Linux i Python w Elektronicznej Sieci #10: Sieci komputerowe – podstawy

Projekt „Matematyka dla Ciekawych Świata”,
Robert Ryszard Paciorek
<rrp@opcode.eu.org>

2023-06-25

1 Podstawy TCP/IP

Sieci komputerowe działają na zasadzie przesyłania informacji w postaci porcji, z których każda posiada co najmniej informację o adresie odbiorcy (zwykle też nadawcy), nazywanych ramkami lub pakietami. Kierowanie pakietów w odpowiednie miejsce odbywa się na podstawie adresu pakietu i nie jest związane z fizycznym zestawianiem łącza pomiędzy nadawcą a odbiorcą - każdy pakiet jest kierowany niezależnie, a w ramach pojedynczego łącza (kanału transmisji) mogą być przekazywane pakiety adresowane do różnych odbiorców. Nazywane jest to komutacją pakietów, w odróżnieniu od komutacji łącza (która występowała np. w klasycznej, analogowej telefonii, gdzie przekaźniki w centralach dokonywały zestawienia połączeń elektrycznych między dwoma aparatami telefonicznymi).

1.1 Struktura warstwowa

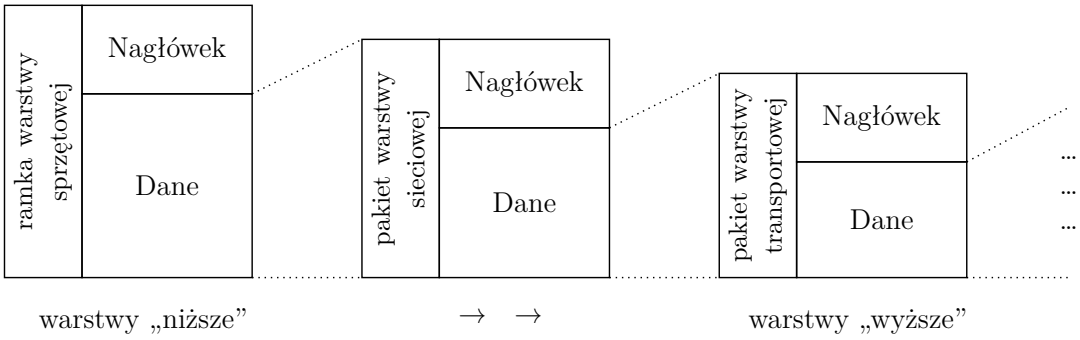
Komunikacja sieciowa typowo posiada strukturę warstwową. W modelu OSI wyróżnia się 7 warstw:

1. fizyczną (pierwszą) definiującą aspekty związane z fizycznym przesyłem sygnału takie jak częstotliwości radiowe, poziomy napięć, etc.; określa sposób transmisji kolejnych bajtów
2. łącza danych (drugą) definiującą aspekty związane z formatem ramki, protokoły ustalania zasad dostępu do medium transmisyjnego, itd.; określa sposób transmisji porcji danych pomiędzy hostami w jednej sieci
3. sieciową (trzecią) definiującą aspekty związane z formatem pakietu, adresacją i zasady routingu umożliwiające zapewnienie łączności pomiędzy różnymi sieciami; określa sposoby transmisji porcji danych pomiędzy sieciami
4. transportową (czwartą) odpowiedzialną za podział strumienia na porcje informacji, kontrolę nad poprawnością transmisji, adresację usług w ramach hosta
5. sesji (piątą)
6. prezentacji (szóstą)
7. aplikacji (siódmą)

W modelu TCP/IP wyróżnia się 4 warstwy:

1. Dostępu do sieci - obejmującą warstwy 1 i 2 modelu OSI
2. Internetu - obejmującą warstwę 3 modelu OSI
3. Transportową - obejmującą warstwę 4 modelu OSI
4. Aplikacji - obejmującą warstwy 5, 6 i 7 modelu OSI

Z punktu widzenia modelu TCP/IP można powiedzieć o enkapsulacji danych kolejnych warstw w ramach warstwy niższej, czyli „surowe” dane (np. strona HTML) obudowywane są strukturą opisywaną przez warstwę aplikacji (np. nagłówkami HTTP), następnie całość ta umieszczana jest w polu danych pakietu warstwy transportowej (np. TCP), ten z kolei w polu danych pakietu IP (warstwy sieciowej), na koniec pakiet IP jest umieszczany w polu danych ramki warstwy dostępu do sieci (np. ramki ethernetowej). W ramach podróży przez kolejne sieci pakiet IP jest wyjmowany i wkładany w kolejne ramki warstwy



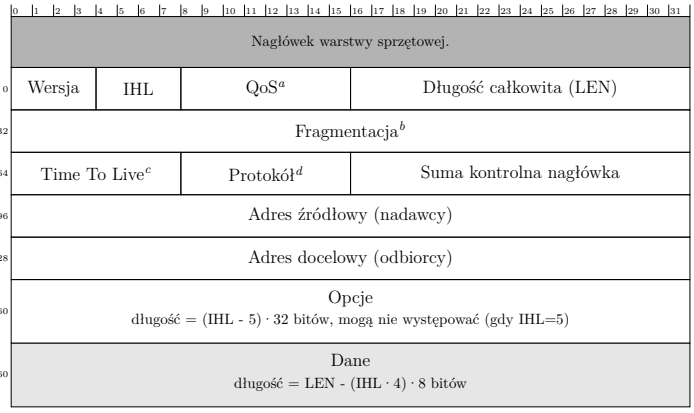
Rysunek 1: Struktura warstwowa protokołów sieciowych

dostępu do sieci, na ogół tylko z niewielkimi ingerencjami w zawartość tego pakietu (prawie zawsze nie dochodzącymi do pola danych pakietu TCP lub datagramu UDP, czyli nie wykraczającymi poza warstwę 4 OSI).

1.2 Protokół IP

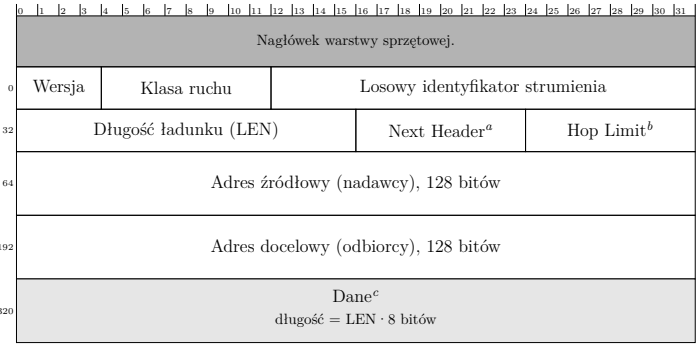
Protokół IP (Internet Protocol) odpowiedzialny jest przede wszystkim za sposób adresacji hostów oraz reguły komutacji pakietów (routing). Jest on wspomagany przez kolejny protokół z tej rodziny - ICMP (Internet Control Message Protocol), którego zadaniem jest przekazywanie informacji kontrolnych np. o nieosiągalności hosta docelowego, odrzuceniu przetwarzania pakietu ze względu na zbyt dużą liczbę skoków (gdy wartość pola TTL z nagłówka IP wyniesie zero) a także pingi (zarówno żądanie jak i odpowiedź).

Pakiet IPv4:



a. Dokładniej dwa pola: DSCP – priorytetyzacja ruchu DiffServ i ECN – informacja o przeciążeniu
b. Obsługa mechanizmu podziału pakietu IP na mniejsze części (gdy nie mieści się do ramki warstwy drugiej). Dokładniej są to trzy pola: Identification – numer identyfikujący zbiór fragmentów, Flagi – informujące o zakazie lub dokonaniu fragmentacji, oraz Offset – przesunięcie fragmentu względem całości pakietu.
c. Licznik zmiejszany gdy pakiet przechodzi prze router, służy eleiminacji zapętleń w sieci IP.
d. Numer identyfikujący protokół warswy wyższej znajdujący się w polu danych.

Pakiet IPv6:



a. Tak samo jak pole „Protokół” w nagłówku IPv4.
Jednak w odróżnieniu od IPv4 pole to może wskazywać nie tylko nagłówek warstwy wyższej, ale także opcjonalny, dodatkowy nagłówek IPv6, zastępujący informacje podawane w polu opcji IPv4. W takim wypadku posiada on także swoje pole „Next Header”, itd., aż do momentu dotarcia do nagłówka warstwy transportowej.
b. Pełni rolę taką jak *Time To Live*.
c. Zależnie od wartości pola „Next Header” mogą to być kolejne opcjonalne nagłówki dodatkowe IPv6 i następnie „dane warstwy wyższej” lub od razu „dane warstwy wyższej”.

Rysunek 2: Struktura pakietów IP

1.3 Adresacja IP

Adresy hostów (nazywane adresami IP) są to 32-bitowe (w IPv4) lub 128-bitowe (w IPv6) liczby. Adresy IPv4 zapisywane są najczęściej w notacji kropkowo-dziesiętnej, gdzie każdy bajt (ciąg 8 bitów) zapisywany jest jako liczba dziesiętna rozdzielana kropką od pozostałych. Adresy IPv6 zapisywane są zazwyczaj w notacji dwukropokowej, polegającej na zapisywaniu 16 bitowych części adresu liczbami szesnastkowymi oddzielanymi dwukropkiem, dodatkowo jeden ciąg zer (o długości będącej wielokrotnością 16 bitów) może być skompresowany (pominięty) co daje w zapisie dwa dwukropki `::`.

1.3.1 Długość prefixu i maska

Adresy hostów grupuje się w adresy sieci, bazując na jednakowym (bitowo) początku takiego adresu (zwanym adresem sieci lub prefixem). Ilość bitów stanowiących adres sieci w danym adresie IP nazywana jest długością prefixu i zapisywana jest zazwyczaj po ukośniku[1]. Na przykład zapis `2001:db8::a17/48` oznacza że pierwsze 48 bity stanowią adres sieci a kolejne $128 - 48 = 80$ bitów stanowi adres hosta w tej sieci.

Długość prefixu jednoznacznie określa maskę danej podsieci, czyli liczbę odpowiadającą długości adresu (32 bity lub 128 bitów), złożoną z ciągu jedynek o długości prefixu oraz ciągu zer (o długości adresu hosta). W przypadku IPv4 spotykane jest także podawanie maski sieci w notacji kropkowo-dziesiętnej zamiast długości prefixu.

Przykład

```
adres IPv4 zapisany z informacją o długości prefixu: 10.23.45.56/13, czyli:
adres: 10.23.45.56 = 00001010000101110010110100111000
maska: 255.248.0.0 = 11111111111110000000000000000000
                     prefix sieci adres hosta w sieci
adres sieci        = 00001010000100000000000000000000 = 10.16.0.0

Adres sieci obliczany jest jako bitowy AND pomiędzy adresem i maską.

Maska zawsze jest złożona z ciągu samych bitów o wartości 1 a następnie o wartości 0.
Bitów o wartości 1 jest tyle ile wynosi długość prefixu (podawana po /), czyli w tym przykładzie 13.

W IPv6 działa to analogicznie, tyle że adres ma 128 bitów długości, stosuje się notacje dwukropkową
zamiast kropkowo-dziesiętnej i nie stosuje się jawnego zapisu maski (a jedynie długość prefixu).
```

Sieć może zostać podzielona na mniejsze sieci (z większą wartością prefixu), jak też grupa sieci może zostać zagregowana w jedną większą (2^n raza) sieć (z prefixem mniejszym o n). Agregacja hostów i sieci w większe całości jest wykorzystywana w mechanizmach routingu, co pozwala na redukcję wielkości tablic routingowych.

1.3.2 Przynależność do sieci

Adres sieci zapisuje się typowo z wyzerowanymi bitami stanowiącymi adres hosta (czyli po dokonaniu bitowego *and* z maską danej sieci) oraz podaną informacją o długości prefixu, dla powyższego przykładu będzie to `2001:db8::/48`. Informacja taka jest wystarczająca do sprawdzenia czy dowolny inny adres IP należy do tej sieci czy nie.

```python
from ipaddress import *

# adres

adr     = ip_interface("2001:0db8::17:15")
adr_int = int(adr.ip)
print("Adress IPv6 jest 128 bitową liczbą całkowitą np.:")
print("  " + str(adr.ip) + " == " + hex(adr_int) + "\n")

# sieć - maska i długość prefixu

net         = ip_interface("::/112");
netmask     = net.network.netmask
netmask_int = int(netmask)
net_preflen = net.network.prefixlen

print("Maska podsieci IPv6 jest 128 bitową liczbą całkowitą np.:")
print("  " + str(netmask) + " == " + hex(netmask_int) + "\n")
```

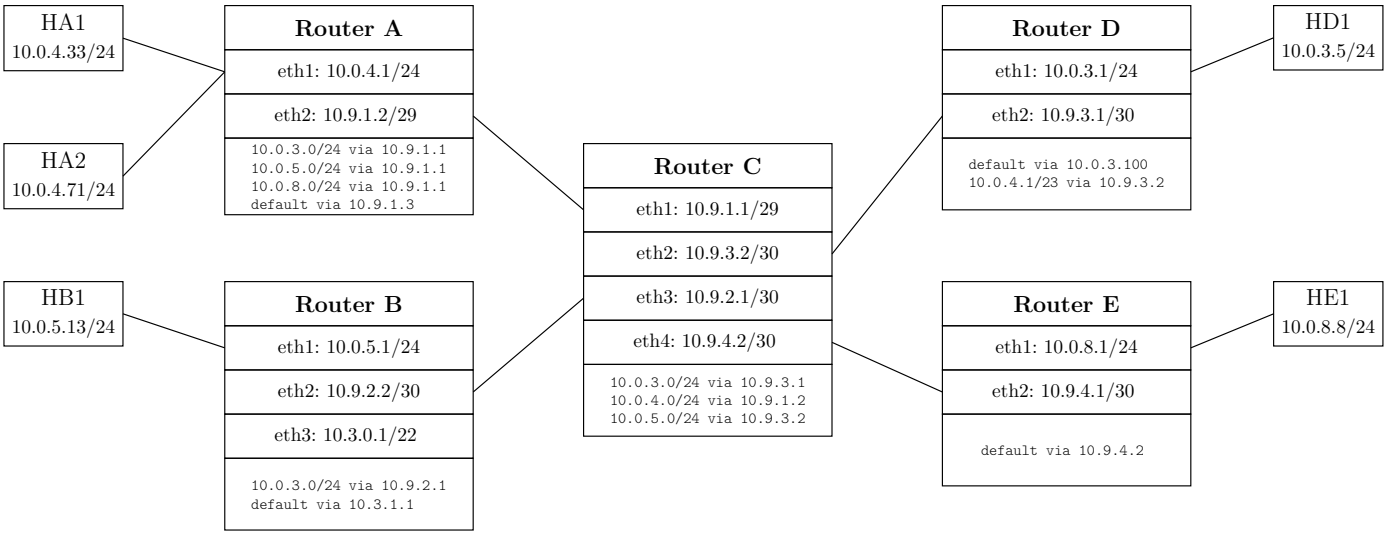
1. Jest to notacja *CIDR*. Przed wprowadzeniem tego mechanizmu w IPv4 funkcjonował klasowy sposób routingu (*classful*), gdzie wielkość maski była deteminowana wartością pierwszych bitów adresu – ale to już historia.

```
print("Jako że maska jest liczbą, która zapisana binarnie, zawsze zawiera ciągły ciąg bitów")
print("o wartości 1, a po nim ciągły ciąg bitów o wartości 0 (mogą być zerowej długości), to")
print("często stosowany jest zapis polegający na podawaniu długości prefiksu: /" + str(net_preflen))
print("jest to ilość bitów o wartości 1 w masce, czyli im większy prefix tym mniejsza sieć.\n")

# adres w sieci

adr2     = ip_interface("2001:0db8::17:15/112");
net2     = adr2.network
net2_int = int(net2.network_address)

print("Aby obliczyć adres sieci (czyli wspólną dla wszystkich hostów w danej sieci część")
print("adresu IP) należy wykonać binarny AND pomiędzy adresem IP hosta a maską podsieci.")
print("Dla powyższego przykładu:")
print("  " + hex(netmask_int & adr_int) + " == " + str(net2) + " == " + hex(net2_int) + "\n")

# aby sprawdzić czy adres IP należy do danej sieci trzeba obliczyć adres sieci tego hosta
# w oparciu o maskę sieci którą sprawdzamy
def sprawdzSiec(n, a):
        nn = int(a) & int(n.netmask)
        if nn == int(n.network_address):
                print(str(a) + " należy do sieci " + str(n))
        else:
                print(str(a) + " NIE należy do sieci " + str(n))

sprawdzSiec(net2, ip_interface("2001:0db8::17:ab13").ip)
sprawdzSiec(net2, ip_interface("2001:0db8::13:a").ip)
```

Zadanie 1.3.1

Ustal czy adres `2001:db8:0:a17::123` należy do sieci `2001:db8::/48`. Możesz posłużyć się narzędziami do obliczania zakresów sieci IP (np. `sipcalc`) lub obliczyć to ręcznie.

1.4 Routing



Uwaga: W tablicach routingu na ilustracji pominięto wpisy związane z pokazanymi interfejsami i ich adresami IP. Pominięto też tablice routingowe hostów H*, należy zakładać że oprócz wpisu związanego z adresem IP ustawionym na interfejsie mają one jedynie wpis default wskazujący na „ich" router np. tablica HA1 ma dwa wpisy: `default via 10.0.4.1` i `10.0.4.0/24 dev eth0`.

Rysunek 3: Przykładowe sieci wraz z trasami routingu.

Router kieruje każdy z pakietów do kolejnego routera lub bezpośrednio do hosta docelowego na podstawie jego adresu docelowego i tablicy routingu. Tablica taka zawiera adresy sieci wraz z adresami następnych

Na ilustracji 3 pokazano kilka przykładowych sieci wraz z ustawieniami routerów zapewniających komunikacje między nimi. Przykłady:

- Pakiet wysłany z HA1 do `10.0.3.13`:
  - zostanie przesłany przez *Router A*, który ma wpis `10.0.3.0/24 via 10.9.1.1`, do *Router C*
  - z *Router C*, który ma wpis `10.0.3.0/24 via 10.9.3.1`, zostanie przekazany do *Router D*
  - *Router D* przekaże go do odpowiedniego hosta (HD1) poprzez interfejs eth1
- Pakiet wysłany z HD1 do `10.0.4.33`:
  - zostanie przesłany przez *Router D*, który ma wpis `10.0.4.1/23 via 10.9.3.2`, do *Router C*
  - z *Router C*, który ma wpis `10.0.4.0/24 via 10.9.1.2`, zostanie przekazany do *Router A*
  - *Router A* przekaże go do odpowiedniego hosta (HA1) poprzez interfejs eth1
- Pakiet wysłany z HD1 do `10.0.5.13`:
  - zostanie przesłany przez *Router D*, który ma wpis `10.0.4.1/23 via 10.9.3.2`, do *Router C*
  - z *Router C*, który ma wpis `10.0.5.0/24 via 10.9.3.2`, zostanie przekazany do *Router B*
  - *Router B* przekaże go do odpowiedniego hosta (HB1) poprzez interfejs eth1

Zwróć uwagę że:

- aby możliwa była komunikacja trasa routingowa musi być:
  - skonfigurowana w obie strony (tak jak ma to miejsce między HA1 i HD1)
    - z tego powodu HE1 nie może komunikować się np. z HD1
  - skonfigurowana zarówno na routerach brzegowych (takich jak A i D), jak i wszystkich routerach pośrednich (takich jak C)
- *Router D* posiada zagregowany wpis w tablicy routingu obejmujący zarówno sieci podłączone do *Router A*, jak i do *Router B*, ale np. wpisy w *Router A* nie są i nie mogą być zagregowane

Polecenia wyświetlające trasy routingowe mogą wypisywać więcej informacji niż w powyższym przykładzie. Na przykład (jest wynik `ip r`, pokolorowany celem ilustracji, w innych, czy przy użyciu różnych opcji poleceniach może to wyglądać inaczej):

```
default via 213.135.60.1 dev eth-pub
192.168.0.0/23 via 10.0.1.100 dev eth-rsc
10.0.1.0/24 dev eth-rsc  proto kernel  scope link  src 10.0.1.13
```

W poszczególnych wpisach są to m.in.:

- sieć docelowa, default oznacza to samo co 0.0.0.0/0 lub ::/0
- via x.x.x.x – adres routera do którego ma trafić pakiet
- dev ... – urządzenie którym ma zostać wysłany
- src x.x.x.x – adres który ma zostać użyty jako adres nadacy

Ogólnie wpisy z via x.x.x.x oznaczają przesłanie do innego routera a wpisy bez tej informacji bezpośrednie wysłanie do hosta docelowego na wskazanym interfejsie.

routerów do nich prowadzących bądź wskazaniem lokalnego interfejsu sieciowego poprzez który powinny być osiągalne hosty z danej sieci. W tym celu korzysta z sprawdzania przynależności adresu do sieci, w celu ustalenia adresu następnego routera i/lub interfejsu sieciowego na który ma zostać przekazany pakiet.

Tablica przeglądana jest od wpisów najbardziej precyzyjnych, czyli z największym prefixem do wpisów najbardziej ogólnych (ostatnim wpisem jest na ogół trasa domyślna czyli sieć ::/0 dla IPv6 lub 0.0.0.0/0 dla IPv4). Dzięki czemu jeżeli kilka wpisów (sieci) z tablicy routingu pasuje do adresu docelowego z nagłówka pakietu, wybierany jest wpis najbardziej precyzyjny (o najdłuższym prefixie), a pasująca do każdego adresu trasa domyślna wybierana jest tylko gdy nie ma żadnej lepszej. Może się zdarzyć że kilka wpisów (nawet z tą samą maską) pasuje do adresu docelowego hosta, w takiej sytuacji do wyboru ścieżki używane są inne dane z tablicy routingu (takie jak metryka).

Tablice routingu mogą zawierać wpisy dodawane statycznie (wpisane do konfiguracji danego urzą-

dzenia), jak też wpisy dodawane dynamicznie w oparciu o protokołu wymiany informacji routingowych (protokoły routingu) takie jak: IGRP, OSPF, BGP. Protokoły routingu dynamicznego mogą być wykorzystywane m.in. do rozkładania obciążenia na różne łącza, zapewnienia redundancji łącz, blokowania ataków (D)DoS.

Także każdy z hostów ma tablice routingu, typowo składa się z dwóch pozycji – trasy do sieci lokalnej (tej sieci z której adres posiada dany host) wskazującej bezpośrednio na urządzenie sieciowe oraz trasy domyślnej wskazującej na router zapewniający dostęp do innych sieci, nazywany bramką (gateway). Jeżeli router nie posiada adresu w tej samej sieci co host konieczna jest dodatkowa trasa wskazująca poprzez jakie urządzenie dostępny jest router domyślny.

Oprócz opisanego powyżej routingu unicastowego (kierowania do jednego odbiorcy) realizowane są także transmisje:

- *anycast* – do dowolnego / najbliższego hosta o danym adresie; zasadniczo jest to transmisja unicast, tyle że adres docelowy nie jest unikalny w skali globalnej a różne routery kieruje te pakiety do różnych hostów docelowych (typowo wybierając najbliższy taki host)
- *multicast* – do grupy hostów, w tym wypadku (multicastowy) adres IP identyfikuje "kanał nadawczy" a nie unikalny host docelowy
- *broadcast* – do wszystkich hostów (w ramach danej sieci – nie są routowne), transmisje rozgłoszeniowe można traktować jako szczególny przypadek transmisji multicastowych w których grupa multicastowa obejmuje wszystkie hosty (można je zastąpić takimi transmisjami multicastowymi)

Zadanie 1.4.1

Wynik polecenia `ip -6 r` pokazującego tablicę rotingową wygląda następująco:

```
2001:db8:0:21::/64 dev eth1  proto kernel  metric 256
2001:db8::/32 via 2001:db8:0:21::100 dev eth2  metric 1024
2001:db8:fff:21::/64 dev eth2  proto kernel  metric 256
2001:db8:abc:21::/64 via 2001:db8:fff:21::1 dev eth2  metric 1024
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
default via 2001:db8:0:21::1 dev eth1  metric 1024
```

Ustal gdzie zostanie skierowany pakiet adresowany do `2001:db8:abc:21:123::ff3`.

2 Komunikacja TCP/IP

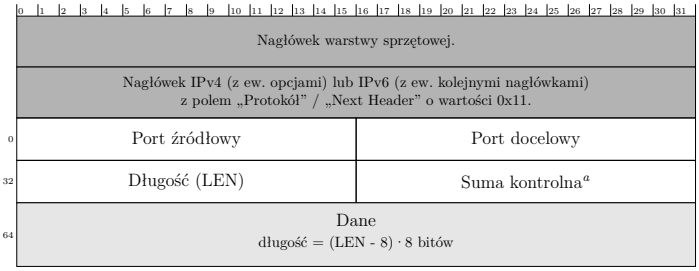
W oparciu o protokół IP działają protokoły warstwy transportowej takie jak UDP, TCP, czy też (mniej znany protokół dedykowany dla strumieniowych transmisji czasu rzeczywistego) SCTP.

Jednym z zadań tych protokołów jest identyfikowanie usługi (procesu) w ramach systemu posiadającego dany adres IP, do którego mają trafić dane. W tym celu zarówno UDP jak i TCP na każdym z hostów wyróżniają numeryczny identyfikator dla aplikacji/procesu/usługi będącego odbiorcą czy też nadawcą informacji zwany **numerem portu**.

Najprostszym protokołem warstwy transmisji wydaje się być UDP, protokół ten umożliwia przesłanie informacji pomiędzy dwoma hostami IP i nie kontroluje on tego czy została ona przesłana poprawnie. Natomiast TCP, w odróżnieniu od UDP, kontroluje to czy przesłana informacja dotarła do adresata i nie została uszkodzona, a w przypadku problemów informacja wysyłana jest ponownie. TCP w związku z tym w przeciwieństwie do UDP musi otworzyć połączenie i wykorzystywać je do kontroli poprawności przesłania informacji, wymaga zatem przesłania większej liczby pakietów (co może prowadzić do pewnych opóźnień itp). W związku z tym TCP używany jest tam gdzie konieczna jest kontrola poprawności transmisji (oraz ponowne wysłanie zgubionego pakietu), UDP tam gdzie nie jest to potrzebne (a liczy się czas). Z połączeniem występującym w protokole TCP związane jest także pojęcie wielkości okna, czyli tego co ile (tysięcy) bajtów odbiorca musi potwierdzać odbiór pakietów. Wielkość ta jest dynamicznie dostosowywana

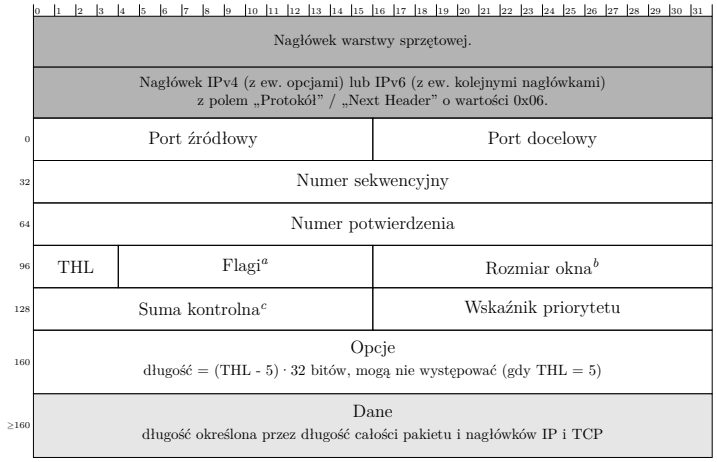
do parametrów łącza, pozwalając na sterowanie przepływem - jeżeli wysycana jest dostępna przepustowość łącza, dochodzi do strat pakietów i wielkość okna jest zmniejszana.

Datagram UDP:



a. Uwzględnia też wybrane pola z nagłówków IPv4 lub IPv6.

Pakiet TCP:



a. 3 bity rezerwy i flagi związane ze stanem połączenia TCP.
b. Informuje o ilości danych które odbiorca może aktualnie przyjąć.
c. Uwzględnia też wybrane pola z nagłówków IPv4 lub IPv6.

Rysunek 4: Struktura pakietów UDP i TCP

2.1 Popularne usługi

W ramach sieci mogą być realizowane różne usługi w oparciu o różne protokoły warstwy aplikacyjnej. Standardowe usługi posiadają zdefiniowane domyślne adresy portów dla swoich protokołów. Wśród usług i protokołów sieciowych należy wymienić przynajmniej:

- DNS (Domain Name System) - odpowiedzialny za system mapujący nazwy alfanumeryczne hostów na adresy IP.
- mechanizmy auto konfiguracji hostów - DHCP, rozgłaszanie informacji routingowej poprzez ICMPv6 (protokół warstwy 3)
- WWW - udostępnianie treści z użyciem protokołu HTTP
- pocztę elektroniczną - przesyłanie wiadomości (protokoły SMTP, IMAP, POP)
- komunikację natychmiastową i telefonię IP (protokoły SIP, XMPP, IAX)
- SSH - zdalny, szyfrowany dostęp do systemów IT, przesył plików oraz tunelowanie innych usług

2.1.1 Domain Name System

DNS umożliwia mapowanie nazwy na adres IP (lub wiele adresów IP) oraz przechowywanie dodatkowych informacji na temat domeny i znajdujących się w niej usług.

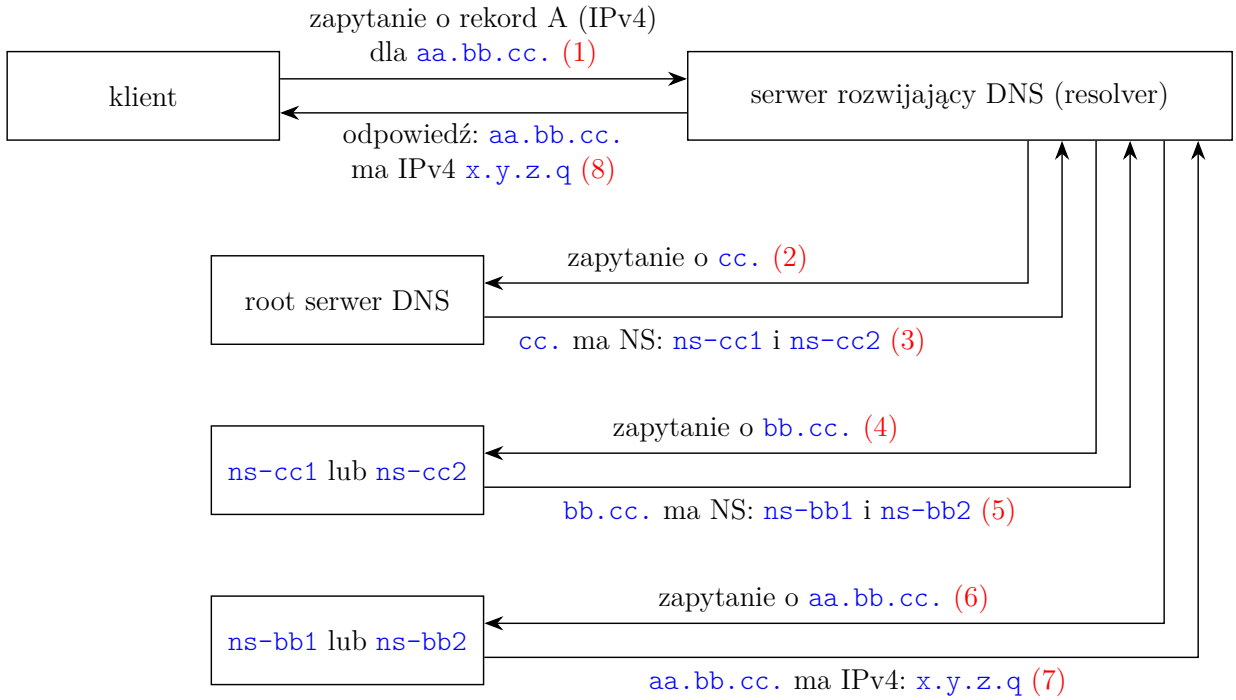
Domeny posiadają budowę hierarchiczną / drzewiastą:

- precyzja rośnie od prawej do lewej
- kolejne poziomy oddzielane są kropkami
- najwyższym poziomem jest kropka będąca ostatnim znakiem w pełnej nazwie domenowej (np. `ciekawi.icm.edu.pl.`), którą najczęściej pomija się w zapisie
- hierarchia ta jest niezależna od hierarchii routingu i wynika z faktu posiadania/użytkowania danej (pod)domeny)

Realizacja odpowiedzi na zapytanie DNS wygląda następująco:

1. host kieruje zapytanie do określonego w jego konfiguracji serwera "rozwijającego" DNS (DNS resolver),
2. serwer taki sprawdza w swojej pamięci podręcznej czy zna odpowiedź na to zapytanie (i nie jest ona przeterminowana - nie upłynął czas TTL od odnalezienia), jeżeli nie ma jej w swojej pamięci to

3. serwer taki zna adresy głównych serwerów DNS (root serwerów) zawierających informacje na temat serwerów obsługujących domeny najwyższego rzędu i kieruje do jednego z nich zapytanie o serwer obsługujący skrajnie prawą część adresu (np. *.org*),
4. do otrzymanego serwera kierowane jest zapytanie o większą część adresu (np. *eu.org*),
5. itd. aż do uzyskania odpowiedzi o pytany adres



(X) oznacza kolejność wykonywanych operacji. Przed wykonaniem zapytania *resolver* sprawdza czy nie posiada w swoim cache zapamiętanej (i nie przeterminowanej) odpowiedzi na to zapytanie.

Rysunek 5: Realizacja zapytania o rekord DNS

DNS przechowuje informacje w postaci rekordów mających określony typ (w większości przypadków dla danej nazwy domenowej może być zdefiniowanych wiele rekordów, tego samego lub innych typów). Wśród najważniejszych typów rekordów należy wymienić:

- `NS` – informacja o serwerach obsługujących DNS danej domeny
- `A` – mapowanie nazwy na adres IPv4
- `AAAA` – mapowanie nazwy na adres IPv6
- `MX` – informacja o serwerach obsługujących pocztę danej domeny
- `SRV` – informacje o hoście świadczącym usługę w tej domenie (usługa określana jest w nazwie domeny o którą pytamy)
- `PTR` – mapowanie adresów IP na nazwy domenowe, realizowane w specjalnym drzewie `in-addr.arpa` (dla IPv4) lub `ip6.arpa` (IPv6), gdzie adres IP zapisywany jest w odwróconej kolejności po bajcie dla IPv4 lub cyfrze szesnastkowej dla IPv6
- `TXT` – informacje dodatkowe (np. jakie serwery pocztowe, są upoważnione do wysyłania poczty z tej domeny)
- `SOA` – informacje podstawowe o strefie opisującej domenę
- `CNAME` – alias na inną domenę (domena którą aliasujemy nie może mieć innych wpisów, nawet SOA)

2.1.2 Standardowe numery portów

Popularne usługi (np. www) posiadają ustalone standardowe numery portów na których nasłuchiwac będzie serwer takiej usługi (np. dla wspomnainego www jest to port 80). Informacja o numerze portu usługi może

być umieszczona także w rekordzie SRV systemu DNS.

3 Diagnostyka sieci

Istnieje wiele poleceń służących do diagnozowania ewentualnych problemów sieciowych lub mogących być w tym przydatnymi. Poniżej znajduje się zestawienie najbardziej popularnych / użytecznych narzędzi z podziałem wg zastosowań.

3.1 Adresy

- `ipcalc` oraz `sipcalc` – kalkulator IP (pozwalający na obliczanie adresów sieci rozgłoszeniowych, zmianę notacji itd)
- `whois` – informacje z bazy whois (o domenie lub adresie IP)

3.2 Dostępność i trasy do hostów

- `ping [opcje] host` lub `ping6 [opcje] host` – sprawdzanie dostępności hosta z użyciem protokołu ICMP (obecnie komenda `ping6` najczęściej jest równoważna poleceniu ping z opcją `-6` wymuszającą używanie jedynie IPv6, na starszych systemach komenda ping może nie obsługiwać adresów IPv6 i wtedy konieczne jest stosowanie do nich polecenia `ping6`), ważniejsze opcje:
  `-c n` wykonaj n zapytań (domyślnie pyta do momentu przerwania przy pomocy np. Ctrl-C, lub sygnału wysłanego z uzyciem komendy `kill`)
  `-n` nie zamieniaj adresu IP hosta który odpowiedział na nazwę domenową
- `traceroute`, `traceroute6`, `tracepath`, `tracepath6`, `tcptraceroute` lub `tcptraceroute6` – sprawdzanie ścieżki do hosta (wypisanie listy routerów przez które przechodzi pakiet w drodze do wskazanego hosta)
  Istnieją różne warianty tych poleceń (nawet pod tą samą nazwą), różnią się one stosowanymi mechanizmami i domyślnymi opcjami. Generalnie wszystkie uruchamia się na zasadzie `polecenie [opcje] host`. Warianty z `6` na końcu nazwy będą używały jedynie adresów IPv6, natomiast polecenia bez `6` na końcu nazwy mogą potrafić ich używać lub nie. Wszystkie popularne warianty pozwalają na podanie opcji `-n` wyłączającej zamienianie adresu IP hosta który odpowiedział na jego nazwę domenową.
  Może zdarzyć się że śledzenie urwie się na jakimś hoście (np. z powodu jego konfiguracji lub błędów w jego oprogramowaniu sieciowym), może się zdarzyć że przy użyciu innej komendy z tej grupy (lub zmianie opcji) uda się prześledzić dalszą trasę pakietu.
- `mtr [opcje] host` – sprawdzanie ścieżki do hosta (czyli podobnie jak traceroute i tracepath) w trybie ciągłym (z ciągłym odświeżaniem) wraz z wypisywaniem informacji o stratach pakietów i opóźnieniach na poszczególnych odcinkach, ważniejsze opcje:
  `-n` nie zamieniaj adresu IP hosta który odpowiedział na nazwę domenową
- `nmap` – skaner sieciowy - sprawdzanie dostępnych hostów w sieci, otwartych portów, itd
- `arping` – narzędzie do pingowania z wykorzystaniem zapytań ARP zamiast ICMP
  istnieją dwie zasadnicze odmiany: z iputils oraz z synscan; ta druga zawarta w debianowym pakiecie "arping" umożliwia także pingowanie po adresie MAC (ale nie przez RARP, bo on nie do tego służy), aby to jednak działało host docelowy nie może ignorować pingów rozgłoszeniowych, metoda obejścia opisana jest w README arping'a
- `arp-scan` – wyszukiwanie hostów w oparciu o zapytania ARP (można powiedzieć że jest to równoważne uruchamianiu komendy arping w pętli)

3.3 DNS

- `dig [opcje] nazwa [typRekordu]` – narzędzia do uzyskania informacji z DNS, pozwala na okre-

ślenie poprzez `@adres` serwera który chcemy odpytać oraz na określenie (poprzez drugi argument) typu rekordu który chcemy uzyskać, zamiast typu rekordu można podać: `ANY` (powoduje odpytanie o wszystkie rekordy) lub `AXFR` (powoduje wysłanie prośby o transfer całej strefy, działa jeżeli dany host ma prawo transferu całej strefy z danego serwera)

- `host [opcje] nazwa|ip [server]` – narzędzia do zamiany adresów domenowych na IP i odwrotnie oraz wyciągania innych informacji z DNS (np. rekordy MX)
- `nslookup [opcje] nazwa|ip [server]` – narzędzia do zamiany adresów domenowych na IP i odwrotnie oraz wyciągania innych informacji z DNS (np. rekordy MX)
- `dnstracer` – śledzenie trasy zapytań DNS
- `dnswalk` – debuger DNS

3.4 IPv6

- `ndisc6` – testowanie ICMPv6 Neighbor Discovery
- `rdisc6` – testowanie ICMPv6 Router Discovery
- `rltraceroute6` – trasa pakietów do danego hosta IPv6 z użyciem UDP/ICMP
- `tcpspray6` – pomiar prędkości łącza z użyciem TCP/IP Discard/Echo
- `na6` / `ns6` – wysyłanie pakietów Neighbor Advertisement / Solicitation
- `ra6` / `rs6` – wysyłanie pakietów Router Advertisement / Solicitation
- `ni6` / `rd6` – wysyłanie pakietów ICMPv6 Node Information / Redirect
- `scan6` – skanowanie sieci IPv6

3.5 debugowanie łączności sieciowej

- `netcat` lub `nc` lub `netcat6` – program pozwalający na wysyłanie pakietów TCP i UDP z definiowaną przez nas zawartością, oraz odbiór pakietów TCP i UDP (słuchanie na wskazanym porcie), umożliwia m.in. testowanie usług sieciowych (takich jak smtp, www, jabber, ...); uwaga: występuje w kilku wersjach różniących się opcjami
- `telnet` – program umożliwiający zdalny (nieszyfrowany, łącznie z hasłem!) dostęp do powłoki, a także (podobnie jak netcat) m.in. testowanie usług sieciowych
- `swaks` – narzędzie do testowania SMTP
- `tcpdump` – przechwytuje komunikację sieciową celem analizy nagłówków lub pełnej zawartości pakietów (wsparcie dla niektórych z protokołów warst wyższych wymaga doinstalowania - np. obsługę DHCP zapewnia dhcpdump)
- `wireshark` lub `tshark` – przechwytuje komunikację sieciową celem analizy nagłówków lub pełnej zawartości pakietów, wspiera dekodowanie wielu protokołów warstwy aplikacyjnej, wireshark posiada graficzny interfejs użytkownika, tshark jest wersją konsolową

3.6 informacje o wykorzystaniu i prędkości sieci

- `netstat` – informacje o sieci (np. `netstat -l46np | sort -t / -k 2` wypisze informację o nasłuchujących (po IPv4 lub IPv6) usługach posortowane po nazwie procesu)
- `iptraf` – monitor IP LAN
- `nload` – graficzne (ncurses) pokazywanie wykorzystania (prędkości) interfejsów sieciowych
- `ttcp` – testuje prędkość połączenia sieciowego (strona domowa, najnowsza wersja oraz mutacja)
- `iperf` – pomiar prędkości połączenia sieciowego

Zadanie 3.0.1

Korzystając z narzędzi służących do diagnozowania sieci sprawdź czy host `ciekawi.icm.edu.pl` jest dostępny. Jakiego polecenia użyłeś(aś) w tym celu? Co jeszcze mówi wynik tego polecenia?

Zadanie 3.0.2

Korzystając z narzędzi służących do diagnozowania sieci ustal jaką trasą podróżują pakiety z Twojego komputera do `www.opcode.eu.org` oraz do `www.example.org`. Jakiego lub jakich poleceń użyłeś(aś) w tym celu? Co jeszcze mówi ich wynik? Co możesz powiedzieć porównując uzyskane trasy?

Zadanie 3.0.3

Ustal (wszystkie) adresy IPv4 i IPv6 serwera `www.bitbucket.org`. Zastanów się czemu może służyć to że niektóre nazwy domenowe rozwijają się na wiele różnych adresów IP.

Zadanie 3.0.4

Korzystając z dwóch instancji programu `nc` (`netcat`) – jednej w roli serwera, drugiej w roli klienta prześlij między nimi jakieś dane. Użyj programu `tcpdump` (z odpowiednimi opcjami) aby podsłuchać komunikację sieciową między tymi programami i zobaczyć przesyłane dane.

Zadanie 3.0.5

Korzystając bezpośrednio z poleceń protokołu HTTP i programu `nc` (`netcat`) lub `telnet`, pobierz i wyświetl kod strony `http://www.opcode.eu.org/`.
Wskazówka: Opis protokołu HTTP odnajdziesz bez problemu w sieci.
Ogólnie żądanie HTTP składa się z pierwszej linii określającej typ wykonywanej operacji, ścieżkę oraz wersję protokołu - np. ***`GET /abc.txt HTTP/1.1`*** *oznacza prośbę o zwrócenie zawartości pliku* ***`/abc.txt`****. Następnie podawane są nagłówki, w wersji HTTP 1.1 obowiązkowy jest nagłówek „Host" określający nazwę domenową serwera - np.* ***`Host: www.example.org`****. Po nagłówkach występuje pusta linia po której mogą być przesłane (przy niektórych typach żądań) jakieś dane (np. z wypełnionego na stronie formularza).*

Zadanie 3.0.6

Zadanie 3.0.5 można rozwiązać przy pomocy netcat'a bez dodatkowych opcji, jednak jeżeli stroną do pobrania byłoby np. `http://www.icm.edu.pl` to należałoby skorzystać z opcji `-C` netcat'a (w przeciwnym razie serwer zwraca błąd 400 "Bad Request"). Sprawdź co robi ta opcja i zastanów się dlaczego w przypadku niektórych serwerów jest konieczna a w przypadku innych nie? Co na ten temat mówi standard HTTP?

Zadanie 3.0.7

Zobacz czy rozwiązanie zadania 3.0.1 zadziała gdy użyjesz nazwy serwera zawierającej polskie znaki: `licealiści.icm.edu.pl`. Jak myślisz, dlaczego polskie znaki są tak rzadko używane w nazwach domenowych?

4 Wykład wideo[2]

- *Podstawy sieci komputerowych* – http://video.opcode.eu.org/10.01.mkv
- *Internet Protocol* – http://video.opcode.eu.org/10.02.mkv
- *Adresacja i routing IP* – http://video.opcode.eu.org/10.03.mkv
- *TCP, UDP, ICMP i DNS* – http://video.opcode.eu.org/10.04.mkv
- *Usługi sieciowe i tunele* – http://video.opcode.eu.org/10.05.mkv

5 Zadania

Poniższe zadania znajdują się także w odpowiednich rozdziałach skryptu. Zostały jednak zamieszczone zbiorczo także w tym miejscu dla wygody czytelnika.

Zadanie 1.3.1

Ustal czy adres `2001:db8:0:a17::123` należy do sieci `2001:db8::/48`. Możesz posłużyć się narzędziami do obliczania zakresów sieci IP (np. `sipcalc`) lub obliczyć to ręcznie.

Zadanie 1.4.1

Wynik polecenia `ip -6 r` pokazującego tablicę rotingową wygląda następująco:

```
2001:db8:0:21::/64 dev eth1  proto kernel  metric 256
2001:db8::/32 via 2001:db8:0:21::100 dev eth2  metric 1024
2001:db8:fff:21::/64 dev eth2  proto kernel  metric 256
2001:db8:abc:21::/64 via 2001:db8:fff:21::1 dev eth2  metric 1024
fe80::/64 dev eth1  proto kernel  metric 256
fe80::/64 dev eth2  proto kernel  metric 256
default via 2001:db8:0:21::1 dev eth1  metric 1024
```

Ustal gdzie zostanie skierowany pakiet adresowany do `2001:db8:abc:21:123::ff3`.

Zadanie 3.0.1

Korzystając z narzędzi służących do diagnozowania sieci sprawdź czy host `ciekawi.icm.edu.pl` jest dostępny. Jakiego polecenia użyłeś(aś) w tym celu? Co jeszcze mówi wynik tego polecenia?

Zadanie 3.0.2

Korzystając z narzędzi służących do diagnozowania sieci ustal jaką trasą podróżują pakiety z Twojego komputera do `www.opcode.eu.org` oraz do `www.example.org`. Jakiego lub jakich poleceń użyłeś(aś) w tym celu? Co jeszcze mówi ich wynik? Co możesz powiedzieć porównując uzyskane trasy?

Zadanie 3.0.3

Ustal (wszystkie) adresy IPv4 i IPv6 serwera `www.bitbucket.org`. Zastanów się czemu może służyć to że niektóre nazwy domenowe rozwijają się na wiele różnych adresów IP.

2. Filmy posiadają napisy wgrane do kontenera multimedialnego jako osobny strumień – napisy mogą być włączone lub wyłączone w odtwarzaczu. W wielu filmach dużo dzieje się "na dole ekranu", dlatego polecamy odtwarzać filmy z napisami umieszczonymi poniżej filmu, np. przy pomocy polecenia: `vlc --video-filter='croppadd{paddbottom=120}' --sub-margin=-10 PLIK.mkv`

Zadanie 3.0.4

Korzystając z dwóch instancji programu `nc` (`netcat`) – jednej w roli serwera, drugiej w roli klienta prześlij między nimi jakieś dane. Użyj programu `tcpdump` (z odpowiednimi opcjami) aby podsłuchać komunikację sieciową między tymi programami i zobaczyć przesyłane dane.

Zadanie 3.0.5

Korzystając bezpośrednio z poleceń protokołu HTTP i programu `nc` (`netcat`) lub `telnet`, pobierz i wyświetl kod strony `http://www.opcode.eu.org/`.
Wskazówka: Opis protokołu HTTP odnajdziesz bez problemu w sieci.
Ogólnie żądanie HTTP składa się z pierwszej linii określającej typ wykonywanej operacji, ścieżkę oraz wersję protokołu - np. `GET /abc.txt HTTP/1.1` *oznacza prośbę o zwrócenie zawartości pliku* `/abc.txt`*. Następnie podawane są nagłówki, w wersji HTTP 1.1 obowiązkowy jest nagłówek „Host" określający nazwę domenową serwera - np.* `Host: www.example.org`*. Po nagłówkach występuje pusta linia po której mogą być przesłane (przy niektórych typach żądań) jakieś dane (np. z wypełnionego na stronie formularza).*

Zadanie 3.0.6

Zadanie 3.0.5 można rozwiązać przy pomocy netcat'a bez dodatkowych opcji, jednak jeżeli stroną do pobrania byłoby np. `http://www.icm.edu.pl` to należałoby skorzystać z opcji `-C` netcat'a (w przeciwnym razie serwer zwraca błąd 400 "Bad Request"). Sprawdź co robi ta opcja i zastanów się dlaczego w przypadku niektórych serwerów jest konieczna a w przypadku innych nie? Co na ten temat mówi standard HTTP?

Zadanie 3.0.7

Zobacz czy rozwiązanie zadania 3.0.1 zadziała gdy użyjesz nazwy serwera zawierającej polskie znaki: `licealiści.icm.edu.pl`. Jak myślisz, dlaczego polskie znaki są tak rzadko używane w nazwach domenowych?

6 Rozwiązania zadań

Poniżej zamieszczone są przykładowe rozwiązania „głównych" zadań z tego skryptu wraz z komentarzami. Wiemy że zajrzenie do nich już przy pierwszej trudności jest kuszące, mimo to rekomendujemy przynajmniej podjąć uczciwą, co najmniej kilkunastominutową na każde z zadań, próbę rozwiązania tych zadania bez zaglądania do odpowiedzi.

Pamiętaj!: Samodzielne rozwiązanie problemu (wraz z wszystkimi trudnościami po drodze i popełnionymi błędami) jest dużo bardziej kształcące od nawet wielokrotnego przepisania gotowego rozwiązania, jednak nawet jednokrotne przepisanie rozwiązania jest bardziej kształcące od wielokrotnego przekopiowania go.

Rozwiązanie zadania 1.3.1

tak — zakres sieci 2001:06a0:0000:0000:0000:0000:0000:0000 - 2001:06a0:0000:003f:ffff:ffff:ffff:ffff

Rozwiązanie zadania 1.4.1

Adres 2001:db8:abc:21:123::ff3 zawiera się w zakresie sieci następujących sieci wystepujących w tablicy routingu:

- 2001:db8::/32
- 2001:db8:abc:21::/64
- default (0::/0)

Najdłuższy prefix spośród tych sieci ma 2001:db8:abc:21::/64 (64 bity), zatem zostanie użyty wpis w tablicy routingowej dla tej sieci.
W efekcie pakiet zostanie skierowany do routera 2001:db8:fff:21::1 poprzez interfejs eth2.

Rozwiązanie zadania 3.0.1

```
$ ping -c4 ciekawi.icm.edu.pl
PING www2.icm.edu.pl (213.135.59.55) 56(84) bytes of data.
64 bytes from www2.icm.edu.pl (213.135.59.55): icmp_seq=1 ttl=60 time=3.91 ms
64 bytes from www2.icm.edu.pl (213.135.59.55): icmp_seq=2 ttl=60 time=3.63 ms
64 bytes from www2.icm.edu.pl (213.135.59.55): icmp_seq=3 ttl=60 time=2.94 ms
64 bytes from www2.icm.edu.pl (213.135.59.55): icmp_seq=4 ttl=60 time=5.03 ms

--- www2.icm.edu.pl ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 6ms
rtt min/avg/max/mdev = 2.942/3.875/5.025/0.752 ms
```

Polecenie na standardowym wyjściu wypisuje m.in.:

- odpytywany adres IP i wynik odwrotnego dns'u dla niego
- ilość danych wysyłanych (może być użyteczne do testowania problemów z MTU)
- dla każdego zapytania, które dostało odpowiedź:
  - ilość przesłanych otrzymanych w pakiecie
  - serwer który odpowiedział (rev-dns i ip)
  - numer kolejny zapytania/odpowiedzi
  - TTL pakietu z odpowiedzią (wskazuje na ilość routerów przez które przeszedł pakiet)
  - czas potrzebny na uzyskanie odpowiedzi
- podsumowujące statystyki związane z ilością zagubionych pakietów i czasami odpowiedzi

Zamiast informacji o poszczególnych odpowiedziach polecenie ping może nic nie wypisywać (brak odpowiedzi):

```
$ ping -c 1 1.5.2.1
PING 1.5.2.1 (1.5.2.1) 56(84) bytes of data.

--- 1.5.2.1 ping statistics ---
1 packets transmitted, 0 received, 100% packet loss, time 0ms
```

lub podawać komunikat błędu wraz z jego nadawcą (jeżeli taki został zwrócony):

```
$ ping -c1 192.168.6.55
PING 192.168.6.55 (192.168.6.55) 56(84) bytes of data.
From 192.168.6.3 icmp_seq=1 Destination Host Unreachable

--- 192.168.6.55 ping statistics ---
1 packets transmitted, 0 received, +1 errors, 100% packet loss, time 0ms
```

Kod powrotu polecenie ping informuje o tym czy host jest osiągalny czy nie, dzięki czemu polecenie to może zostać uzyte np. w bash'owym warunku `if`:

```
if ping -c 1 10.0.1.1 >& /dev/null; then
        echo "10.0.1.1 jest dostępny"
else
        echo "10.0.1.1 nie jest dostępny"
fi
```

Rozwiązanie zadania 3.0.2

Należy skorzystać z polecenia `treacroute`, `tracepath`, `mtr` lub podobnych.
Wynik polecenia host może wyglądać następująco (ale nie będzie nic dziwnego jeżeli otrzymasz inny – trasa routingowa zależy od miejsca z którego się łączysz, może też zmieniać się z czasem)

```
traceroute  www.example.org
traceroute to www.example.org (93.184.216.34), 30 hops max, 60 byte packets
 1  funbox.home (192.168.6.1)  0.570 ms  9.648 ms  9.624 ms
```

```
 2  192.0.0.1 (192.0.0.1)  7.981 ms  8.007 ms  8.082 ms
 3  195.117.0.225 (195.117.0.225)  8.120 ms  8.189 ms  8.223 ms
 4  poz-r1.tpnet.pl (194.204.175.94)  18.474 ms  18.512 ms  18.547 ms
 5  hbg-b1-link.telia.net (213.248.96.144)  23.234 ms  23.268 ms  23.304 ms
 6  hbg-bb4-link.telia.net (213.155.135.86)  110.655 ms hbg-bb3-link.telia.net
    (213.155.135.80)  104.215 ms hbg-bb4-link.telia.net (213.155.135.86)  104.056
    ms
 7  * * ldn-bb4-link.telia.net (62.115.122.161)  101.809 ms
 8  * * *
 9  nyk-b6-link.telia.net (80.91.254.36)  106.471 ms nyk-b6-link.telia.net
    (62.115.125.63)  104.446 ms  104.562 ms
10  edgecast-ic-317660-nyk-b5.c.telia.net (62.115.147.201)  109.616 ms  109.669 ms
    112.148 ms
11  152.195.69.131 (152.195.69.131)  106.924 ms 152.195.68.141 (152.195.68.141)
    109.509 ms 152.195.69.139 (152.195.69.139)  106.711 ms
12  93.184.216.34 (93.184.216.34)  103.418 ms  108.218 ms  106.420 ms
13  93.184.216.34 (93.184.216.34)  106.454 ms  106.587 ms  110.858 ms
```

Podobnie jak w przypadku polecenia ping wypisywana jest informacja o odpytywanym adresie IP (i wyniik odwrotnego dns'u dla niego), ilość wysłanych danych. Wypisywane są kolejne routery (ip i rev-dns) wraz z czasami odpowiedzi (domyślnie traceroute na każdym poziomie robi 3 zapytania). Gwiazdki oznaczają brak odpowiedzi. Jeżeli od pewnego momentu wystepują same gwiazdki oznacza to że wraz z dalszym zwiększaniem TTL nie dostajemy już kolejnych komunikatów o jego przekroczeniu. Może to wynikać z osiągnięcia hostu docelowego, który ma zablokowany testowany port bez wysyłania komunikatu o niedostępności portu lub błędu routera który nie przekazuje poprawnie komunikatów o zbyt małym TTL.

Rozwiązanie zadania 3.0.3

Należy uźyć na przykład polecenia `host www.bitbucket.org` lub polecenń `dig AAAA www.bitbucket.org ; dig A www.bitbucket.org`.
Wynik polecenia host może wygląadć nastepująco (ale nie będzie nic dziwnego jeżeli otrzymasz inne adresy – dane w DNS ulegają zmianom, niekiedy nawet dynamicznie wprowadzanym).

```
www.bitbucket.org is an alias for bitbucket.org.
bitbucket.org has address 104.192.141.1
bitbucket.org has IPv6 address 2406:da00:ff00::22c5:2ef4
bitbucket.org has IPv6 address 2406:da00:ff00::6b17:d1f5
bitbucket.org has IPv6 address 2406:da00:ff00::22e9:9f55
bitbucket.org has IPv6 address 2406:da00:ff00::34cc:ea4a
bitbucket.org mail is handled by 1 aspmx.l.google.com.
bitbucket.org mail is handled by 5 alt1.aspmx.l.google.com.
```

Warto zauważyć że zostaliśmy poinformowani także o tym że www.bitbucket.org jest aliasem (CNAME) na bitbucket.org.
Zwracanie wielu IP jest jednym ze sposobów rozkładania obciążenia i zapewnienia redundancji. Innymi rozwiązaniami jest udzielanie różnych odpowiedzi różnym klientom (tak robi np. www.google.com):

```
$ host www.google.com
www.google.com has address 172.217.20.164
www.google.com has IPv6 address 2a00:1450:401b:802::2004
```

z innego hosta:

```
$ host www.google.com
www.google.com has address 216.58.206.4
www.google.com has IPv6 address 2a00:1450:4001:821::2004
```

Jeszcze innym jest używanie mechanizmów routingowych takich jak anycast.

Rozwiązanie zadania 3.0.4

Należy uruchomić w osobnych oknach terminala kolejno polecenia:

1. `tcpdump -i lo -A port 5555` - podsłuchujemy komunikację używającę portu 5555
2. `nc -lp 5555` - uruchamiamy nasłuch na porcie 5555

3. `nc localhost 5555` - łączymy się na port 5555

Dane wpisywane w jednym z uruchomionych netcat'ów będą wypisywane w drugim i odwrotnie. Wszystkie te dane (wraz z informacjami zawartymi w nagłówkach IP i TCP) będą wyświetlane w okienku w którym działa tcpdump.
Oczywiście możemy użyć innego numeru portu, itd.

Rozwiązanie zadania 3.0.5

```
netcat www.opcode.eu.org 80
GET / HTTP/1.1
Host: www.opcode.eu.org

HTTP/1.1 200 OK
Server: nginx/1.14.2
Date: Sun, 26 Apr 2020 09:12:09 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 11490
Last-Modified: Wed, 08 Apr 2020 17:36:32 GMT
Connection: keep-alive
ETag: "5e8e0ba0-2ce2"
Accept-Ranges: bytes

<?xml version="1.0" encoding="UTF-8" ?>
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="pl" lang="pl">
<head>
        <title>OpCode.eu.org - strona główna</title>
```

[...]

Jeżeli użyjemy innego nagłówka `Host:` w zapytaniu możemy dostać np. komunikat o przekierowaniu:

```
netcat www.opcode.eu.org 80
GET / HTTP/1.1
Host: opcode.eu.org

HTTP/1.1 301 Moved Permanently
Server: nginx/1.14.2
Date: Tue, 21 Apr 2020 17:58:44 GMT
Content-Type: text/html
Content-Length: 185
Connection: keep-alive
Location: http://www.opcode.eu.org/

<html>
<head><title>301 Moved Permanently</title></head>
<body bgcolor="white">
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx/1.14.2</center>
</body>
</html>
```

Przeglądarki WWW nie wyświetlają tych komunikatów tylko automatycznie przechodzą pod wskazany w nagłówku `Location:` adres.

Rozwiązanie zadania 3.0.6

Wynika to z stosowania w większości protokołów sieciowych (w tym w HTTP) jako znaku linii sekwencji dwu bajtowej \n\r (nowa linia, powrót karetki). Serwer WWW obsługujący domenę opcode.eu.org postępuje według filozofii nakazującej liberalne podejście do danych otrzymywanych i restrykcyjne do generowanych przez siebie (wysyłanych) i poprawnie interpretuje formalnie błędne żądanie zawierające znaki nowej linii w postaci \n. Serwer obsługujący www.icm.edu.pl nie jest już tak liberalny i wymaga znaków nowej linii w postaci \n\r, dlatego do netcata musimy dodać opcję -C aby konwertował wprowadzone znaki nowej linii na taką postać przed wysłaniem.

Rozwiązanie zadania 3.0.7

```
ping -c2 liceališći.icm.edu.pl
PING www2.icm.edu.pl (213.135.59.55) 56(84) bytes of data.
64 bytes from www2.icm.edu.pl (213.135.59.55): icmp_seq=1 ttl=60 time=4.20 ms
64 bytes from www2.icm.edu.pl (213.135.59.55): icmp_seq=2 ttl=60 time=3.26 ms

--- www2.icm.edu.pl ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 2ms
rtt min/avg/max/mdev = 3.260/3.732/4.204/0.472 ms
```

Jak widać działa. Standard kodowania dowolnych znaków Unicode, tak aby mogły być użyte w nazwach domenowych został opracowany w 2003 roku (Punycode, RFC3492), jest dość powszechnie zaimplementowany. Możemy użyć go także w pythonowej metodzie encode (np. `print( "źółw".encode("punycode") )`).
Ciężko powiedzieć dlaczego (przynajmniej w Polsce) znaki z poza ASCII są tak rzadko stosowane w nazwach domenowych.
Ciekawostka: kodowanie znaków nie ASCII w nazwie domenowej jest określone jednoznacznie, natomiast w dalszej części adresu URL niezbyt - zasadniczo zależy od konfiguracji serwera, ale standardem defacto jest tutaj UTF8).