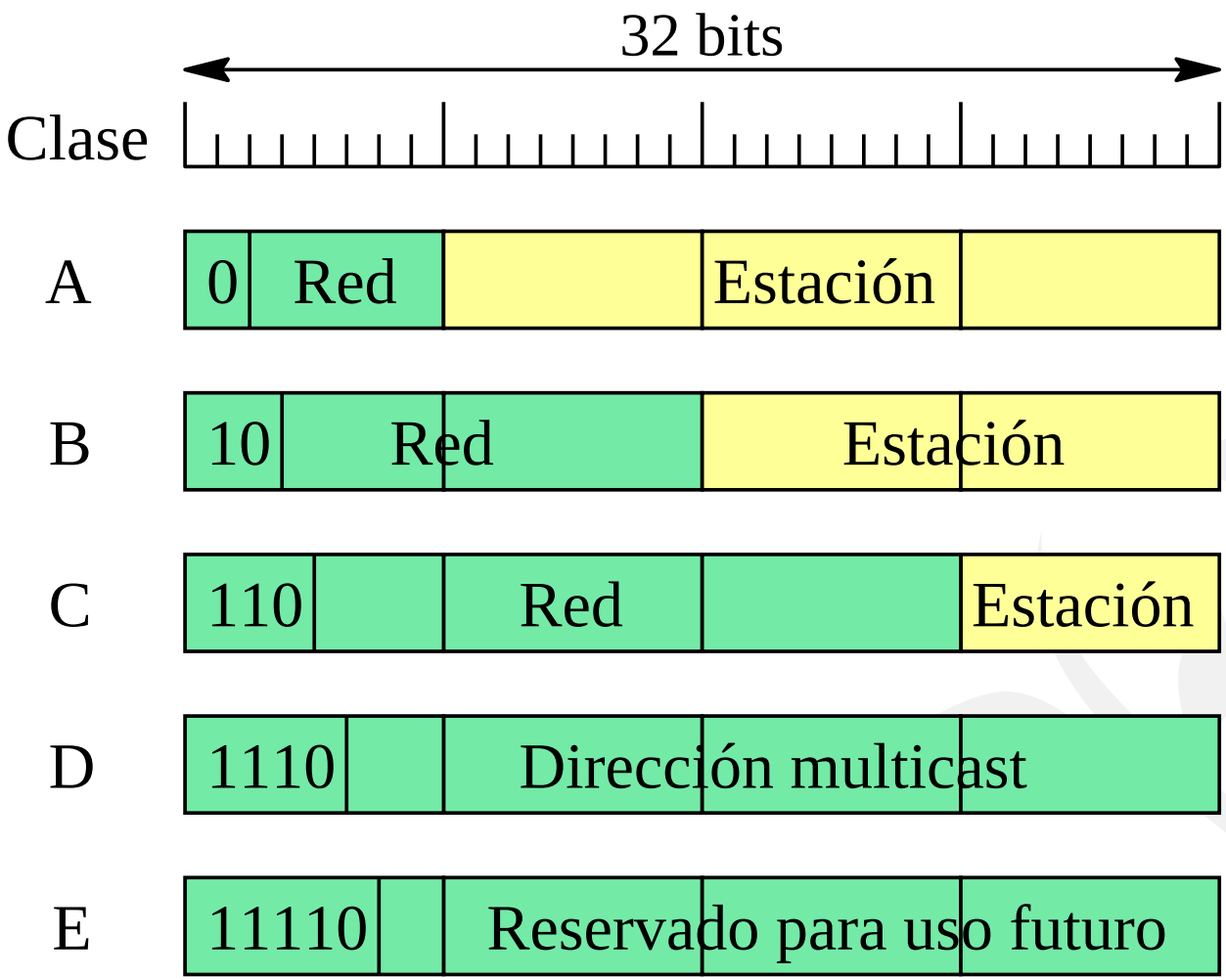
Redes e comunicacións, 3º Curso GrR

USC UNIVERSIDADE DE SANTIAGO DE COMPOSTELA
CAMPUS VIDA
CiTiUS Centro Singular de Investigación en Tecnoloxías Intelixentes

Índice

Índice

Direccionamiento IPv4 tradicional



Rango de direcciones

A: 1.0.0.1 hasta 126.255.255.254

B: 128.0.0.1 hasta 191.255.255.254

C: 192.0.0.1 hasta 223.255.255.254

D: 224.0.0.1 hasta 239.255.255.254

E: > 240.0.0.1

Direccionamiento IPv4 con clases

- Restricción: un campo de rede ou de host non pode estar todo a 1s ou todo a 0s
- Según isto temos:

  Clase A: 126 ($2^7 - 2$) redes con $\approx$ 16 millóns de estacións cada unha ($2^{24} - 2 = 16777214$)

  Clase B: 16384 (2^{14}) redes con 65534 ($2^{16} - 2$) estacións cada unha

  Clase C: $\approx$ 2 millóns de redes ($2^{21} = 2097152$) con 254 estacións cada unha ($2^8 - 2$)

- En C existen funcións de manexo das partes de rede e de host
  - `uint32_t inet_lnaof(),uint32_t inet_netof(), struct in_addr inet_makeaddr()`
  - So funciona con IPv4 de clases A, B y C

Direcciones especiais reservadas

- Identificación de redes: o nº de rede e o resto a 0
- Exemplos:
  - Clase A → 10.0.0.0
  - Clase B → 172.16.0.0
  - Clase C → 193.144.84.0
- Dirección de broadcast: o nº de rede e o resto a 1
- Exemplos:
  - Clase A → 10.255.255.255
  - Clase B → 172.16.255.255
  - Clase C → 193.144.84.255

Subredes e máscaras

Subredes

- Problema: o número de estacións nunha rede pode ser demasiado grande $\Rightarrow$ dificultades de administración
- Solución: dividir a rede en subredes, que se xestionen de forma independente pero que actúen como unha soa de cara ao exterior

Máscara de subrede

- Utilizamos parte do campo estación para indicar a subrede
- Empleamos **máscaras** para delimitar a subrede
- Formato de máscara: 32 bits dos que os n máis significativos están a 1 e os $32 - n$ restantes a 0
- Exemplo: máscara de 27 bits, denotase como sufixo $/27$
  $255.255.255.224 \equiv 11111111.11111111.11111111.11100000$

Direccionamiento IPv4: exemplo de máscara

- Dirección clase C 193.168.17.0/27 (ou máscara 255.255.255.224)
  - Os 24 primeros bits indican a rede (192.168.17)
  - Os 3 seguintes a subrede
  - Os 5 últimos a posición da estación na subrede
  - Temos $2^3 = 8$ subredes, con $2^5 - 2 = 30$ estacións por subrede
  - En total, podemos direccionar $8 \times 30 = 240$ estacións (254 en clase C sen máscara)

Redes sen clase

- En 1993, suprimense as clases
- Direccións CIDR (**Classless Inter-Domain Routing**)
  - Sufixo $/s \Rightarrow s$ bits para indicar a rede e $32 - s$ para indicar a estación ($2^{(32-s)} - 2$ estacións)
- Exemplos: 193.168.64.0/18, 130.0.0.0/8
- Tamén se coñecen como **superredes**
- Exemplo: 193.168.173.253/18
  - Nº de rede: 11000001.10101000.10000000.00000000 = 193.168.128.0 (tamén se usa 193.168.128)
  - Broadcast: 11000001.10101000.10111111.11111111 = 193.168.191.255
  - Nº estación: 11000001.10101000.10101101.11111101 = estación nº 11773
  - Nº total de estacións: $2^{14} - 2 = 16382$

Identificación de redes

- Para identificar as redes especificase o número de rede (coa parte de host a 0) e a máscara. Exemplos:
  - 193.168.64.0/18
  - 130.0.0.0/8
  - Pódense omitir os ceros da parte de host. Por exemplo:
    - 193.168.64/18
    - 130/8
- Lóxicamente, en formato binario son equivalentes
- Os routers inclúen entradas deste tipo para saber cara qué interface encamiñar os paquetes
- En C hai funcións para manexar estas notacións
  - `int inet_net_pton()` de textual a binario
  - `char *inet_net_ntop()` de binario a textual

Índice

e

Función `inet_net_pton`: textual a binario

```
int inet_net_pton(int af, const char *pres,
                  void *netp, size_t nsize)
```

- Converte a binario a IP especificada e devolve o sufixo[1]
- Parámetros:
  - `af` enteiro que debe valer `AF_INET`[2]
  - `pres` punteiro a un string coa IP/sufixo a convertir
  - `netp` punteiro ao resultado, debe apuntar a unha `struct in_addr`[3] e debe estar inicializada a 0
    - A IP guardase en orden de rede
  - `nsize` número de bytes en `netp`
- Valor devolto: un enteiro indicando o sufixo especificado, -1 en caso de erro
- Ao compilar, usar a opción `-lresolv`

[1] Función non estándar pero ampliamente dispoñible
[2] So soporta IPv4
[3] Debe apuntar a un área reservada co tamaño adecuado

Función `inet_net_pton`: textual a binario

Exemplo de uso:

```
struct in_addr mired;
if((sufijo = inet_net_pton(AF_INET, "193.20.102.40/14", (void *) &mired,
      sizeof(struct in_addr))) < 0) {
  fprintf(stderr, "Formato de direccion incorrecto");
  exit(EXIT_FAILURE);
}
printf("%X  %u\n", mired.s_addr, sufijo); // Imprime 286614C1  14

if((sufijo = inet_net_pton(AF_INET, "193.20.64/14", (void *) &mired,
     sizeof(struct in_addr))) < 0) {
  fprintf(stderr, "Formato de direccion incorrecto");
  exit(EXIT_FAILURE);
}
printf("%X  %u\n", mired.s_addr, sufijo); // Imprime 4014C1  14
```

Función `inet_net_pton`: binario a textual

```
char *inet_net_ntop(int af, const void *netp,
                    int bits, char *pres,
                    size_t psize)
```

- Convirte o número rede a formato presentación[4] (textual)
- Parámetros:
  - `af` igual que antes
  - `netp` punteiro a unha `struct in_addr`
  - `bits` número de bits da parte de rede (el sufixo)
  - `pres` punteiro á cadea na que se gardará o resultado[5]
  - `psize` número de bytes dispoñibles en `pres`
- Se en `netp` metese unha IP completa, en `pres` garda a parte de rede/sufijo
- Valor devolto: punteiro a `pres`, `NULL` en caso de erro
- Ao compilar, usar a opción `-lresolv`

[4] Función non estándar pero ampliamente dispoñible

[5] Debe apuntar a un área reservada co tamaño adecuado

Función `inet_net_pton`: binario a textual

Exemplo de uso:

```
struct in_addr mired, miip;
char red_text[INET_ADDRSTRLEN+3];

mired.s_addr = 0x4014C1;
miip.s_addr = 0x286614C1;
if(inet_net_ntop(AF_INET, (void *) &mired, 14, red_text, INET_ADDRSTRLEN
    +3) != NULL) {
  printf("%s\n", red_text);  // Imprime 193.20/14
}

if(inet_net_ntop(AF_INET, (void *) &miip, 12, red_text, INET_ADDRSTRLEN
    +3) != NULL) {
  printf("%s\n", red_text);  // Imprime 193.16/12
}
```

Manexo en Python, librería IPaddress

`ipaddress.ip_address(address)`,
`ipaddress.ip_network(address, strict=True)`

- Úsase para menexar direccións IPv4 ou IPv6 en Python
- As máscaras, direccións de rede, lonxitude de prefixo e demáis se ofrecen coma atributos dos obxetos
  - Ex: ip_network("193.144.16.0/24").prefixlen -> 24
- Devolven os tipos `IPV4Address`, `IPV6Address`, `IPV4Network` ou , `IPV6Network`, según corresponda (se se usan directamente os contructures co tipo adecuado os posibeis erros aparecen máis detallados)
- Tamén hai `IPV4Inteface`, `IPV6Inteface`
- Os de tipo `Network` teñen máis atributos, conteñen todas as direccións da rede como `Address`
  - ip_network("193.144.16.0/24")[0] -> 193.144.16.0
  - ip_address("193.144.16.3") in ip_network("193.144.16.0/24") -> True