Ćwiczenie 1

Wyznaczyć reprezentację dziesiętną liczby dwójkowej 10011101001

Mnożymy od prawej strony kolejne cyfry dwójkowe przez kolejne potęgi liczby 2 (począwszy od 0) i sumujemy iloczyny.

1*2^0 + 0*2^1 + 0*2^2 + 1*2^3 + 0*2^4 + 1*2^5 +1*2^6 + 1*2^7 + 0*2^8 + 0*2^9 + 1*2^10 = 1 + 8 + 32 + 64 + 128 + 1024 = 1257

Ćwiczenie 2

Jaką największa liczbę można zapisać na k bitach?

Największa liczba zapisana na k bitach to 2^0 + … + 2^(k – 1), w reprezentacji dwójkowej jest to liczba 1…1 (k bitów) = 10…0 – 0…01 (k+1 bitów) = 2^k – 1

Ćwiczenie 3

Zamienić liczbę dziesiętną 125 na liczbę dwójkową

Reprezentację dwójkową liczby dziesiętnej tworzą kolejne reszty z dzielenia przez 2, zapisywane w kolejności od ostatniej do pierwszej.

125/2 = 62 r 1, 62/2 = 31 r 0, 31/2 = 15 r 1, 15/2 = 7 r 1, 7/2 = 3 r 1, 3/2 = 1 r 1, 1/2 = 0 r 1

$(125)_{10} = (1111101)_2$

Ćwiczenie 4

Wypisać reprezentacje binarne kolejnych cyfr szesnastkowych

0: 0000

1: 0001

2: 0010

3: 0011

...

9: 1001 <- 9/2 = 4 r 1, 4/2 = 2 r 0, 2/2 = 1 r 0, 1/2 = 0 r 1

a: 1010

b: 1011

c: 1100

d: 1101

e: 1110

f: 1111

Reprezentacje bitowe przykładowych dwucyfrowych liczb szesnastkowych
44: 0100 0100, 8a: 1000 1010

Ćwiczenie 5

Ile razy więcej jest adresów IPv6 od adresów IPv4?

Ze standardowego zapisu adresu IPv6 wynika, że tych adresów jest 2^(8*16) = 2^128. Z kolei adres IPv4 jest zapisywany w postaci czterech liczb dziesiętnych z przedziału [0, 255], więc adresów IPv4 jest 2^(4*8) = 2^32. Zatem adresów IPv6 jest 2^(128 – 32) = 2^96 razy więcej od adresów IPv4.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Podstawowe informacje na temat adresów IPv4

Standardowo, adres IPv4 jest zapisywany w postaci 4 liczb dziesiętnych z przedziału [0, 255] oddzielonych kropkami. W zapisie dwójkowym adres IPv4 składa się z 4 oktetów, więc każdą z tych liczb dziesiętnych można przedstawić jako 8-cyfrową liczbę binarną, a największą taką liczbą jest $(11111111)_2 = (255)_{10}$. Przykładowy adres IPv4: 10.215.100.43

Adres IP służy do identyfikacji komputera w środowisku wielosieciowym, składa się z dwóch części – sieciowej identyfikującej sieć, oraz hostowej identyfikującej komputer w sieci. Słowo host oznacza urządzenie końcowe w sieci komputerowej.

Adresy IPv4 dzielą się na 5 klas – A, B, C, D i E. Do adresowania komputerów używane są adresy z klas A, B i C, adresy z pozostałych klas są używane do innych celów.
Klasę adresu IPv4 określa 1 oktet. Zakresy 1-szego oktetu dla klas A, B, C, D, E:
Dla klasy A: od 00000001 do 01111111, czyli od 1 do 127
Dla klasy B: od 10000000 do 10111111, czyli od 128 do 191
Dla klasy C: od 11000000 do 11011111, czyli od 192 do 223
Dla klasy D: od 11100000 do 11101111, czyli od 224 do 239 (adresy multicast)
Dla klasy E: od 11110000 do 11111111, czyli od 240 do 255 (adresy zarezerwowane dla uprawnionych organizacji zajmujących się rozwojem Internetu)

Klasa adresu A, B lub C określa domyślny rozmiar części sieciowej i hostowej.

Klasa	Część sieciowa identyfikująca sieć	Część hostowa ident. komputer w sieci
A	1-szy oktet (8 bitów)	Pozostałe 3 oktety (24 bity)
B	1-sze 2 oktety (16 bitów)	Pozostałe 2 oktety (16 bitów)
C	1-sze 3 oktety (24 bity)	Pozostały 1 oktet (8 bitów)

Przykłady adresów IP (część hostowa dziesiętnie albo część hostowa bitowo):

10.0.0.0 - adres całej sieci klasy A (same zera na bitach części hostowej)

10.0.1.5 - adres unicast w sieci 10.0.0.0

10.255.255.255 – adres broadcast w sieci 10.0.0.0 (same jedynki na bitach części hostowej, pakiet z takim adresem docelowym jest przeznaczony dla wszystkich stacji w danej sieci)

150.10.0.0 albo 150.10.00000000.00000000 – adres całej sieci klasy B

150.10.1.6 albo 150.10.00000001.00000110 – adres unicast w sieci 150.10.0.0

150.10.255.255 albo 150.10.11111111.11111111 – adres broadcast w sieci 150.10.0.0

213.135.45.0 albo 213.135.45.00000000 - adres całej sieci klasy C (same zera w cz. host.)

213.135.45.22 albo 213.135.45.00010110 - adres unicast w sieci 213.135.45.0

213.135.45.255 albo 213.135.45.11111111 - adres broadcast w sieci 213.135.45.0

127.0.0.0 - sieć adresów loopbackowych (sieć wyodrębniona po to, aby komputer mógł wysyłać pakiety do samego siebie bez znajomości własnego adresu IP)

127.0.0.1 - najczęściej używany adres loopbackowy

127.10.15.183 - też adres loopbackowy

127.255.255.255 - adres b-cast w sieci 127.0.0.0

Adres broadcast (rozgłoszeniowy) jest to adres pakietu przeznaczonego dla wszystkich komputerów w sieci. Są dwa rodzaje adresów broadcast – globalny i skierowany. Pierwszy z nich to adres 255.255.255.255 (32 jedynki w zapisie bitowym) i pakiet z tym adresem docelowym jest przeznaczony dla wszystkich komputerów w sieci lokalnej. Taki pakiet nie przejdzie przez żaden router. Drugi rodzaj adresu broadcast ma w części sieciowej adres sieci docelowej, a w części hostowej same jedynki (w zapisie bitowym). Posługując się broadcastem skierowanym możemy wysłać pakiet do wszystkich komputerów w innej sieci niż lokalna.

Przykład użycia broadcastu skierowanego:
Mamy sieć lokalną klasy C o adresie 213.135.45.0
Jeśli z powyższej sieci chcemy wysłać pakiet danych
do wszystkich komp. w sieci klasy C o adresie 213.135.44.0,
to wysyłamy go na adres 213.135.44.255 (broadcast skierowany do sieci 213.135.44.0)

Jeśli część sieciowa adresu ma długość inną od domyślnej (czyli 8 dla kl. A, 16 dla kl. B lub 24 dla kl. C), długość tę podaje się za pomocą maski zapisywanej łącznie z adresem.
Maska: ciąg 32 bitów dzielący adres IP na część sieciową i hostową; w części sieciowej maska ma same jedynki, a w hostowej – same zera. Liczba jedynek maski jest długością części sieciowej.

Przykłady masek:
11111111 00000000 00000000 00000000, 255.0.0.0 - maska domyślna adresu klasy A
11111111 11110000 00000000 00000000, 255.240.0.0 - maska o długości 12
11111111 11111111 00000000 00000000, 255.255.0.0 - maska domyślna adresu klasy B
11111111 11111111 11110000 00000000, 255.255.240.0 - maska o długości 20
11111111 11111111 11111111 00000000, 255.255.255.0 - maska domyślna adresu klasy C
11111111 11111111 11111111 11100000, 255.255.255.224 – maska o długości 27

Przykład adresu zapisanego łącznie z maską:
213.135.45.140, 255.255.255.128 <- postać dziesiętna adresu i maski 25-bitowej
11010101.10000111.00101101.1 0001100 <- postać binarna powyższego adresu
11111111.11111111.11111111.1 0000000 <- postać binarna powyższej maski
Część sieciowa powyższego adresu to pierwsze 25 bitów, a hostowa – ostatnie 7 bitów. Adres sieci, w której znajduje się powyższy adres unicast, otrzymujemy wpisując same zera w części hostowej. Adresem tej sieci jest więc 11010101.10000111.00101101.1 0000000, czyli 213.135.45.128. Jest to jedna z dwóch podsieci powstałych z podziału sieci klasy C o adresie 213.135.45.0 na dwie równe części.

Podział sieci IP na podsieci

Część sieciowa adresu może mieć dowolną długość, nie tylko 8, 16 albo 24 bity. Sieci o maskach innych długości powstają z podziału (albo połączenia) sieci klas A, B lub C.

Definicja sieci IP

Sieć IP jest to grupa KOLEJNYCH adresów spełniająca nast. warunki:

1. adresy są jednakowe w cz. sieciowej
2. adresy różnią się między sobą w cz. hostowej
3. pierwszy adres z grupy ma same zera w cz. hostowej (adres sieci)
4. ostatni adres z grupy ma same jedynki w cz. hostowej (adres b-cast w tej sieci)

Z trzech ostatnich warunków wynika, że liczba adresów w sieci IP musi być potęgą liczby 2. Dokładniej, jest to liczba 2^k, gdzie k to liczba bitów części hostowej.

Powyższa definicja jest równoważna następującej:

Grupa adresów jest siecią IP, jeśli spełnia nast. warunki:

1. Są to kolejne adresy
2. Ich liczba wynosi 2^k
3. Pierwszy adres (adres sieci) ma same zera na ostatnich k bitach

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Ćwiczenie 6

Podzielić sieć klasy C o adresie 213.135.45.0 na dwie równe części.

Podział sieci danej klasy na 2 części jest realizowany poprzez wydłużenie maski domyślnej o jeden bit, w wyniku czego powstają dwie grupy adresów (dwie podsieci), z których każda jest siecią IP. W przypadku sieci klasy C maska wydłuża się z 24 do 25 bitów.

Wydłużona o 1 bit maska sieci kl. C zapisana bitowo: 1--1.1--1.1--1.1 0000000

Wydłużona o 1 bit maska sieci kl. C zapisana dziesiętnie: 255.255.255.128

Kolejne adresy w 1 podsieci (ostatni oktet rozpisany na bity, w nawiasie ostatni oktet dziesiętnie):

213.135.45.0 0000000 (0) <- adres 1 podsieci

213.135.45.0 0000001 (1) <- 1-szy adres unicast w 1 podsieci

...

213.135.45.0 1111110 (126) <- ostatni adres unicast w 1 podsieci

213.135.45.0 1111111 (127) <- adres b-cast w 1 podsieci

Kolejne adresy w 2 podsieci:

213.135.45.1 0000000 (128) <- adres 2 podsieci

213.135.45.1 0000001 (129) <- 1-szy adres unicast w 2 podsieci

...

213.135.45.1 1111110 (254) <- ostatni adres unicast w 2 podsieci

213.135.45.1 1111111 (255) <- adres b-cast w 2 podsieci

Adresy podsieci zapisane z długością maski (po znaku dzielenia) dają pełną informację o podsieciach: adres 1 podsieci: 213.135.45.0/25

adres 2 podsieci: 213.135.45.128/25

Ćwiczenie 7

Podzielić sieć klasy C o adresie w.x.y.0 ($192 \leq w \leq 223$) na 4 równe części.

Podział sieci na 4 części realizujemy wydłużając maskę domyślną o dwa bity, w wyniku czego powstają cztery grupy adresów (4 podsieci), z których każda jest siecią IP. W przypadku sieci klasy C maska wydłuża się z 24 do 26 bitów.

Wydłużona maska w zapisie bitowym: 1--1.1--1.1--1.11 000000

Wydłużona maska w zapisie dziesiętnym: 255.255.255.192

Kolejne adresy w 1 podsieci (ostatni oktet rozpisany na bity):

w.x.y.00 000000 (0) <- adres 1 podsieci

w.x.y.00 000001 (1) <- pierwszy adres unicast w 1 podsieci

…

w.x.y.00 111110 (62) <- ostatni adres unicast w 1 podsieci

w.x.y.00 111111 (63) <- adres b-cast w 1 podsieci

Kolejne adresy w 2 podsieci (ostatni oktet rozpisany na bity):

w.x.y.01 000000 (64) <- adres 2 podsieci

w.x.y.01 000001 (65) <- pierwszy adres unicast w 2 podsieci

…

w.x.y.01 111110 (126) <- ostatni adres unicast w 2 podsieci

w.x.y.01 111111 (127) <- adres b-cast w 2 podsieci

Kolejne adresy w 3 podsieci (ostatni oktet rozpisany na bity):

w.x.y.10 000000 (128) <- adres 3 podsieci

w.x.y.10 000001 (129) <- pierwszy adres unicast w 3 podsieci

…

w.x.y.10 111110 (190) <- ostatni adres unicast w 3 podsieci

w.x.y.10 111111 (191) <- adres b-cast w 3 podsieci

Kolejne adresy w 4 podsieci (ostatni oktet rozpisany na bity):

w.x.y.11 000000 (192) <- adres 4 podsieci

w.x.y.11 000001 (193) <- pierwszy adres unicast w 4 podsieci

…

w.x.y.11 111110 (254) <- ostatni adres unicast w 4 podsieci

w.x.y.11 111111 (255) <- adres b-cast w 4 podsieci

Adresy kolejnych podsieci zapisane z długością maski (po znaku dzielenia):

w.x.y.0/26, w.x.y.64/26, w.x.y.128/26, w.x.y.193/26,

Wydłużenie maski o k bitów skutkuje podziałem na 2^k równych podsieci

Tabela poddziału sieci klasy C (w.x.y.0, gdzie $192 \leq w \leq 223$) na równe podsieci

Liczba podsieci	Maska	Adresy kolejnych pods.	Zakresy adresów u-cast	Adresy b-cast
2^1 = 2	255.255.255.128	w.x.y.0 w.x.y.128	w.x.y.1 - 126 w.x.y.129 - 254	w.x.y.127 w.x.y.255
2^2 = 4	255.255.255.192	w.x.y.0 w.x.y.64 w.x.y.128 w.x.y.192	w.x.y.1 – 62 w.x.y.65 – 126 w.x.y.129 – 190 w.x.y.193 - 254	w.x.y.63 w.x.y.127 w.x.y.191 w.x.y.255

Ćwiczenie 8

Utworzyć kolejne wiersze powyższej tabeli dla 8 i 16 podsieci

Ćwiczenie 9

Podzielić sieć klasy B o adresie 150.10.0.0/16 na 4 podsieci.

Wydłużamy maskę 255.255.0.0 o 2 bity, powstaje maska 255.255.192.0,

czyli 1--1.1--1.11 000000.0--0

Kolejne adresy:

150.10.0.0 150.10.00 000000.00000000

... ...

150.10.63.255 150.10.00 111111.11111111

150.10.64.0 150.10.01 000000.00000000

... ...

150.10.127.255 150.10.01 111111.11111111

150.10.128.0 150.10.10 000000.00000000

... ...

150.10.191.255 150.10.10 111111.11111111

150.10.192.0 150.10.11 000000.00000000

... ...

150.10.255.255 150.10.11 111111.11111111

Adresy kolejnych podsieci:

150.10.0.0/18

150.10.64.0/18

150.10.128.0/18

150.10.192.0/18

Zadanie 1

Podzielić sieć 200.10.5.0 na trzy jak najmniejsze podsieci w taki sposób, aby w pierwszej móc zaadresować 50, w drugiej 55, a w trzeciej 110 komputerów.

0 63 64 127 128 255

|----------|----------|----------| <- ostatnie oktety kolejnych adresów

Maska dziesiętnie	Adres podsieci	Zakres adresów unicast	Adres b-cast w podsieci	Liczba adresów unicast
255.255.255.192	200.10.5.0	200.10.5.1 – 62	200.10.5.63	62
255.255.255.192	200.10.5.64	200.10.5.65 – 126	200.10.5.127	62
255.255.255.128	200.10.5.128	200.10.5.129 – 254	200.10.5.255	126

200.10.5.00 000000 <- 1 grupa jest siecią IP (same zera w części hostowej)

200.10.5.01 000000 <- 2 grupa jest siecią IP

200.10.5.1 0000000 <- 3 grupa jest siecią IP

Łączna liczba adresów unicast = 250 (o 4 mniej niż w niepodzielonej sieci klasy C)

Adresy podsieci w zapisie z długością maski: 200.10.5.0/26, 200.10.5.64/26, 200.10.5.128/25

Zadanie 2

Czy można tak podzielić sieć 200.10.5.0 na jak najmniejsze podsieci, aby w pierwszej z nich umieścić 50 komputerów, w drugiej 110, a w trzeciej 55? Podział taki wyglądałby następująco:

0 63 64 191 192 255 <- ostatnie oktety adresów z całej sieci

|----------|----------|----------|

Wypiszmy początkowe adresy kolejnych grup adresów, zapisując binarnie ostatni oktet i wstawiając spację przed częścią hostową adresu.

200.10.5.00 000000 <- 1 grupa jest siecią IP (same zera w części hostowej)

200.10.1.0 1000000 <- 2 grupa nie jest siecią IP (jedynka w części hostowej)

200.10.1.11 000000 <- 3 grupa jest siecią IP

Kolejne adresy w drugiej grupie, w nawiasie ostatni oktet dziesiętnie:

200.10.5.0 1000000 (64)

200.10.5.0 1000001 (65)

...

200.10.5.0 1111111 (127)

200.10.5.1 0000000 (128)

...

200.10.5.1 0111110 (190)

200.10.5.1 0111111 (191)

Widać, że konsekwencją wystąpienia jedynki w pierwszym adresie drugiej grupy jest zmiana w części sieciowej przy przejściu z adresu 127 do adresu 128, co jest niedopuszczalne, bo w części sieciowej wszystkie adresy podsieci muszą być jednakowe.

Odp. Nie można podzielić sieci 200.10.5.0 w powyższy sposób, bo druga grupa adresów nie jest siecią IP.

Zadanie 3

Wydzielić z sieci 200.10.5.0 trzy jak najmniejsze podsieci, tak aby w pierwszej umieścić 100, w drugiej 25, a w trzeciej 55 komputerów.

Pierwszy potencjalny sposób podziału:

```
 0                                    127|128 159|160            223|
|-----------------------------------------|-----------|----------------------|-----------|
```

Początkowe adresy kolejnych grup z ostatnim oktetem zapisanym binarnie i spacją po części sieciowej:

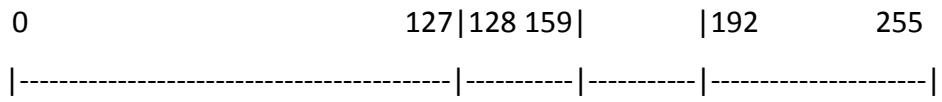
200.10.5.0 0000000 <- 1 grupa jest siecią IP (same zera w części hostowej)

200.10.5.100 00000 <- 2 grupa jest siecią IP

200.10.5.10 100000 <- 3 grupa nie jest siecią IP (jedynka w części hostowej)

Powyższy podział jest nieprawidłowy

Drugi potencjalny sposób podziału:



Początkowe adresy kolejnych grup z ostatnim oktetem zapisanym binarnie i spacją po części sieciowej:

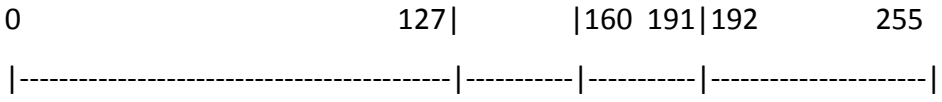
200.10.5.0 0000000 <- 1 grupa jest siecią IP

200.10.5.100 00000 <- 2 grupa jest siecią IP

200.10.5.11 000000 <- 3 grupa jest siecią IP

Powyższy podział jest prawidłowy

Trzeci potencjalny sposób podziału:



Początkowe adresy kolejnych grup z ostatnim oktetem zapisanym binarnie:

200.10.5.0 0000000 <- 1 grupa jest siecią IP

200.10.5.101 00000 <- 2 grupa jest siecią IP

200.10.5.11 000000 <- 3 grupa jest siecią IP

Powyższy podział też jest prawidłowy

Zadanie 4

Podzielić sieć 192.168.1.0 na 4 podsieci, tak aby w kolejnych podsieciach można było umieścić odpowiednio 100, 50, 25 i 20 komputerów

Jedyny możliwy sposób podziału:

```
 0                                    127|128            191|192 223|224  255
|-----------------------------------------|----------------|-----------|-----------|
```

Początkowe adresy kolejnych grup:

192.168.1.0/25 192.168.1.0 0000000 <- adres sieci IP

192.168.1.128/26 192.168.1.10 000000 <- adres sieci IP

192.168.1.192/27 192.168.1.110 00000 <- adres sieci IP

192.168.1.224/27 192.168.1.111 00000 <- adres sieci IP

Powyższy podział jest prawidłowy

Zadanie 5

Podzielić sieć 192.168.1.0 na 4 podsieci, tak aby w kolejnych podsieciach można było umieścić odpowiednio 100, 25, 50 i 20 komputerów

Jedyny możliwy sposób podziału:

```
 0                                    127|128 159|160           223|224  255|
|-----------------------------------------|----------|----------------|-----------|
```

Początkowe adresy kolejnych grup z ostatnim oktetem zapisanym binarnie:

192.168.1.0 0000000 <- adres sieci IP

192.168.1.100 00000 <- adres sieci IP

192.168.1.10 100000 <- to nie jest adres sieci IP (jedynka w części hostowej)

192.168.1.111 00000 <- adres sieci IP

Powyższe zadanie nie ma rozwiązania, bo trzecia grupa adresów nie jest siecią IP.

Zadanie 6

Podzielić sieć klasy B o adresie 150.20.0.0 na 4 kolejne podsieci o rozmiarach 2^15, 2^14, 2^13 i 2^13.

Kolejne adresy sieci 150.20.0.0, binarnie na ostatnich dwóch oktetach, w podziale na podsieci:	Kolejne adresy sieci 150.20.0.0 dziesiętnie z długością maski po znaku dzielenia:
150.20.0 0000000.00000000	150.20.0.0/17
…	
150.20.0 1111111.11111111	

150.20.10 000000.00000000	150.20.128.0/18
…	
150.20.10 111111.11111111	

150.20.110 00000.00000000	150.20.192.0/19
…	
150.20.110 11111.11111111	

150.20.111 00000.00000000	150.20.224.0/19
…	
150.20.111 11111.11111111	

Wszystkie powyższe grupy są sieciami IP, bo początkowe adresy tych grup mają same zera w częściach hostowych.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

rozmiar sieci = 2^k, gdzie k to długość części hostowej

długość części hostowej = 32 – długość maski

długość maski (liczba jedynek) = długość cz-ci sieciowej = 32 – długość cz-ci hostowej

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Jeśli liczba adresów w grupie jest potęgą dwójki, są to kolejne adresy,

a pierwszy adres grupy składa się w części hostowej z samych zer,

to taka grupa adresów jest siecią IP.

W przeciwnym przypadku grupa adresów nie jest siecią IP.

xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

Sieci można nie tylko dzielić, ale też łączyć. Jest to jednak odejście od zasady podziału na klasy mówiącej, że maksymalny rozmiar sieci wynika z pierwszego oktetu adresu IP (ten rozmiar to 256, jeśli sieć jest klasy C, czyli pierwszy oktet adresu sieci jest liczbą z przedziału 192-223). Skrócenie maski o k bitów skutkuje połączeniem 2^k równych rozmiarem sieci w jedna grupę adresów.

Zadanie 7

Czy można połączyć w sieć IP następujące 4 sieci klasy C?

200.10.0.0, 200.10.1.0, 200.10.2.0, 200.10.3.0

Grupa adresów utworzona z powyższych sieci to $4*2^8 = 2^{10}$ kolejnych adresów, więc są spełnione 2 pierwsze kryteria dla sieci IP. Trzeba jeszcze sprawdzić, czy 200.10.0.0 (pierwszy adres grupy) ma w zapisie binarnym same zera w części hostowej. Zapiszmy binarnie dwa ostatnie oktety tego adresu ze spacją między częścią sieciową (22 bity) i hostową (10 bitów):

200.10.000000 00.00000000

Pierwszy adres grupy składa się z samych zer w części hostowej, więc powyższa agregacja jest możliwa. W jej wyniku powstaje sieć 200.10.0.0/22

Zadanie 8

Czy można połączyć w sieć IP następujące 4 sieci klasy C:

200.10.6.0, 200.10.7.0, 200.10.8.0, 200.10.9.0

Grupa adresów utworzona z powyższych sieci to 2^10 kolejnych adresów, więc są spełnione 2 pierwsze kryteria dla sieci IP. Trzeba jeszcze sprawdzić, czy pierwszy adres grupy, czyli 200.10.6.0 ma w zapisie binarnym same zera w części hostowej. Zapiszmy binarnie dwa ostatnie oktety tego adresu wstawiając spację między część sieciową (22 bity) i hostową (10 bitów):

200.10.000001 10.00000000

W pierwszym adresie grupy jest jedynka w części hostowej, więc powyższa agregacja nie jest możliwa.

Zadanie 9

Czy następujące sieci klasy C można połączyć w sieć IP?

200.10.4.0, 200.10.5.0, 200.10.6.0, 200.10.7.0, 200.10.8.0, 200.10.9.0

W powyższych sieciach jest łącznie 6*2^8 = 3^2*9 adresów, więc ich liczba nie jest potęgą dwójki. W związku z tym nie można tych sieci połączyć w sieć IP, bo rozmiar sieci IP musi być potęgą dwójki.

Zadanie 10

Czy następujące sieci klasy C można połączyć w sieć IP?

200.10.4.0, 200.10.5.0, 200.10.8.0, 200.10.9.0

Nie można, bo powyższe adresy nie są kolejne, po adresie 200.10.5.255 (ostatni w 2 sieci) powinien być adres 200.10.6.0, a nie 200.10.8.0.

Zadanie 11

Czy blok adresów 200.10.5.0/21 jest siecią IP?

Ponieważ maska w powyższym bloku ma długość 21, więc część hostowa ma długość 11.

Pierwszy adres powyższego bloku (dwa ostatnie oktety binarnie i spacja przed częścią hostową):

200.10.00000 101.00000000

W tym adresie są jedynki w części hostowej, więc blok nie jest siecią IP.

Wysyłanie danych z sieci IP

Przed wysłaniem danych stacja źródłowa sprawdza, czy adresat znajduje się w sieci lokalnej, czy w innej. W tym celu wykonuje operację nałożenia własnej maski na docelowy adres IP.

Jeśli wynik nałożenia JEST adresem sieci lokalnej, to adresat znajduje się w tej sieci.

Jeśli wynik NIE JEST adresem sieci lokalnej, to adresat znajduje się w innej sieci.

Adres sieci lokalnej jest wynikiem nałożenia własnej maski na własny adres IP.

Nałożenie maski na adres polega na pomnożeniu kolejnych bitów adresu przez kolejne bity maski.

Przykład 1: Dane wysyła host o adresie 200.10.1.1/26

Ustalenie adresu sieci lokalnej:

Własny IP: 200. 10. 1.00000001 (1)

Własna maska: 255.255.255.11000000 (192)

IP sieci lokalnej: 200. 10. 1.00000000 (0)

Przypadek 1. Host wysyła dane do 200.10.1.2

Docelowy IP: 200. 10. 1.00000010 (2)

Własna maska: 255.255.255.11000000 (192)

Wynik nałożenia: 200. 10. 1.00000000 (0) <- IP sieci lokalnej

Przypadek 2. Host wysyła dane do 200.10.1.129

Docelowy IP: 200. 10. 1.10 000001 (129)

Własna maska: 255.255.255.11 000000 (192)

Wynik nałożenia: 200. 10. 1.10000000 (128) <- IP inny niż sieci lokalnej

Uwaga: Jeśli adresat nie jest w sieci lokalnej, to wynikiem nałożenia własnej maski na docelowy adres IP nie musi być adres sieci docelowej.

Przykład 2: Dane wysyła 200.10.1.129/25 do 200.10.1.100/26

Własny IP: 200. 10. 1.10000001 (129)
Własna maska: 255.255.255.10000000 (128)

Wynik nałożenia: 200. 10. 1.10000000 (128) <- IP sieci lokalnej

Docelowy IP: 200. 10. 1.01100100 (100)
Własna maska: 255.255.255.10000000 (128)

Wynik nałożenia: 200. 10. 1.00000000 (0) <- IP inny niż sieci lokalnej, ale nie IP sieci docelowej, którym jest 200.10.1.64 uzyskany z nałożenia maski o długość 26 na adres stacji docelowej.

Jeśli adresat jest w sieci lokalnej, to komputer źródłowy uzyskuje jego adres MAC (z tablicy ARP albo uruchamiając protokół ARP) i wysyła dane bezpośrednio do adresata.

Jeśli adresat jest poza siecią lokalną, to komputer źródłowy uzyskuje adres MAC routera (z tablicy ARP albo uruchamiając protokół ARP) i wysyła dane do routera. Adres IP routera, oprócz adresu IP i maski danej stacji, jest jednym z 3 podstawowych parametrów konfiguracyjnych protokołu IP.

Przekazywanie danych między sieciami IP (trasowanie, routing)

Po odebraniu pakietu danych, router wysyła go albo bezpośrednio do adresata, jeśli jest on w jednej z sieci przyłączonych do routera, albo do jednego z routerów sąsiednich, czyli do następnego routera. W tym celu router sprawdza, który wiersz tabeli trasowania jest zgodny z adresem docelowym pakietu.

Tabela trasowania R1 (A to i-fejs z lewej, B to i-fejs z prawej):

IP sieci docelowej	Maska sieci docelowej	IP następnego routera	Przykładowa nazwa i-fejsu wyjściowego w syst. oper. routera
200.10.1.0	255.255.255.192	0.0.0.0	R1_A
200.10.1.64	255.255.255.192	0.0.0.0	R1_B
200.10.1.128	255.255.255.128	200.10.1.126	R1_B
0.0.0.0	0.0.0.0	200.10.1.125	R1_B

Pierwsze wiersze w tabeli routingu określają trasy do sieci przyłączonych do routera. W tych wierszach adres następnego routera składa się z samych zer, co oznacza że router ma wysłać pakiet bezpośrednio do stacji docelowej. Wiersz, w którym IP sieci docelowej i jej maska składają się z samych zer, określa tzw. trasę domyślną, na którą kierowany jest pakiet z adresem docelowym niezgodnym z żadnym innym wierszem.

Adres docelowy jest zgodny z wierszem tabeli trasowania, jeśli nałożenie maski z tego wiersza na adres docelowy daje w wyniku IP sieci docelowej z tego wiersza. Proces decyzyjny zachodzący w routerze polega na sprawdzaniu zgodności adresu docelowego z kolejnymi wierszami tabeli, a następnie, po znalezieniu właściwego wiersza, wysłaniu pakietu tak jak określa to 3 i 4 kolumna tego wiersza.

Przykład 1:

R1 odbiera dane z IP docelowym 200.10.1.129, następnie sprawdza zgodność tego adresu z kolejnymi wierszami w tabeli routingu:

Docelowy IP: 200. 10. 1.10 000001

Maska z 1 wiersza: 255.255.255.11 000000

Wynik nałożenia: 200. 10. 1.10 000000 (128); niezgodny z 1 wierszem

Docelowy IP: 200. 10. 1.10 000001

Maska z 2 wiersza: 255.255.255.11 000000

Wynik nałożenia: 200. 10. 1.10 000000 (128); niezgodny z 2 wierszem

Docelowy IP: 200. 10. 1.1 0000001

Maska z 3 wiersza: 255.255.255.1 0000000

Wynik nałożenia: 200. 10. 1.1 0000000 (128); zgodny z 3 wierszem

Po ustaleniu, że adres docelowy jest zgodny z 3 wierszem, R1 uzyskuje adres MAC interfejsu 200.10.1.126 (IP następnego routera) i na ten MAC wysyła dane z interfejsu R1_B

Przykład 2:

R1 odbiera dane z IP docelowym 200.10.1.100

Docelowy IP: 200. 10. 1.01100100

Maska z 1 wiersza: 255.255.255.11000000

Wynik nałożenia: 200. 10. 1.01000000 (64); niezgodny z 1 wierszem

Docelowy IP: 200. 10. 1.01100100

Maska z 2 wiersza: 255.255.255.11000000

Wynik nałożenia: 200. 10. 1.01000000 (64); zgodny z 2 wierszem

Po ustaleniu, że adres docelowy jest zgodny z 2 wierszem, R1 wysyła dane bezpośrednio do komputera docelowego (brak następnego routera), czyli R1 uzyskuje adres MAC interfejsu 200.10.1.100 (docelowy adres IP) i na ten MAC wysyła dane z interfejsu R1_B.

W przykładzie 1 adresat nie jest w sieci przyłączonej do R1, więc R1 wysyła dane do następnego routera, którego IP odczytuje z 3 kolumny. W przykładzie 2 adresat jest w sieci przyłączonej do R1, więc R1 wysyła dane bezpośrednio do niego, nie korzystając z następnego routera.

Uwaga: Adres docelowy 200.10.1.129 pasuje do 3 wiersza, ale pasuje też do wiersza 4. Wiersz, w którym adres sieci docelowej i jej maska składają się z samych zer jest nazywany trasą domyślną (określenie "trasa" jest nieścisłe, bo wiersz tabeli trasowania nie opisuje całej trasy, a tylko najbliższy etap trasy). Zgodnie z 3 wierszem dane należy wysłać do R2, a zgodnie z 4 wierszem - do R3.

W przypadku, gdy adres docelowy pasuje do więcej niż jednego wiersza, wybierany jest wiersz o dłuższej masce.

Zgodnie z powyższą zasadą, na trasę domyślną są kierowane dane z adresem docelowym nie pasującym do żadnego innego wiersza z maską niezerową, bo maska trasy domyślnej ma długość zero (zero jedynek), więc jest krótsza od każdej innej maski.

Tabela trasowania R2 (A to i-fejs z lewej, B to i-fejs z prawej):

IP sieci docelowej	Maska sieci docelowej	IP następnego routera	Przykładowa nazwa i-fejsu wyjściowego w syst. oper. routera
200.10.1.128	255.255.255.128	0.0.0.0	R2_B
200.10.1.64	255.255.255.192	0.0.0.0	R2_A
200.10.1.0	255.255.255.192	200.10.1.124	R2_A
0.0.0.0	0.0.0.0	200.10.1.125	R2_A

Tabela trasowania R3 (niepełne dane, A to i-fejs górny, B to i-fejs dolny):

IP sieci docelowej	Maska sieci docelowej	IP następnego routera	Przykładowa nazwa i-fejsu wyjściowego w syst. oper. routera
?	?	0.0.0.0	R3_B
200.10.1.64	255.255.255.192	0.0.0.0	R3_A
200.10.1.0	255.255.255.192	200.10.1.124	R3_A
200.10.1.128	255.255.255.128	200.10.1.126	R3_A
0.0.0.0	0.0.0.0	?	R3_B

Tabele routingu mają nie tylko routery, mają je również hosty, ale w tabelach trasowania hostów jest zapisana informacja jak dane wysyłać, a nie jak je przekazywać.

Jeśli w sieci jest tylko jeden router (domyślny), to w tabeli routingu hosta są dwa wiersze.

Pierwszy to trasa do sieci lokalnej, a drugi - trasa domyślna.

Załóżmy, że host o adresie IP 200.10.1.1 ma jeden interfejs o nazwie eth0.

Tabela trasowania tego hosta:

IP sieci docelowej	Maska sieci docelowej	IP routera	Przykładowa nazwa i-fejsu wyjściowego w syst. oper. hosta
200.10.1.0	255.255.255.192	0.0.0.0	eth0
0.0.0.0	0.0.0.0	200.10.1.62	eth0

Jeśli w sieci jest więcej niż 1 router, to w tabeli routingu hosta mogą być więcej niż 2 wiersze, bo host może kierować dane do różnych routerów zależnie od IP docelowego.

Załóżmy, że host o adresie 200.10.1.100 znajdujący się w sieci 200.10.1.64/26 ma jeden interfejs o nazwie eth0.

Tabela trasowania tego hosta:

IP sieci docelowej	Maska sieci docelowej	IP routera	Przykładowa nazwa i-fejsu wyjściowego w syst. oper. hosta
200.10.1.64	225.225.225.192	0.0.0.0	eth0
200.10.1.0	225.225.225.192	200.10.1.124	eth0
200.10.1.128	225.225.225.128	200.10.1.126	eth0
0.0.0.0	0.0.0.0	200.10.1.125	eth0

Dzielimy sieć klasy C o adresie 192.168.1.0 na 4 podsieci:

Adres podsieci/maska	Zakres adresów unicast
192.168.1.0/25	od 1 do 126 na ostatnim oktecie
192.168.1.128/26	od 129 do 190 na ostatnim oktecie
192.168.1.192/27	od 193 do 222 na ostatnim oktecie
192.168.1.224/27	od 225 do 254 na ostatnim oktecie

Przypisanie numerów na liście obecności do poszczególnych komputerów przy założeniu, że w grupie jest 15 osób:

Hosty w 1 podsieci: 1, 2, 3

R1: 4

Hosty w 2 podsieci: 5, 6, 7

R2: 8

Hosty w 3 podsieci: 9, 10, 11

R3: 12

Hosty w 4 podsieci: 13, 14, 15

Polecenia konfiguracji interfejsów na hostach i routerach:

Na hostach w 1 podsieci 192.168.1.0/25:

ip addr add 192.168.1.x/25 dev enp3s0

gdzie x jest liczbą od 1 do 3

Na R1:

ip addr add 192.168.1.126/25 dev enp3s0

ip addr add 192.168.1.188/26 dev enp3s0

Na hostach w 2 podsieci 192.168.1.128/26:

ip addr add 192.168.1.x/26 dev enp3s0

gdzie x jest liczbą od 129 do 131

Na R2:

ip addr add 192.168.1.190/26 dev enp3s0

ip addr add 192.168.1.222/27 dev enp3s0

Na hostach w 3 podsieci 192.168.1.192/27:

ip addr add 192.168.1.x/27 dev enp3s0

gdzie x jest liczbą od 193 do 195

Na R3:

ip addr add 192.168.1.189/26 dev enp3s0

ip addr add 192.168.1.254/27 dev enp3s0

Na hostach w 4 podsieci 192.168.1.224/27:

ip addr add 192.168.1.x/27 dev enp3s0

gdzie x jest liczbą od 225 do 227

Polecenia konfiguracji tabel trasowania (dodawania wierszy do tabel):

Tabele trasowania na routerach:

Na R1:

ip route show <- wypisuje tabelę trasowania

ip route add 192.168.1.192/27 via 192.168.1.190 dev enp3s0

ip route add 192.168.1.224/27 via 192.168.1.189 dev enp3s0

ip route show <- sprawdza poprawność tabeli

Na R2:

ip route show <- wypisuje tabelę trasowania

ip route add 192.168.1.0/25 via 192.168.1.188 dev enp3s0

ip route add 192.168.1.224/27 via 192.168.1.189 dev enp3s0

ip route show <- sprawdza poprawność tabeli

Na R3:

ip route show <- wypisuje tabelę trasowania

ip route add 192.168.1.0/25 via 192.168.1.188 dev enp3s0

ip route add 192.168.1.192/27 via 192.168.1.190 dev enp3s0

ip route show <- sprawdza poprawność tabeli

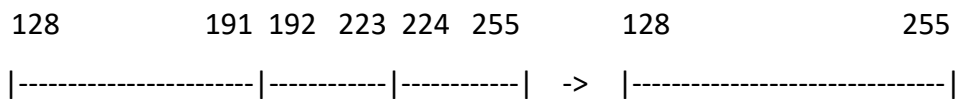
Tabele trasowania na hostach:

Na wszystkich hostach w podsieci 192.168.1.0/25:

ip route add 192.168.1.128/25 via 192.168.1.126 dev enp3s0

ip route show <- sprawdzenie poprawności

Uwaga: w powyższym poleceniu wykorzystano możliwość połączenia sieci 192.168.1.128/26, 192.168.1.192/27 i 192.168.1.224/27 w jedną sieć 192.168.1.128/25 (do każdej z nich prowadzi ten sam router), czyli trzy wiersze w tabeli trasowania zagregowano w jeden wiersz



Na wszystkich hostach w podsieci 192.168.1.128/26:

ip route add 192.168.1.0/25 via 192.168.1.188 dev enp3s0

ip route add 192.168.1.192/27 via 192.168.1.190 dev enp3s0

ip route add 192.168.1.224/27 via 192.168.1.189 dev enp3s0

ip route show <- sprawdzenie poprawności

Na wszystkich hostach w podsieci 192.168.1.192/27:

ip route add 192.168.1.0/25 via 192.168.1.222 dev enp3s0

ip route add 192.168.1.128/26 via 192.168.1.222 dev enp3s0

ip route add 192.168.1.224/27 via 192.168.1.222 dev enp3s0

ip route show <- sprawdzenie poprawności

Na wszystkich hostach w podsieci 192.168.1.224/27:

ip route add 192.168.1.0/25 via 192.168.1.254 dev enp3s0

ip route add 192.168.1.128/26 via 192.168.1.254 dev enp3s0

ip route add 192.168.1.192/27 via 192.168.1.254 dev enp3s0

ip route show <- sprawdzenie poprawnościnet.ipv4.

Dodatkowe 4 polecenia wydawane tylko na R1, R2, R3:

sysctl -w net.ipv4.ip_forward=1 <- włączenie trasowania w syst. Linux

systemctl stop iptables <- wyłączenie zapory sieciowej (firewall) w syst. Linux

sysctl –w net.ipv4.conf.enp3s0.send_redirects=0 (enp3s0 jest nazwą interfejsu)

sysctl –w net.ipv4.conf.all.send_redirects=0 (all oznacza wszystkie interfejsy)

Ostatnie dwa polecenia wyłączają wysyłanie przez routery komunikatów „redirect”. Router wykrywa (za pomocą protokołu ARP), że komputer docelowy jest w tej samej sieci fizycznej co komputer źródłowy, a następnie informuje tym komunikatem komputer źródłowy o możliwości bezpośredniej komunikacji z komputerem docelowym bez konieczności angażowania routera.

Budowa ramki Ethernet II

Preambuła + znacznik początku ramki: 7 + 1 = 8 oktetów (ang. preamble, start-of-frame delimiter)

oktet preambuły: 10101010

oktet znacznika: 10101011

Docelowy MAC: 6 oktetów

Źródłowy MAC: 6 oktetów

Typ: 2 oktety określające protokół przenoszony w ramce zaraz za jej nagłówkiem

Uwaga: w protokole Ethernet II wartość pola Typ jest większa od 1536 (0x600)

Dane przenoszone w ramce: 46 - 1500 oktetów

Suma kontrolna: 4 oktety (ang. Frame Control Sequence)

S.K. jest liczona z pól adresowych, pola typ i pola danych

Odstęp czasowy między ramkami: czas wysyłania 12 oktetów (ang. Inter-Frame Gap)

pr. + zn. MAC d. MAC ź. t. Dane SK odstęp

- - - - - - - - | - - - - - - | - - - - - - | - - | - - - . . . - - - | - - - - | - - - - - - - - - - - - |

Zadanie 1:

Obliczyć efektywną przepustowość segmentu sieci Ethernet o przepustowości nominalnej 100 Mb/s, jeśli przesyłane są w nim ramki o długości pola danych równej 500 B?

przep. efekt. = (500 / łączna liczba bajtów potrzebnych do przesłania 500 B danych)*100 Mb/s

przep. efekt. = (500 / 538)*100 Mb/s ≈ 0,93*100 Mb/s = 93 Mb/s

Zadanie 2:

Jaka jest efektywna przepustowość segmentu sieci Ethernet o przepustowości nominalnej 100 Mb/s, jeśli przesyłane są w nim ramki o największej dopuszczalnej długości pola danych?

przep. efekt. = (1500 / liczba bajtów potrzebnych do przesłania 1500 B)*100 Mb/s

przep. efekt. = (1500 / 1538)*100 Mb/s ≈ 0,975*100 Mb/s = 97,5 Mb/s

Zadanie 3:

Policzyć algorytmem CRC sumę kontrolną ciągu 101010010110

przy zastosowaniu klucza generującego 1011

Ciągowi 101011000110 odpowiada wielomian x^11 + x^9 + x^7 + x^4 + x^2 + x (dzielna)

Kluczowi 1011 odpowiada wielomian x^3 + x + 1 (dzielnik)

Suma kontrolna to reszta z dzielenia wielomianów modulo 2:

x^8 + x^5 + x^4 + x^3 + x +1

---

x^11 + x^9 + x^7 + x^4 + x^2 + x : x^3 + x + 1

x^11 + x^9 + x^8

---

x^8 + x^7 + x^4 + x^2 + x

x^8 + x^6 + x^5

---

x^7 + x^6 + x^5 + x^4 + x^2 + x

x^7 + x^5 + x^4

---

x^6 + x^2 + x

x^6 + x^4 + x^3

---

x^4 + x^3 + x^2 + x

x^4 + x^2 + x

---

x^3

x^3 + x + 1

---

x + 1

Wielomianowi x+1 odpowiada suma kontrolna 011 (suma kontrolna ma 3 bity – o 1 bit mniej niż klucz).

Liczba bitów sumy kontrolnej = Liczba bitów klucza – 1

Powyższa reguła wymaga wypisania wiodących zer sumy kontrolnej, jeśli takie występują. Zgodnie z tą regułą liczba bitów klucza stosowanego do liczenia FCS w ramce Ethernet II wynosi 33.

Zadanie 4

Obliczyć algorytmem CRC sumę kontrolną ciągu 1100101101 przy zastosowaniu klucza 1001

Wielomian dla ciągu: x^9 + x^8 + x^5 + x^3 + x^2 + 1

Wielomian dla klucza: x^3 + 1

x^6 + x^5 + x^3

---

x^9 + x^8 + x^5 + x^3 + x^2 + 1 : x^3 + 1

x^9 + x^6

---

x^8 + x^6 + x^5 + x^3 + x^2 + 1

x^8 + x^5

---

x^6 + x^3 + x^2 + 1

x^6 + x^3

---

x^2 + 1

Wielomianowi x^2 + 1 odpowiada suma kontrolna 101

Zadanie 5

Policzyć algorytmem CRC sumę kontrolną ciągu 101011000110 przy zastosowaniu klucza generującego 1011

ciągowi 101011000110 odpowiada wielomian x^11 + x^9 + x^7 + x^6 + x^2 + x (dzielna)

kluczowi 1010 odpowiada wielomian x^3 + x + 1 (dzielnik)

operacja dzielenia wielomianów modulo 2:

x^8 + x^5 + x^4 + x

---

x^11 + x^9 + x^7 + x^6 + x^2 + x : x^3 + x + 1

x^11 + x^9 + x^8

---

x^8 + x^7 + x^6 + x^2 + x

x^8 + x^6 + x^5

---

x^7 + x^5 + x^2 + x

x^7 + x^5 + x^4

---

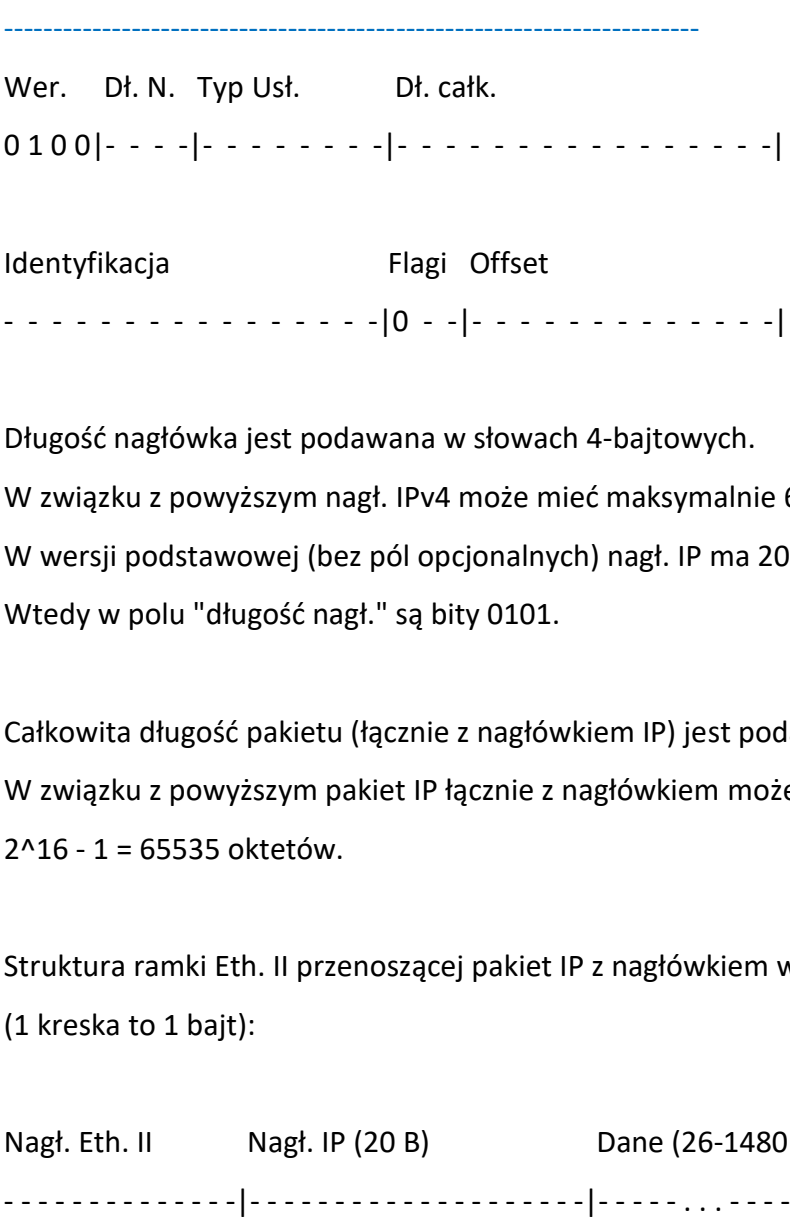
x^4 + x^2 + x

x^4 + x^2 + x

0

Wielomianowi 0 odpowiada suma kontrolna 000.

Dwa pierwsze słowa nagłówka IPv4 (1 kreska to 1 bit)



Długość nagłówka jest podawana w słowach 4-bajtowych.
W związku z powyższym nagł. IPv4 może mieć maksymalnie 60 oktetów (15*4 = 60).
W wersji podstawowej (bez pól opcjonalnych) nagł. IP ma 20 oktetów (5*4 = 20).
Wtedy w polu "długość nagł." są bity 0101.

Całkowita długość pakietu (łącznie z nagłówkiem IP) jest podawana w bajtach.
W związku z powyższym pakiet IP łącznie z nagłówkiem może mieć maksymalnie
2^16 - 1 = 65535 oktetów.

Struktura ramki Eth. II przenoszącej pakiet IP z nagłówkiem w podstawowej wersji
(1 kreska to 1 bajt):

Jeśli pakiet IP nie mieści się w jednej ramce Eth. II, to jest dzielony na fragmenty i każdy fragment jest przesyłany w osobnej ramce. Informacje dotyczące fragmentacji znajdują się w 2 słowie nagłówka IP. Składa się ono z 3 pól: 16-bitowe pole Identyfikacja, 3-bitowe pole flag, oraz 13-bitowe pole Offset.

W pole Identyfikacja każdego fragmentu jest wpisywana ta sama wartość, przepisana z tego pola całego pakietu. Pozwala to stacji docelowej rozpoznać fragmenty pochodzące z tego samego pakietu.

Flagi: pierwsza to 0, druga to DF (don't fragment), trzecia to MF (more fragments)

Router znajdujący się na trasie pakietu może łączyć sieci o różnych MTU. Jeśli pakiet ma być przekazany z sieci o większym MTU do sieci o mniejszym MTU, to może być konieczna jego fragmentacja. Jeśli DF=1, to router nie przekaże dalej pakietu (bo nie zezwala na to wartość flagi DF), a do jego nadawcy wyśle odpowiedni komunikat ICMP.
MTU - maximum transfer unit, maksymalna długość (w bajtach) pola danych ramki protokołu realizującego komunikację wewnątrzsieciową (MTU=1500 dla Ethernet II)

MF=1: dany fragment nie jest ostatni
MF=0: dany fragment jest ostatni

Offset: odsunięcie pola danych fragmentu od początku pola danych całego pakietu
Uwaga: ze względu na długość pola Offset (13 bitów), offset jest podawany w słowach 8-bajtowych!!!
(największa liczba, którą można zapisać na 13 bitach to 2^13 - 1 = 8191)
Z tego względu liczba bajtów pola danych każdego fragmentu (z wyjątkiem ostatniego) musi być wielokrotnością ośmiu.

Zadanie 4:

W sieć Ethernet wysyłany jest pakiet IP z nagłówkiem o dług. 20 B i polem danych o dług. 5600 B. Przedstawić powstałe fragmenty zakładając, że mają być jak najdłuższe.

Cały pakiet:

$\lfloor x \rfloor$ = największa liczba całkowita mniejsza lub równa x

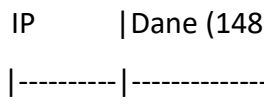
$\lceil x \rceil$ = najmniejsza liczba całkowita większa lub równa x

maks. dług. pola danych fragmentu = $8 \cdot \lfloor (MTU - 20) / 8 \rfloor = 8 \cdot 185 = 1480$

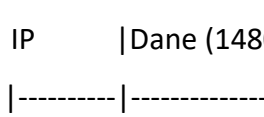
liczba fragmentów = $\lceil$ dług. pola danych pakietu / maks. dług. pola danych fragmentu $\rceil$ =

$= \lceil 5600 / 1480 \rceil = 4$

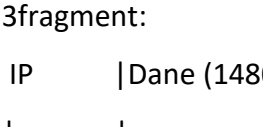
1 fragment:



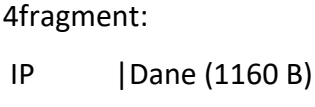
2 fragment:



3fragment:



4fragment:



Powyższe fragmenty w notacji „długość pola danych w bajtach @ offset w bajtach MF/LF”:

1480 @ 0 MF

1480 @ 1480 MF

1480 @ 2960 MF

1160 @ 4440 LF

Zadanie 5:

Zapisać w postaci bitowej pole Offset trzeciego fragmentu z poprzedniego zadania.

Wartość w polu offset dziesiętnie: 2960 / 8 = 370

370

2^8 = 256

370 – 256 = 114

2^6 = 64

114 – 64 = 50

2^5 = 32

50 – 32 = 18

2^4 = 16

18 – 16 = 2

2^1 = 2

2 – 2 = 0

Wartość w polu offset binarnie: 101110010

Zawartość pola offset: 0000101110010 (13 bitów)

Prostszy technicznie algorytm zamiany liczb dziesiętnych na binarne (kolejne reszty z dzielenia przez 2 zapisywane od prawej do lewej):

370 / 2 = 185 r 0

185 / 2 = 92 r 1

92 / 2 = 46 r 0

46 / 2 = 23 r 0

23 / 2 = 11 r 1

11 / 2 = 5 r 1

5 / 2 = 2 r 1

2 / 2 = 1 r 0

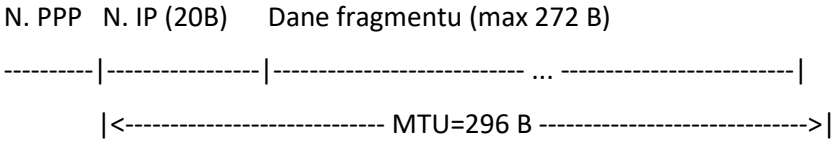
1 / 2 = 0 r 1

Wartość w polu offset binarnie: 101110010 (kolejne reszty od ostatniej do pierwszej)

Zawartość pola offset: 0000101110010 (13 bitów)

Zadanie 1:

W łącze obsługiwane przez protokół PPP (MTU=296) wysyłany jest pakiet IP z nagłówkiem bez pola opcji (nagłówek ma 20 B) i polem danych o dług. 1000 B. Przedstawić na rysunku i w standardowym zapisie powstałe fragmenty przy założeniu, że wszystkie oprócz ostatniego mają maksymalną długość.



Nieprzekraczalna długość fragmentu łącznie z nagł. IP = 296 B

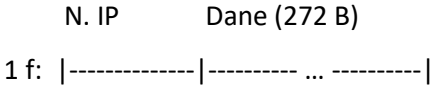
Nieprzekraczalna długość pola danych fragmentu = 276 B

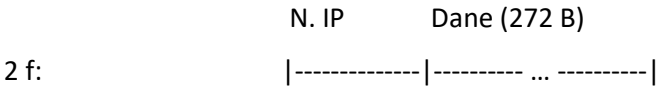
Maksymalna długość pola danych fragmentu = 272 (276 nie jest całkowitą wielokrotnością 8)

Liczba fragmentów = $\lceil 1000 / 272 \rceil = 4$

Ilustracja fragmentacji:

Cały pakiet:





Równanie, z którego wyliczamy długość pola danych ostatniego fragmentu:

184 = 1000 – 3*272

Powyższe fragmenty zapisane w standardowej notacji:

(długość_pola_danych_w_B @ offset_w_B MF/LF)

272 @ 0 MF

272 @ 272 MF

272 @ 544 MF

184 @ 816 LF

Zadanie 2

Wypisać zawartość pola Offset trzeciego fragmentu z Zad. 1

W polu Offset będzie liczba 544/8 = 68 zapisana na 13 bitach

Zamiana liczby 68 na postać binarną:

68

2^6 = 64, 68 – 64 = 4

2^2 = 4, 4 – 4 = 0

$(68)_{10} = (1000100)_2$

Pole Offset 3 fragmentu: 0000001000100 (sześć wiodących zer jest uzupełnieniem do 13 bitów)

Zadanie 3

W łącze telekomunikacyjne obsługiwane przez protokół X.25 (MTU = 576) wysyłany jest segment TCP z 20-bajtowym nagłówkiem i polem danych o dług. 2100 B. Segment ten znajduje się w pakiecie IP z 20-bajtowym nagłówkiem. Przedstawić na rysunku i w standardowym zapisie powstałe fragmenty przy założeniu, że wszystkie oprócz ostatniego mają maksymalną długość.

Za nagłówkiem TCP jest 2100 B

Za nagłówkiem IP jest 2120 B

Pole danych całego (niepofragmentowanego) pakietu ma 2120 B (tyle bajtów jest za nagł. IP)

Nieprzekraczalna długość fragmentu (razem z nagł. IP i TCP): 576 B

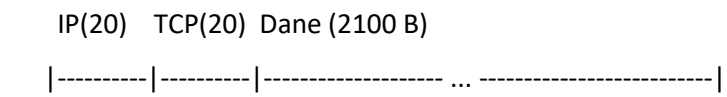
Nieprzekraczalna długość pola danych fragmentu: 556 B (nagł. IP ma 20 B)

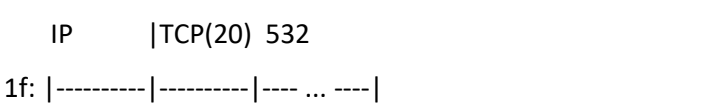
Maksymalna długość pola danych fragmentu: 552 B (556 nie dzieli się bez reszty przez 8)

Liczba fragmentów = $\lceil 2120 / 552 \rceil = 4$

Ilustracja fragmentacji:

Cały pakiet:





równanie, z którego wyliczamy długość pola danych ostatniego fragmentu:

464 = 2120 – 3*552

Zapis powyższych fragmentów w standardowej notacji:

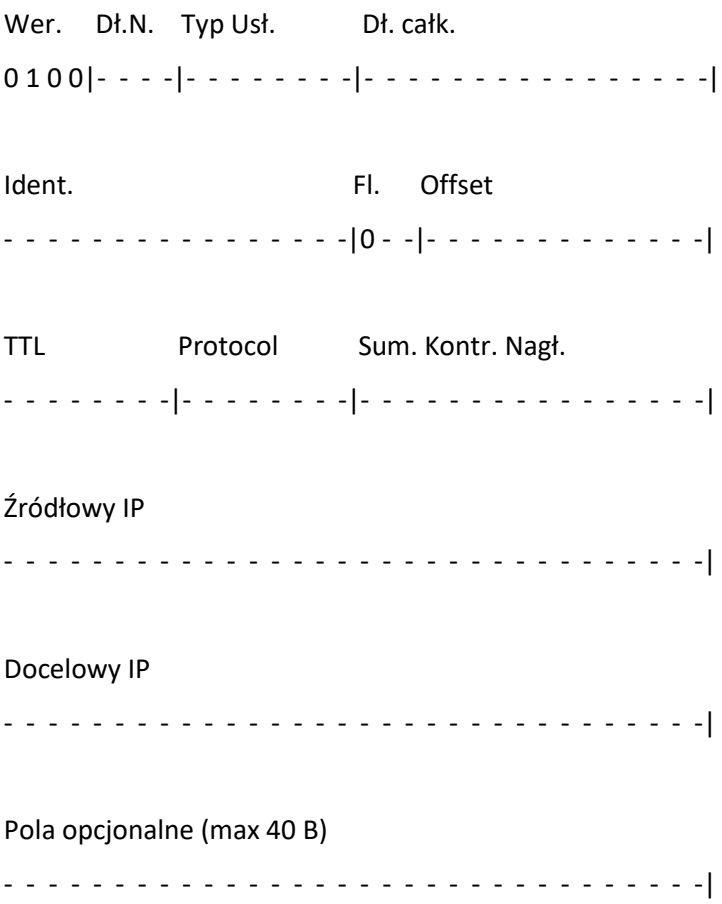
552 @ 0 MF

552 @ 552 MF

552 @ 1104 MF

464 @ 1656 LF

Budowa nagłówka IPv4 (1 kreska to 1 bit)



Każdy router na trasie pakietu zmniejsza wartość TTL o jeden i sprawdza czy TTL=0. Jeśli tak, to pakiet jest odrzucany, a router wysyła do jego nadawcy komunikat ICMP o przekroczeniu czasu życia pakietu (TTL exceeded).

Polecenie traceroute (Linux) albo tracert (Windows)

1. Wykorzystuje pole TTL nagłówka IP

2. Generuje porcje pakietów (domyślnie datagramów UDP z wysokim portem docelowym) z tą samą początkową wartością TTL, począwszy od 1

3. Domyślnie traceroute wysyła po 3 pakiety z tym samym TTL. W systemie Linux można to zmienić opcją –q (traceroute -q5 wysyła po 5 pakietów z tym samym TTL)

4. Pakiet z TTL=k nie przechodzi przez k-ty router na trasie, który odsyła komunikat o wyzerowaniu TTL. Komunikat jest odsyłany z tego interfejsu routera, przez który pakiet do routera wpłynął.

5. Komputer docelowy odsyła komunikat o niedziałającej aplikacji (z dużym prawdopodobieństwem na tzw "wysokim porcie" nie działa serwer żadnej aplikacji)

6. Czasy podawane przez traceroute są czasami RTT (ang. round trip time) do i od kolejnych routerów, oraz do i od komp. docelowego.

Przykład działania traceroute:

output z polecenia traceroute tvp.pl wydanego na komputerze o adresie 213.134.45.22:

```
 1  213.135.45.254       0.165 ms  0.152 ms  0.145 ms
 2  213.135.44.136       0.281 ms  0.283 ms  0.277 ms
 3  80.50.83.9    0.783 ms  0.779 ms  0.771 ms
 4  195.205.0.9          0.914 ms  0.912 ms  0.897 ms
 5  212.91.0.71          1.452 ms  1.448 ms  1.436 ms
 6  195.245.213.249      1.173 ms  1.124 ms  1.107 ms
```

..45.22| (1 sieć) ..45.254X? (2 sieć) ..44.136X? (3 sieć) ..83.9X? (4 sieć) ..0.9X? (5 sieć) ..0.71X? (6 sieć) |..213.249