Uniwersytet w Białymstoku

Wydział Matematyki i Informatyki

Instytut Informatyki

Fabian Neumann

Wieloadresowy serwer DNS, router i firewall na Solaris 11

Praca dyplomowa napisana
pod kierunkiem
dr. Mariusza Żynela

Białystok 2017

Z całego serca chciałbym serdecznie podziękować
promotorowi dr. Mariuszowi Żynelowi za cenną pomoc,
opiekę, istotne rady i życzliwość w trakcie realizacji
niniejszej pracy licencjackiej.
Dzięki tej pracy nauczyłem się wiele pokory
oraz poznałem nowy system operacyjny,
na którym można więcej niż się wydaje

Fabian Neumann

Spis treści

Wstęp

Cel pracy

Głównym celem mojej pracy jest skonfigurowanie serwera działającego pod kontrolą systemu operacyjnego Solaris 11 tak, aby pełnił rolę routera dla kilku sieci. Stąd też nazwa *router wieloadresowy*, gdyż będąc w kilku różnych sieciach, nasz serwer wyposażony będzie w odpowiednią ilość interfejsów sieciowych, z których każdy będzie wpięty do osobnej sieci i będzie miał w niej swój adres.

W praktyce router poza trasowaniem pakietów pełni w sieci prywatnej kilka dodatkowych funkcji związanych z dostępem do sieci Internet. Gdy mówimy o sieciach prywatnych, to właściwie integralną funkcją routera jest translacja adresów, czyli NAT. Router, jako urządzenie znajdujące się na styku różnych sieci, jest idealnym miejscem na uruchomienie zapory firewall. Przy okazji, dla usprawnienia zarządzania siecią prywatną, warto uruchomić system automatycznej konfiguracji hostów, czyli DHCP, oraz swój własny, prywatny DNS, nawet jeśli nie posiadamy domen, które chcielibyśmy obsługiwać. Doskonałym miejscem na uruchomienie DHCP i DNS jest właśnie router jako centralny system do zarządzania i kontroli sieci prywatnych. Oszczędzamy w ten sposób na sprzęcie i upraszcza się administracja sieciami prywatnymi. Przy obecnej mocy sprzętu nie ma co się przejmować wydajnością routera, bo nie budujemy sieci dla Pentagonu a oprogramowanie IPFilter, ISC DHCP, czy ISC Bind nie obciąża zbytnio maszyny.

Zasadniczym problemem postawionym w tej pracy jest taka konfiguracja serwera DNS na routerze, aby jedna jego instalacja obsługiwała kilka sieci prywatnych i generowała różne wyniki w zależności od sieci, z jakiej pochodzi zapytanie. W przypadku orprogramowania ISC Bind (por. [2]), czyli chyba najbardziej popularnej implementacji serwera DNS, cel ten można osiągnąć stosując tak zwane *widoki*.

Trochę historii

System Solaris nie jest i chyba nigdy nie był bardzo popularnym systemem, nawet wśród Unixów. Posiada jednak swoją długą i dość burzliwą historię oraz może niezbyt liczną grupę użytkowników, ale są to osoby o wysokich kwalifikacjach w branży IT. Solaris używany jest tam, gdzie bezkompromisowe bezpieczeństwo i stabilność są najwyższym priorytetem, czyli między innymi w przemyśle lotniczym, kosmicznym i w wojsku.

Z naszego punktu widzenia w historii Solaris istotne są dwa wydarzenia: udostępnienie kodu źródłowego w 2004 roku jako systemu OpenSolaris (por. [3]) rozpowszechnianego na zasadach otwartej licencji oraz przejęcie w 2010 roku firmy Sun Microsystems – twórcy Solaris – przez firmę Oracle. Dzięki pierwszemu zdarzeniu mamy dostęp do kodu systemu od wersji 10 wzwyż. Natomiast drugie wydarzenie spowodowało, że Solaris od wersji 11 został ponownie skomercjalizowany i za darmo jest dostępny tylko do użytku prywatnego. Jednocześnie powstało kilka dytsrybucji opartych o jądro Solaris rozwijane w ramach otwartego projektu zwanego Illumos (por. [4]), spadkobiercy OpenSolaris. Wśród najbardziej liczących się dystrubucji jest: OpenIndiana, SmartOS i XStreamOS.

Między dystrybucjami Solaris są oczywiście pewne różnice, ale przy realizacji mojego zadania nie powinny one mieć większego znaczenia. Aby moje rozwiązania można było zastosować także w sytemach komercyjnych wybrałem najbardziej aktywną i dlatego też najbardziej popularną, otwartą dystrybucję OpenIndiana (por. [5]).

Zakres prac

Zadaniem praktycznym postawionym w tej pracy jest opracowanie konfiguracji routera działającego na Solaris 11, który znajduje się w dwóch sieciach WAN i dwóch sieciach LAN. Tych sieci może być teoretycznie więcej, ale istotna różnica w konfiguracji routera jest między jedną a wieloma sieciami WAN, czy też LAN. Aby nadać zadaniu praktyczny sens rozważam sytuację dość typową w przypadku niedużej firmy z branży IT:

WAN-1: Szerokopasmowe, podstawowe łącze od dostawcy internetowego A. W tej sieci udostępniamy krytyczne dla działania naszej firmy usługi na przykład HTTP, FTP, DNS, SMTP, IMAP, POP. To łącze zapewnia dostęp do sieci Internet dla zaufanej sieci prywatnej LAN-1.

WAN-2: Szerokopasmowe łącze zapasowe od dostawcy internetowego B. To łącze zapewnia dostęp do sieci Internet dla niezaufanej sieci prywatnej LAN-2 oraz jako failover w przypadku awarii WAN-1.

LAN-1: Zaufana, lokalna sieć biurowa.

LAN-2: Niezaufana, lokalna sieć serwisowa.

Uruchomienie, przetestowanie i diagnostyka takiego schematu sieci wymaga użycia przynajmniej 5 różnych maszyn: routera oraz po jednej maszynie w każdej z sieci poza routerem. Ponadto maszyna pełniąca rolę routera musi posiadać 4 karty sieciowe. Ponieważ mam ograniczone możliwości sprzętowe całe zadanie zrealizuję na jednej fizycznej maszynie z zainstalowanym oprogramowaniem VirtualBox (por. [6]) pozwalającym na wirtualizację niezbędnych komputerów i sprzętu sieciowego. W tej sytuacji zadanie praktyczne to:

1. instalacja maszyn wirtualnych,
2. konfiguracja sieci wirtualnych na hoście i odpowiednich interfejsów sieciowych w maszynach wirtualnych,
3. konfiguracja IPFilter na routerze: policy/source routing, firewall i NAT,
4. instalacja i konfiguracja serwerów DHCP oraz DNS na routerze,
5. opracowanie skryptu do przełączenia failover łączy sieci rozległych.

Problematyka zagadnienia

O ile ogólne sposoby konfigurowania IPFilter w celu uruchomienia translacji adresów, czy zapory firewall są dość dobrze rozpracowane, udokumentowane i opatrzone przykładami, to jeśli chodzi o reguły tak zwane *policy* albo *source* routing, czyli reguły trasowania pakietów oparte o adres źródłowy, dokumentacja poza kilkoma zdaniami w instrukcji obsługi (manualu) nie istnieje. W tym wypadku wykonanie zadania będzie wymagać eksperymentowania. Opracownie ostatecznej konfiguracji polegać będzie na metodzie prób i błędów.

Instalacja i konfiguracja DHCP oraz DNS powinna przebiegać standardowo, choć tutaj zdecydowana większość dokumentacji dotyczy systemów Linuxowych i zapewne pojawi się odmienność systemu operacyjnego.

Rozdział 1

Podstawy teoretyczne

1.1 DNS - system nazw domen

Rozszyfrowanie typowego, trzyliterowego skrótu DNS niestety niewiele powie osobie, która nie zna podstaw funkcjonowania sieci TCP/IP. Skrót ten oznacza Domain Name System, czyli z angielskiego *system nazw domen* (por. [1]). DNS jest jednym z filarów sieci Internet - używa go każdy. Spróbujmy wyjaśnić co kryje się pod nazwą DNS, tak aby było to zrozumiałe dla każdego użytkownika Internetu.

1.1.1 Zasada działania

W sieci Internet mamy destępne całe mnóstwo przeróżnych usług. Każda z nich używa swego specyficznego protokołu do komunikacji klient-serwer. Internet powszechnie kojarzony jest z dokładnie jedną z tych usług, a mianowicie ze stronami WWW (ang. World Wide Web). Rzeczywiście używany przez nie protokół HTTP (ang. Hyper-Text Transfer Protocol) dominuje w sieci Internet. Ze względu na swą prostotę i uniwersalność służy nie tylko do przesłania stron. Musimy zdać sobie sprawę, że Internet to nie WWW. Poza stronami WWW w sieci Internet mamy takie usługi jak:

- poczta elektroniczna SMTP,
- serwery plików FTP,
- serwery czasu NTP,
- telefonię VoIP,
- telewizję IPTV,
- komunikatory IRC,
- grupy dyskusyjne NNTP

i wiele innych. Każda z tych usług wymaga serwera, czyli maszyny na której jest uruchomiona i która udostępnia swoje zasoby. Aby możliwe było połączenie z taką maszyną musi ona mieć swój unikalny adres. W sieci Internet, która oparta została na protokole TCP/IP, ten adres nazywa się *adresem IP*. Warto tu zwrócić uwagę, że na jednej maszynie może być uruchomionych kilka różnych usług, a na jednym serwerze WWW może być umieszczonych wiele różnych stron. Nie zmienia to jednak faktu, że każda usługa, każda strona WWW związana jest z konkretnym adresem IP. Adres IP w protokole TCP/IP wersji 4 to 4 bajty, czyli liczby z zakresu od 0 do 255, rozdzielone kropkami, np. `212.33.71.77`. Gdybyśmy musieli pamiętać adresy naszych ulubionych stron w tej postaci mielibyśmy nie lada kłopot. Ciężko jest skojarzyć ciąg liczb z konkretnym serwisem. Ile numerów telefonów jesteśmy w stanie zapamiętać? To jest ten sam problem.

DNS całkowicie rozwiązuje problem pamiętania i kojarzenia adresów IP. Zamiast adresu IP `212.33.71.77` łatwiej zapamiętać jest adres domenowy `uwb.edu.pl`. Przetłumaczeniem jednego na drugi zajmuje się właśnie DNS. System DNS ułatwia korzystanie z Internetu w tak dużym stopniu, że nawet osoby nieznające się na komputerach bez problemu radzą sobie z przeglądaniem stron internetowych w domowym zaciszu.

1.1.2 Rozproszona baza danych

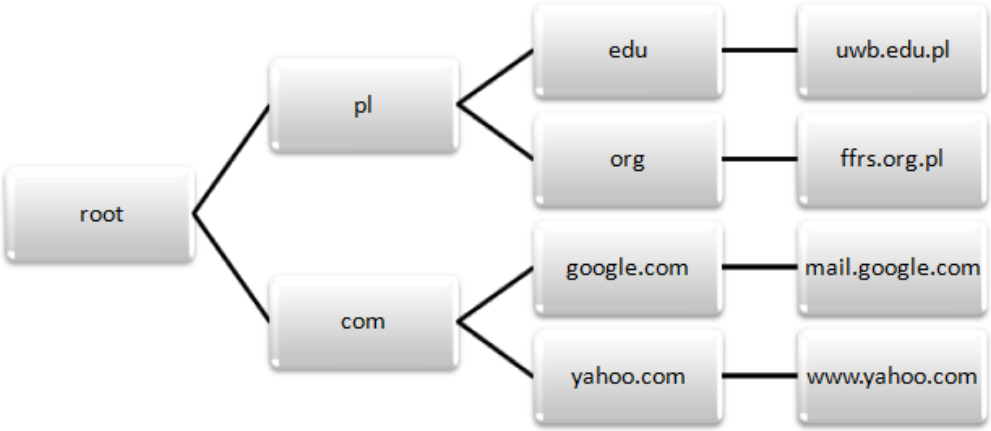
Pod nazwą DNS kryje się tak naprawdę kilka różnych rzeczy. Po pierwsze DNS narzucając pewną hierarchię zapewnia porządek w nazewnictwie serwisów. Dlatego też same nazwy domenowe tworzą pewien system o strukturze drzewiastej. Po drugie DNS to rozproszona baza danych adresów IP powiązanych z nazwami domenowymi. Poprzez DNS często też rozumie się serwer, który jest elementem tej rozproszonej bazy danych i na którym jest zgromadzona informacja o adresach IP wybranych domen. Chyba najrzadziej, ale jednak też, przez DNS rozumie się klienta, albo *resolver*, czyli oprogramowanie pozwalające pobrać z bazy danych DNS adres IP na podstawie adresu domenowego, albo na odwrót. No i na koniec DNS to także protokół pozwalający komputerom i programom dogadywać się ze sobą. Precyzuje on jak ma być sformułowane zapytanie do bazy DNS i jak wygląda ewentualna odpowiedź. Widać zatem, że DNS to bardzo szerokie zagadnienie i sam skrót mieści w sobie wiele pojęć.

Jako twórcę DNS uważa się Jona Postela, który w latach 1969-73 wraz z innymi opracował podstawowe koncepcje sieci TCP/IP. W roku 1982, kiedy powstawała dopiero sieć Internet i nikt nie przypuszczał, że rozrośnie się ona do obecnych rozmiarów, wszystkie adresy IP spisane były w jednym pliku `hosts.txt`. Plik ten aktualizowano centralnie w Stanford Research Institute i rozsyłano do administratorów sieci. Taka forma przechowywnia i dystrybucji adresów serwerów była bardzo niewydajna i łatwo było o błędy. Pozostałością tego systemu nazw jest plik `/etc/hosts` w systemach Unixowych. Sytuacja

zmieniła się, gdy opublikowano specyfikację DNS w roku 1983 i zaimplementowano pierwszy, zgodny z nią, serwer nazw w 1984 roku. Był to BIND od Berkeley Internet Name Domain. Dzięki skalowalności system DNS przetrwał w niezmienionej formie do dzisiaj.

1.1.3 Hierarchia domen

Hierarchia domen jest ważną zasadą, ponieważ domena jest przypisana do odpowiedniej dla siebie kategorii, dlatego też nie mamy w Internecie zbytniej swobody w tworzeniu nazw domen. Dzięki temu możemy też łatwo określić do jakiej organizacji należy dana strona albo usługa. Dzięki hierarchicznej strukturze domen w Internecie zachowany jest porządek. Adresy stron nie mieszają się ze sobą oraz nie są tak łatwe do podrobienia przez osoby próbujące wyłudzić wrażliwe dane. Poniższy schemat przedstawia kilka nazw domenowych oraz ich hierarchię.



Rysunek 1.1: Przykład hierarchii domen.

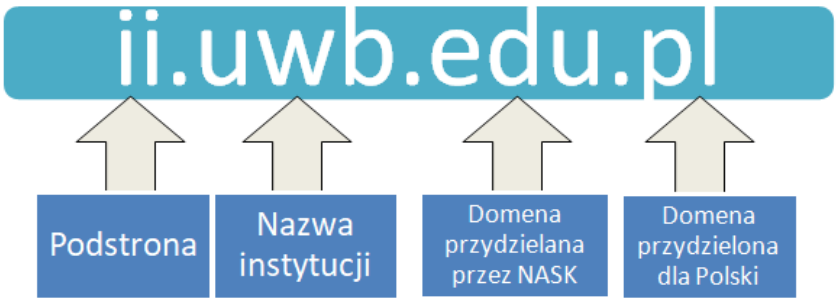
Wśród hierarchi domen poza głównym `root` ważne są też domeny TLD (ang. Top-Level Domain). Tworzeniem i zarządzaniem domen TLD zajmują się dwie instytucje: IANA (Internet Assigned Numbers Authority) oraz ICANN (The Internet Corporation for Assigned Names and Numbers), które zarządzają w zasadzie całą siecią Internet. Domeny najwyższego poziomu pozwalają bezbłędnie określić kraj z jakiego domena pochodzi, rodzaj instytucji czy też ukierunkowanie domeny. Dzięki takiemu podziałowi można w prosty sposób odróżnić domeny instytucji państwowych od domen należących do systemu oświaty. Z racji dużego zapotrzebowania na domeny oraz chęci utrzymania porządu w całej hierarchi nastąpiła potrzeba podziału TLD na dwie różne kategorie (por. [7]):

- krajowe, w skrócie ccTLD (ang. country code TLD), oraz
- funkcjonalne, w skrócie gTLD (ang. generic TLD).

Do domen krajowych należą domeny, w których ostatnia część składni adresu domenowego posiada skrót od państwa pod którym są zarejestrowane. Jako przykład można dać domeny z końcówką `.uk`, należące do Wielkiej Brytani lub też `.de`, należące do naszych zachodnich sąsiadów Niemiec. Niektóre z państw posiadają ciekawe skróty. Na przykład `.tv` należy do Tuvalu i jest bardzo często wykorzystywany do tworzenia stron osób nagrywających materiały na popularny serwis `YouTube`. Przykładem może tu być strona `http://5sposobowna.tv/`. Drugim krajem jest Dżibuti, który posiada końcówkę `.dj` używaną przez wiele wytwórni muzycznych oraz DJ-ów w Polsce.

Drugim rodzajem domen TLD są domeny funkcjonalne. Należą do nich na przykład `.edu`, `.com`, `.net`. Są to domeny stosowane zazwyczaj dla Stanów Zjednoczonych. Pozostałe kraje posiadają swoje domeny funkcjonalne które są przypisane pod domeną krajową. Przykładami takich domen są `.edu.pl`, `.org.pl`, `.com.pl`. Domeny funkcjonalne krajowe są często stosowane przez organizacje państwowe oraz przez uczelnie i szkoły. W Polsce instytucją odpowiedzialną za nadzór jest NASK (Naukowa i Akademicka Sieć Komputerowa), która znajduje się w Warszawie. Jej zadaniem jest kontrolowanie przyznawania domen dla odpowiednich instytucji bądź firm.

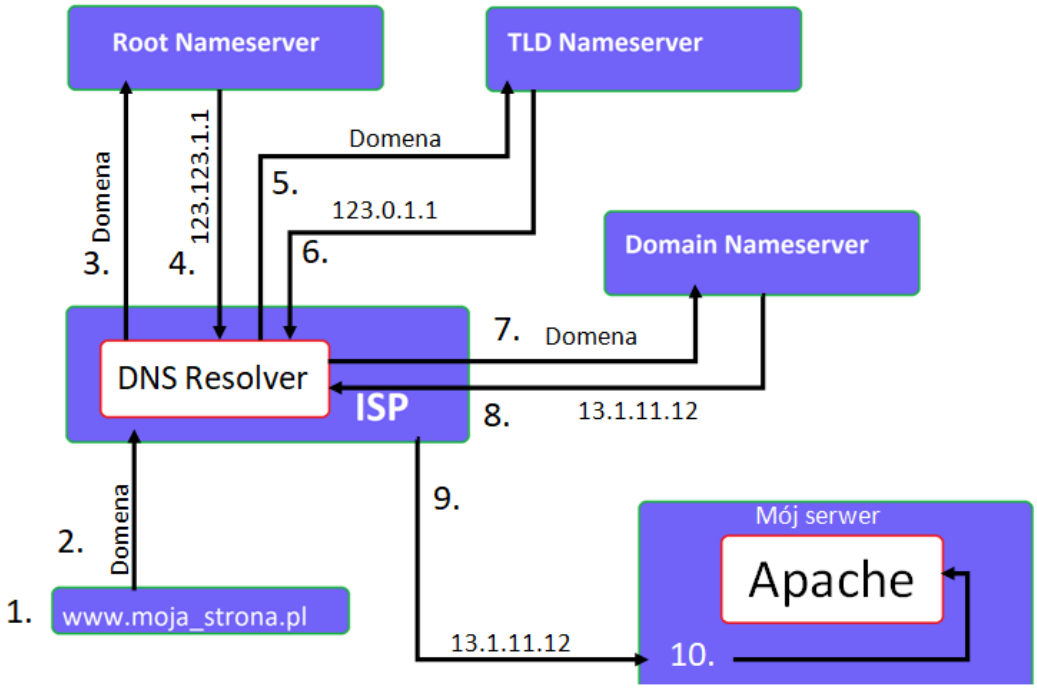
Jako przykład domeny funkcjonalnej, edukacyjnej pod domeną krajową jest domena `uwb.edu.pl` należąca do Uniwersytetu w Białymstoku. Aby pokazać z czego składa sie nazwa domenowa można wziąć pod uwagę stronę naszego Instytutu Informatyki. Poszczególne składowe adresu domenowego w tym wypadku mają znaczenie jak na rysunku 1.2.



Rysunek 1.2: Hierarchia adresu domenowego `ii.uwb.edu.pl`.

1.1.4 Rozwiązywanie nazw domenowych

Z uwagi na hierarchiczną budowę systemu nazw DNS do zamiany adresu domenowego na adres IP stosowany jest odpowiedni algorytm, zwany *rozwiązywaniem nazw* (ang. name resolution). Przykładowy schemat działania tego algorytmu znajduje się na rysunku 1.3.



Rysunek 1.3: Schemat rozwiązywania nazw w systemie DNS.

Prześledźmy działanie tego algorytmu na przykładzie (patrz rys. 1.3):

1. Wpisanie do przeglądarki internetowej adresu strony jaka nas interesuje na przykład `www.moja_strona.pl`.
2. Wysłanie zapytania o domenę do programu pytającego (ang. DNS resolver) serwery o adres domeny docelowej.
3. Program pyta się głównego serwera zarządzającego `TLD` o adres IP domeny docelowej.
4. Uzyskanie przez program adresu IP do `TLD` zarządzającego serwerem danej domeny.
5. Zapytanie się przez program o adres IP serwera zarządzającego domeną.
6. Uzyskanie adresu IP do serwera zarządzającego domeną od `serwera TLD`.

7. Zapytanie serwera zarządzającego domeną o adres IP do domeny docelowej.
8. Uzyskanie adresu IP domeny docelowej.
9. Zgłoszenie się do serwera, na którym jest umieszczona docelowa domena.
10. Dostanie się na docelową stronę WWW poprzez Apache'a uruchomionego na serwerze obsługującym intresującą nas domenę.

1.1.5 Narzędzia

Zapytania o adres IP przeglądarka i inne aplikacje użytkowe wykonują same za pośrednictwem resolvera w systemie operacyjnym. W końcu co obchodzą zwykłego użytkownika jakieś tam adresy IP. Jeśli jednak obchodzą, to można je sprawdzić na żądanie. W systemach Unixowych i Linuxie jest do tego narzędzie `host`:

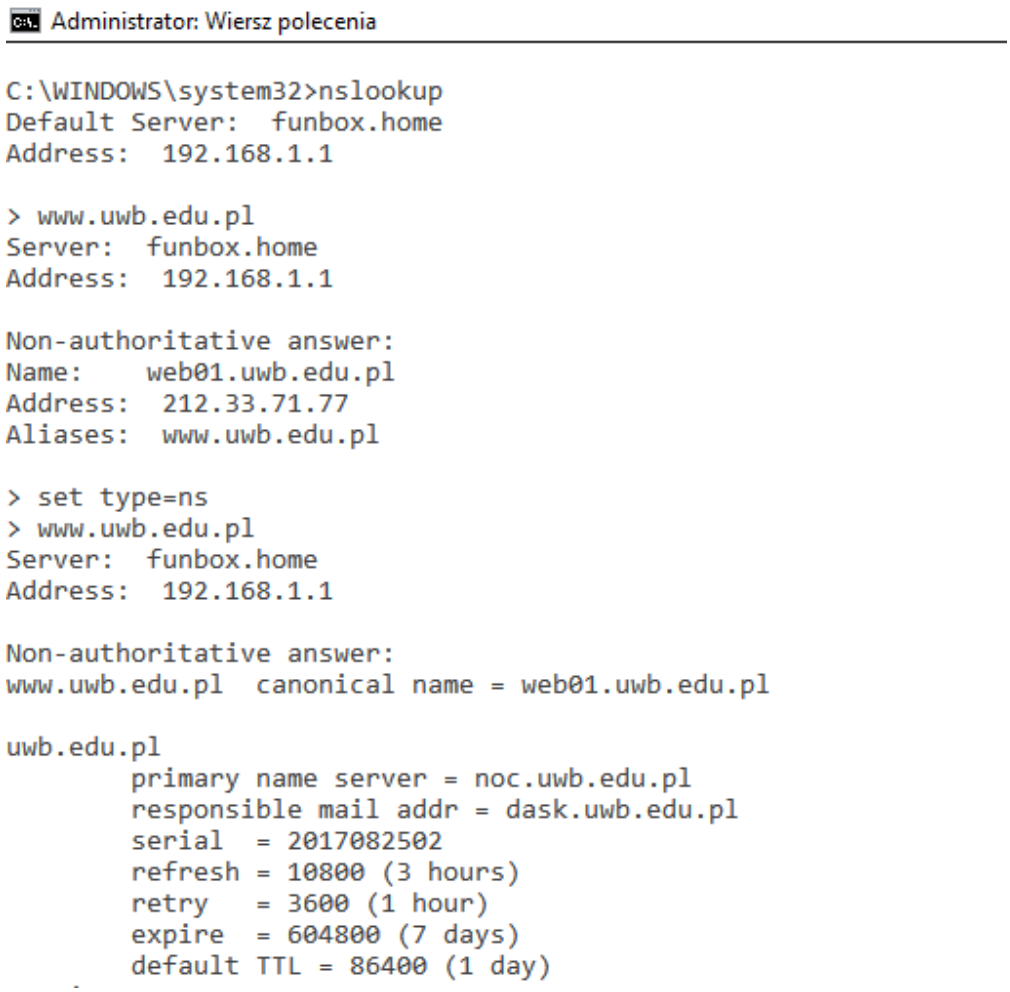
```
indiana$ host uwb.edu.pl
uwb.edu.pl has address 212.33.71.77
uwb.edu.pl mail is handled by 10 mx.uwb.edu.pl.
```

oraz nieco bardziej wygadane `dig`:

```
indiana$ dig uwb.edu.pl

; <<>> DiG 9.3.6-P1 <<>> uwb.edu.pl
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1880
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;uwb.edu.pl.                    IN   A

;; ANSWER SECTION:
uwb.edu.pl.             78732   IN   A    212.33.71.77

;; AUTHORITY SECTION:
uwb.edu.pl.             54833   IN   NS   dns.man.bialystok.pl.
uwb.edu.pl.             54833   IN   NS   moc.uwb.edu.pl.
uwb.edu.pl.             54833   IN   NS   master.man.bialystok.pl.
uwb.edu.pl.             54833   IN   NS   noc.uwb.edu.pl.

;; ADDITIONAL SECTION:
dns.man.bialystok.pl.   50288   IN   A    212.33.64.2
moc.uwb.edu.pl.         54833   IN   A    212.33.71.72
```

```
noc.uwb.edu.pl.          54833   IN   A    212.33.71.71
master.man.bialystok.pl. 69113   IN   A    212.33.64.18

;; Query time: 1 msec
;; SERVER: 172.16.0.1#53(172.16.0.1)
;; WHEN: Sun Aug 27 13:44:56 2017
;; MSG SIZE  rcvd: 197
```

Progarm `dig`, jak widać, poza adresem IP, o który pytamy, zdradza w jakich serwerach DNS (wiersze oznaczone jako rekordy `NS`) znajduje się ta informacja. W systemach Windows mamy do dyspozycji polecenie `nslookup`:



Rysunek 1.4: `nslookup` w systemie Windows.

Czasem interesujący może być tak zwany odwrotny DNS (ang. reverse DNS), gdy znamy adres IP a nie wiemy do jakiej domeny należy. Rozbudowany `dig` załatwia tutaj sprawę:

```
indiana$ dig -x 212.33.73.194

; <<>> DiG 9.3.6-P1 <<>> -x 212.33.73.194
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1954
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 4, ADDITIONAL: 4

;; QUESTION SECTION:
;194.73.33.212.in-addr.arpa.    IN      PTR

;; ANSWER SECTION:
194.73.33.212.in-addr.arpa. 86400 IN    PTR     math.uwb.edu.pl.

;; AUTHORITY SECTION:
73.33.212.in-addr.arpa. 86400   IN      NS      noc.uwb.edu.pl.
73.33.212.in-addr.arpa. 86400   IN      NS      moc.uwb.edu.pl.
73.33.212.in-addr.arpa. 86400   IN      NS      dns.man.bialystok.pl.
73.33.212.in-addr.arpa. 86400   IN      NS      master.man.bialystok.pl.

;; ADDITIONAL SECTION:
dns.man.bialystok.pl.   49892   IN      A       212.33.64.2
moc.uwb.edu.pl.         54437   IN      A       212.33.71.72
noc.uwb.edu.pl.         54437   IN      A       212.33.71.71
master.man.bialystok.pl. 68717  IN      A       212.33.64.18

;; Query time: 21 msec
;; SERVER: 172.16.0.1#53(172.16.0.1)
;; WHEN: Sun Aug 27 13:51:32 2017
;; MSG SIZE  rcvd: 226
```

Jak widać adres `212.33.73.194` oznacza hosta `math.uwb.edu.pl`.

1.1.6 Strefy

Niestety nie każdy z użytkowników Internetu wie czym są, oraz jak działają strefy (ang. zones) w DNS. Często są one mylone z samymi domenami.

Domeny mogą być bardzo duże w sensie wysokości drzewa nazw. To powoduje, że zarządzanie nimi może być kłopotliwe. Wyobraźmy sobie, że cała domena `edu.pl` jest umieszczona na jednym serwerze nazw, w skrócie NS (ang. Name Server). Administrator takiego serwera byłby nieustannie nękany prośbami o aktualizacje ze wszystkich uczelni w Polsce. Dlatego taką domenę dzieli się na poddomeny i oddelegowuje ich zarządzanie do właścicieli tych poddomen. Zabieg delegacji można wykonać ponownie, dla głębiej zagnieżdżonych poddomen w systemie nazw DNS.

Strefa DNS zaczyna się w punkcie delegacji, czyli na poddomenie i kończy albo na liściach drzewa nazw, albo w miejscu, gdzie zaczyna się inna strefa. Inaczej mówiąc strefa DNS jest to pakiet informacji o danej domenie jakie posiada konkretny serwer DNS.

Każda, nawet najprostsza domena musi posiadać kilka wpisów w serwerze DNS. Każdy taki wpis nazywa się *rekordem* i stanowi opis pewnej własności domeny. Podstawowe typy rekordów DNS są następujące:

SOA – Start Of Authority – rekord niezbędny dla każdej strefy. Zawiera on nazwę serwera danej strefy, adres kontaktowy administratora, numer seryjny, częstotliwość odświeżania, po jakim czasie wznowić pobranie danych, czas wygaśnięcia oraz czas życia danych strefy.

A – przypisuje adres IP do hosta (dla IPv6 jest to AAAA)

CNAME – Canonical Name – przypisuje alias do hosta, czyli inną jego nazwę. Alias może znajdować się w innej strefie.

NS - Name Server - informuje jaki serwer DNS należy zapytać, aby uzyskać dane z naszej strefy. Ten adres przechowywany jest w serwerze nazw delegującym naszą strefę.

MX - Mail Exchange – zawiera informacje o adresie serwera pocztowego danej domeny. Rekord MX należy poprzedzić liczbą określającą priorytet połączenia. Im mniejsza liczba tym wyższy priorytet. Jeśli mamy kilka serwerów pocztowych dla domeny, różnicując priorytety możemy decydować do którego z tych serwerów trafi poczta w pierwszej kolejności.

PTR – Pointer – przypisuje domenę do adresu IP. Zapewnia odwrotną funkcjonalność do rekordu typu A i realizuje tak zwany odwrotny DNS (reverse DNS).

TXT - zawiera dodatkowe informacje o domenie. Nie jest wymagany. Zwykle zawiera informacje dotyczące wpisów SPF, w którym są wpisane komputery upoważnione do wysyłania poczty.

Każda strefa zaczyna się od rekordu SOA, musi być w niej określony chociaż jeden host za pomocą rekordu A oraz musi być wskazany przynajmniej jedne serwer nazw za pomocą rekordu NS. Dobrym zwyczajem jest umieszczenie rekordu MX

1.1.7 Widoki

Widoki zostały wprowadzone w BIND 9. Pozwalają zaimplementować tak zwany *Split DNS* oraz *Stealth DNS*. Pod tymi tajemniczo brzmiącymi nazwami kryje się bardzo prosta idea. Chodzi o to by serwer DNS udzielał zróżnicowanych odpowiedzi w zależności od tego kto go pyta. Na ogół odróżnia się klientów z sieci prywatnej od tych z sieci publicznej.

Czasem różnicuje się klientów z uwagi na ich położenie geograficzne, które jest odzwierciedlone w ich adresach IP. Na przykład możemy mieć serwis WWW replikowany na serwerach w Europie, Azji i Ameryce. Aby równo rozłożyć obciążenie serwerów i skrócić trasę pakietów możemy klientom z różnych kontynentów zwracać adres bliższego serwera.

Jeśli mamy bazę klienów, których identyfikujemy po adresach IP, to pod jednym adresem domenowym możemy udostępniać różne wersje swojej aplikacji WWW albo sklep on-nline z produktami w różych cenach dla różnych grup klientów.

Generalnie, jak widać są dobre powody na wdrożenie Split lub Stealth DNS. Zarówno Split jak i Stealth DNS można zaimplementować bez widoków opierając się jedynie o dyrektywę `allow-query`, która pozwala spośród wszystkich pytających odfiltrować tych uprawnionych. Różnica między Split a Stealth jest w zasadzie filozoficzna. O ile Split polega na rozdzieleniu klientów DNS i udzielaniu im zróżnicowanych odpowiedzi, to Stealth polega na ukryciu serwera DNS tak, że jest dostępny tylko z sieci prywatnej. Implementacja w obu wypadkach jest analogiczna.

Mówiąc o widokach w serwerze BIND należy doprecyzować, że w zależności od tego kto pyta, innna może być nie tylko sama odpowiedź serwera DNS, ale też może być dostępna inna funkcjonalność tego serwera.

Syntaktyka klauzuli widoku w pliku konfiguracyjnym `/etc/named.conf` jest bardzo prosta (por. [10]):

```
view "view_name" [class] {
  [ match-clients {  address_match_list } ; ]
  [ match-destinations { address_match_list } ; ]
  [ match-recursive-only { yes | no } ; ]
  // view statements
  // zone clauses
};
```

W widoku definiujemy zestaw stref (`zone clauses`), które będą widziane dla określonego zbioru klientów. Możemy tam także zadeklarować różne parametry (`view statements`), które wpływają na sposób udzielania odpowiedzi przez serwer DNS. Klient *pasuje* do widoku, znajduje się w tym zbiorze, gdy jego adres IP, czyli adres źródłowy zapytania, znajduje się w podsieci określonej w `match-clients` lub gdy adres docelowy, czyli adres na jaki przyszło zapytanie, znajduje się w podsieci określonej w `match-destinations`. Jeśli

nie zostały określone, domyślnie obie deklaracje zawierają `any`, czyli dopasowują każdy adres. Widok może być określony jako `match-recursive-only`, co oznacza, że wyłącznie zapytania rekurencyjne od dopasowanych klientów będą pasowały do tego widoku. Domyślnie widok nie jest `match-recursive-only`. Kolejność deklaracji widoków ma znaczenie. Zapytanie będzie rozwiązane w kontekście pierwszego widoku, który zostanie dopasowany do klienta.

Typowy podział DNS na prywatny (internal) i publiczny (external) wygląda tak:

```
view "internal" {
      match-clients { 192.168.0.0/24; };
      recursion yes;
      zone "domena.tld" {
            type master;
            file "internal/db.domena.tld";
      };
};
view "external" {
      match-clients { any; };
      recursion no;
      zone "domena.tld" {
           type master;
           file "external/db.domena.tld";
      };
};
```

Hosty z sieci prywatnej `192.168.0.0/24` mogą kierować zapytania rekurencyjne i odpowiedź o domenę `domena.tld` zostanie udzielona w oparciu o plik strefy `internal/db.domena.tld`:

```
; internal/private zone master file
domena.tld.  IN      SOA   ns.domena.tld. root.domena.tld. (
                           20170829   ; serial number
                           3h         ; refresh
                           15m        ; update retry
                           3w         ; expiry
                           3h         ; ttl
                           )
              IN      NS      ns1.domena.tld.
              IN      NS      ns2.domena.tld.
              IN      MX  10  mail.domena.tld.
; public hosts
ns1           IN      A       192.168.0.1
ns2           IN      A       192.168.0.2
mail          IN      A       192.168.0.3
www           IN      A       192.168.0.4
ftp           IN      A       192.168.0.5
```

```
; private hosts
biuro          IN      A       192.168.0.6
serwis         IN      A       192.168.0.7
faktury        IN      A       192.168.0.8
```

Pozostałe hosty nie mogą kierować zapytań rekurencyjnych, a przy odpowiedzi o domenę `domena.tld` zostanie użyty plik strefy `external/db.domena.tld`:

```
; external/public zone master file
domena.tld.  IN      SOA   ns.domena.tld. root.domena.tld. (
                               20170828   ; serial number
                               3h         ; refresh
                               15m        ; update retry
                               3w         ; expiry
                               3h         ; ttl
                               )
             IN      NS      ns1.domena.tld.
             IN      NS      ns2.domena.tld.
             IN      MX  10  mail.domena.tld.
ns1          IN      A       212.33.73.194
ns2          IN      A       212.33.73.195
mail         IN      A       212.33.73.196
www          IN      A       212.33.73.197
ftp          IN      A       212.33.73.198
```

Jak widać, w sieci prywatnej, DNS odpowiada adresami prywatnymi. Ponadto w przykładowej domenie `domena.tld` znajdują się dodatkowe hosty `biuro`, `serwis` i `faktury`. Nie widać tych hostów z sieci publicznej. W tym sensie nasz DNS jest Stealth. Poza tym na pytania z sieci publicznej nasz DNS odpowie adresami publicznymi. To akurat oznacza, że mamy Split DNS.

1.1.8 Blokowanie reklam

Kolejną ważną zaletą posiadania własnego serwera DNS jest możliwość blokowania reklam oraz niechcianych skryptów. W tym celu należy utworzyć oddzielny plik, w moim przypadku będzie to plik `/etc/reklamy.conf`, w którym to będę skrupulatnie umieszczał domeny będące źródłem reklam i skryptów. Uaktualniana, gotowa do użycia lista takich domen jest do pobrania z:

```
http://pgl.yoyo.org/adservers/serverlist.php
```

W głównym pliku konfiguracyjnym `/etc/named.conf` należy dopisać

```
include "/etc/reklamy.conf";
```

Plik `/etc/reklamy.conf` zawiera deklaracje stref. Oto jego fragment:

```
zone "adventory.com" { type master; notify no; file "null.zone.file"; };
zone "advert.bayarea.com" { type master; notify no; file "null.zone.file"; };
zone "adverticum.com" { type master; notify no; file "null.zone.file"; };
zone "adverticum.net" { type master; notify no; file "null.zone.file"; };
zone "adverticus.de" { type master; notify no; file "null.zone.file"; };
zone "advertise.com" { type master; notify no; file "null.zone.file"; };
```

Jak widać, deklaracja wszystkich stref znajduje się w jednym, wspólnym pliku o nazwie `null.zone.file`. Zawartość tego pliku może wyglądać tak:

```
domena.tld.  IN      SOA   ns.domena.tld. root.domena.tld. (
                                20170828   ; serial number
                                3h         ; refresh
                                15m        ; update retry
                                3w         ; expiry
                                3h         ; ttl
                                )
             IN      NS      ns1.domena.tld.
@            IN      A       127.0.0.1
*            IN      A       127.0.0.1
```

Dwie ostatnie linijki określają adres hosta reklamodawcy jako `127.0.0.1`, co oznacza, że niepożądana reklama lub skrypt będzie pobierana z naszego lokalnego serwera HTTP. Jedyny minus tego rozwiązania jest taki, że nasz serwer będzie generował stosy błędów 404 Not found. Jeśli nie mamy uruchomionego lokalnie serwera HTTP, to zapytania zakończą się przekroczeniem czasu oczekiwania. Dlatego też, jeśli z naszego DNS korzysta więcej klientów, to lepiej adres localhost `127.0.0.1` zastąpić adresem hosta z uruchomionym serwerem HTTP.

Dzięki małemu oszustwu na poziomie DNS pobranie reklamy lub niechcianego skryptu zakończy się niepowodzeniem. Takim oto prostym sposobem prywatny DNS pomaga, aby odwiedzane strony internetowe działały jak najbardziej płynnie oraz były wolne od uporczywych reklam.

1.2 Router - trasowanie pakietów

W sieci TCP/IP mamy dwie klasy urządzeń: hosty i routery. Różnica polega na tym, że te pierwsze nie zmieniają trasy pakietów z danymi, które do nich docierają. Dosłownie z angielskiego *router*, lub bardziej po polsku *ruter*, czyli *trasownik* to aktywne urządzenie sieciowe, wyposażone w więcej niż jeden interfejs sieciowy, znajdujące się na styku dwóch lub więcej sieci, które decyduje o trasie pakietów trafiających do niego.

1.2.1 Protokoły trasowania

Ruter pracuje w trzeciej warstwie, czyli w warstwie sieciowej, modelu OSI (por. rys. 1.5).

Rysunek 1.5: Model OSI.

Podstawowa idea routera jest taka, aby na podstawie docelowego adresu IP, dany pakiet skierować do następnego routera znajdującego się możliwie najbliżej celu, chyba że droga pakietu dobiegła końca i można skierować go bezpośrednio do odbiornika. Określenie *najbliżej* w kontekście sieci TCP/IP nie jest jednak jednoznaczne. Decydująca w kwestii *odległości* jest *metryka*. Jest to wartość zapisana w tablicy trasowania (ang. routing table) obok trasy ścieżki do konkretnego obszaru sieci. Im wartość ta jest niższa, tym ścieżka jest krótsza. Metryką może być ilość routerów do miejsca docelowego, wskaźnik przepustowości danej trasy, albo stopień ilości możliwych błędów, które mogą pojawić się po drodze. O tym jak jest ustalana trasa i jak jest dobierana metryka decyduje zastosowany protokół, a jest ich kilka.

- RIP – Routing Information Protocol,
- IGRP – Interior Gateway Routing Protocol,
- EIGRP – Enhanced Interior Gateway Routing Protocol,

- OSPF – Open Shortest Path First,
- IS-IS – Intermediate System to Intermediate System.

Pierwszy z nich – RIP – jako metryki używa ilości przeskoków (ang. hops), inaczej mówiąc, ilości routerów po drodze. Protokoły IGRP i EIGRP wyliczają metrykę z szerokości pasma, obciążenia, opóźnienia i niezawodności. Ostatnie dwa protokoły: OSPF oraz IS-IS, oparte są o algorytm Dijkstry do znajdowania najlepszej trasy do transportu pakietów danych.

Każde urządzenie pełniące rolę routera w sieci ma obowiązek obniżyć wartość TTL (ang. Time To Live), czyli czas życia pakietu, o 1 punkt. W przypadku, gdy wartość życia pakietu spadanie do 0 router odrzuca ten pakiet i wysyła do nadawcy informację o jego wygaśnięciu za pomocą protokołu ICMP. Zalecana, domyślna wartość TTL to 64, przy czym wartość maksymalna to 255. Jak widać nie można w sieci uzyskać efektu zapętlenia i krążenia pakietów bez końca.

1.2.2 Hardware

Ruter to zwykle dedykowane urządzenie. Rolę routera może także pełnić komputer z odpowiednim oprogramowaniem. W zasadzie większość dedykowanych urządzeń to zwykle bardzo uproszczony komputer oparty o procesor ARM z okrojonym Linuxem lub innym systemem operacyjnym oraz specjalnym oprogramowaniem zaprojektowany do wykonywania jednego zadania.

O ile dedykowane routery wyższej klasy takie jak Juniper czy Cisco wykonują wyłącznie zadanie trasowania, to routery klasy SOHO (Small Office - Home Office) zwykle wykonują przy okazji wiele innych zadań. Ułatwiają w ten sposób życie mniej zaawansowanym użytkownikom i pozwalają zaoszczędzić na sprzęcie bo w jednym urządzeniu mamy od razu NAT, DHCP, punkt dostępowy Wi-Fi, mniej lub bardziej wymyślne zapory z kontrolą rodzicielską włącznie, filtrowanie MAC, współdzielenie plików i drukarek poprzez USB i wiele innych wygodnych funkcji. Prowadzi to niestety do pewnego uproszczenia, że router to mała skrzynka z antenkami, do której z jednej strony podpinamy kabelek od dostawcy internetowego, z drugiej strony kabelki do naszych komputerów, albo łączymy się jednym i drugim laptopem oraz smartfonami po Wi-Fi no i wszystku pięknie hula. Tak, tylko pamiętajmy że w tej małej skrzynce jest komputer i cała masa różnych technologii sieciowych.

Router możemy podłączyć do dostawcy internetowego na wiele sposobów. Jednak każdy rodzaj podłączenia ma swoje wady i zalety. Niestety nie stworzono jeszcze idealnego rozwiązania. Oto kilka z nich:

- Kabel Ethernet – zaletami tego podłączenia są przede wszystkim duże możliwości łącza, stabilność oraz odporność na zmienne warunki atmosferyczne. Wadą jest natomiast brak mobilności.

- Radiowo poprzez Wi-Fi – zaletą tego połączenia jest mobilność, aczkolwiek ograniczona do zasięgu sieci Wi-Fi. Niestety to połączenie ma dość liczne wady: brak stabilności łącza, brak odporności na zmienne warunki atmosferyczne, zakłócenia oraz przepustowość łącza która jest ograniczana po stronie karty sieciowej, oraz urządzenia rozgłaszającego.

- Radiowo poprzez sieć komórkową GSM – zaletą tego połączenia jest duża mobilność zapewniona przez mocno rozbudowaną sieć nadajników. Jednak w tym przypadku są też wady. Między innymi brak odporności na warunki atmosferyczne, brak pełnej stabilności, zakłócenia, przeciętne możliwości łącza.

W przypadku, gdy z komputera chcemy uczynić router łączący kilka sieci, dość istotnym ograniczeniem jest ilość interfejsów sieciowych. W średniej klasy komputerze PC mamy najwyżej dwa interfejsy RJ45 wbudowane na płycie głównej. Możemy wprawdzie włożyć dodatkowe interfejsy jako karty rozszerzeń, ale ilość slotów PCI/PCI-X/PCIe waha się w granicach 2-5 no i nie zawsze wszystkie możemy wykorzystać. Istnieją podwójne (ang. dual) i poczwórne (ang. quad) karty sieciowe, ale są one drogie.

Dysponując jednym łączem od dostawcy Internetu, aby utworzyć sieć z n komputerów podpiętych bezpośrednio do routera kablami Ethernet potrzebujemy $n + 1$ gniazd w routerze. Częściowo problem rozwiązuje sieć Wi-Fi, gdy nie mamy zbyt dużych wymagań co do przepustowości, odległość od routera nie jest za duża i po drodze nie mamy stropów i ścian. Rutery klasy SOHO najczęściej posiadają jedno łącze WAN do komunikacji z Internetem, 4 gniazda LAN do podłączenia komputerów za pomocą kabli oraz wbudowany punkt dostępowy (ang. access point) rozgłaszający poprzez sieć radiową Wi-Fi.

Innym znanym rodzajem routera jest router bezprzewodowy, który pracuje w sieci komórkowej. Odbiera on sygnały z Internetu za pomocą karty SIM, która jest umieszczona w urządzeniu oraz rozgłasza swoją sieć poprzez Wi-Fi. Zaletą tego typu urządzenia jest możliwość podłączenia do dowolnego komputera przy użyciu złącza USB, mobilność oraz wbudowany akumulator, który pozwala na pracę niezależnie od źródła stałego zasilania.

1.3 NAT - translacja adresów

Skrót NAT pochodzi od słów Network Adress Translation, co z angielskiego tłumaczone jest jako *translacja adresów sieciowych.*

1.3.1 Adresy publiczne i prywatne

Ze względu na szybko rosnącą ilość urządzeń z dostępem do sieci Internet zaczynamy zbliżać się do wyczerpania dostępnych adresów IP w 4 wersji protokołu TCP/IP. Przypomnijmy, że adres taki to 4 bajty, czyli w sumie 32

bity, co daje 2^{32} kombinacji, choć w praktyce wszystkich nie możemy wykorzystać do adresowania TCP/IP. Aby nie wyczerpać dostępnej puli adresów wymyślono właśnie technologę NAT. Nie wchodząc zbytnio w szczegóły dzięki NAT możemy niemal dowolną ilość komputerów w sieci prywatnej wypuścić do Internetu posiadając tylko jeden publiczny adres IP.

Mówiąc o NAT należy wyjaśnić, co tutaj rozumiemy przez adres publiczny i prywatny. Adres publiczny w sieci Internet to taki, dzięki któremu możliwa jest komunikacja ze wszystkimi jej użytkownikami. Pakiet danych wysłany z jednego publicznego adresu na inny powinien dotrzeć niezależnie od ilości routerów znajdujących się po drodze. Innymi słowy adresy publiczne to te, które są *rutowalne*, innymi słowy *osiągalne*. Dla odmiany adresy prywatne nie są rutowalne i są odrzucane przez routery w sieci Internet. OK, ale które adresy z całej puli są publiczne, a które prywatne? Otóż, wspominana wcześniej organizacja IANA, zarządzająca adresacją w sieci Internet, zarezerwowała kilka zakresów adresów IP jako prywatne. Znajdują się one w tabeli 1.1.

Zakres	Ilość adresów	Blok CIDR (maska)
10.0.0.0 – 10.255.255.255	16 777 216	10.0.0.0/8 (255.0.0.0)
172.16.0.0 – 172.31.255.255	1 048 576	172.16.0.0/12 (255.240.0.0)
192.168.0.0 – 192.168.255.255	65 536	192.168.0.0/16 (255.255.0.0)

Tabela 1.1: Adresy prywatne w sieci Internet.

Zgodnie z klasowym podziałem sieci mamy więc jedną całą sieć klasy A, 16 sieci klasy B i 256 sieci klasy C jako sieci prywatne i możemy ich spokojnie używać nie martwiąc się, że spowodujem konflikty adresów w sieci Internet. Na marginesie, w naszym kampusie używane są prywatne adresy z pierwszego zakresu z tabeli 1.1. NAT najczęściej jest stosowany w sieciach korporacyjnych, osiedlowych oraz instytucjach.

Technologia NAT nierozerwalnie związana jest z routerami dlatego, że translacja adresów w naturalny sposób wymusza zmianę trasy pakietów w sieciach TCP/IP, a mianowicie między interfejsem sieci prywatnej a interfejsem sieci publicznej. Tak więc translacja adresów z definicji odbywa się na routerze.

1.3.2 Zasada działania

Prosta matematyka jest jednak bezlitosna i jak się okazuje nie ma możliwości umieszczenia 16 777 216 komputerów za jednym publicznym adresem IP. Aby odpowiedzieć na pytanie dlaczego tak jest musimy nieco głębiej wejść w szczegóły technologii NAT.

Ci którzy choć trochę znają zasady przepływu pakietów w sieciach TCP/IP zaczynają się zastanawiać jak to możliwe, że gdy dwa komputery z sieci prywatnej wyślą w tym samym czasie dane do sieci publicznej i po drodze przetłumaczone zostaną ich różne adresy źródłowe na ten sam adres publiczny wewnątrz pakietów danych to oba dostaną odpowiedź, nic się nie zgubi i nie pokręci. Dzieję się tak dlatego, że NAT do identyfikacji pakietów wykorzystuje porty TCP. Gdy z sieci prywatnej, powiedzmy że z komputera o adresie `192.168.1.10`, z portu `24360`, trafia do NAT pakiet danych adresowany do `52.0.14.116` na port `80`. Jeśli nasz NAT ma publiczny adres powiedzmy `212.33.73.197`, to NAT zmienia adres źródłowy `192.168.1.10` na `212.33.73.197`, ale zmienia również port źródłowy z `32400` na pierwszy wolny jakim dysponuje, powiedzmy na `32800` i w swojej wewnętrznej tablicy NAT zapamiętuje jakiej zmiany dokonał. W tej tablicy mamy zatem dwie pary postaci: adres IP:port TCP, a mianowicie:

```
192.168.1.10:24360 -> 212.33.73.197:32400
```

W momencie, gdy z `52.0.14.116:80` wraca odpowiedź jest ona skierowana do `212.33.73.197:32400`. Wówczas NAT przepisuje adres i port docelowy w takim pakiecie na prywatny zgodnie z wpisem w swojej tablicy. Tak więc odpowiedź ta niezawodnie dostarczana jest do `192.168.1.10:24360` (por. rys. 1.6).

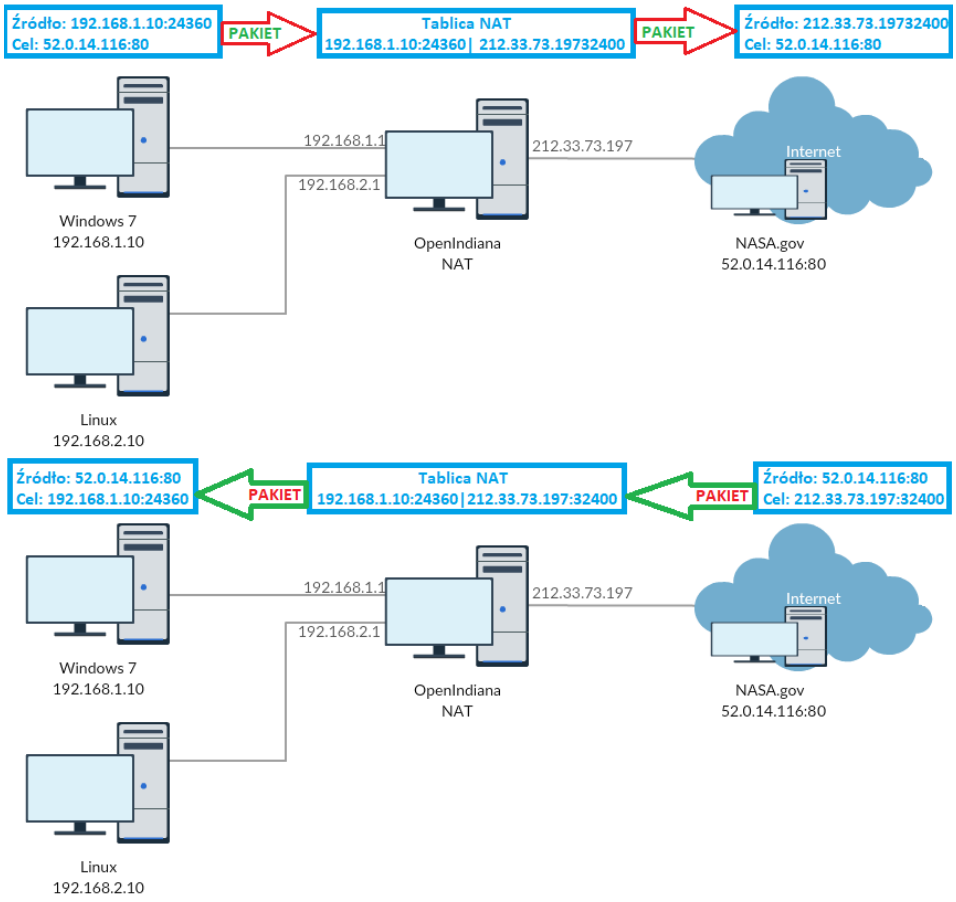
Widać teraz, że aby NAT mógł skutecznie pracować potrzebuje portów TCP. Ponieważ jest ich 2^{16}, czyli 65 536, to teoretycznie, mając jeden adres publiczny, może przetłumaczyć pakiety pochodzące z maksymalnie 65 536 komputerów. W praktyce ta ilość jest znacznie mniejsza bo część portów TCP, między innymi od 0 do 1024, jest zarezerwowana i nie może być użyta przez NAT.

Większe sieci prywatne mają do dyspozycji nie jeden lecz kilka adresów publicznych, które mogą przypisać do NAT. Zwiększa to dopuszczalną ilość komputerów w sieci prywatnej bo jest więcj możliwych par adres:port po stronie publicznej.

Reguły NAT można podzielić na dwie kategorie:

- SNAT (Source NAT) - translacja opisana przed chwilą. Polega na zamianie adresu IP urządzenia z LAN, które wysyła pakiet do sieci publicznej. NAT podmienia w tym pakiecie źródłowy adres IP na swój adres publiczny. Oczywiście podmianie ulega także port TCP.

- DNAT (Destination NAT) - jest to zmiana docelowego adresu IP na adres urządzenia w sieci LAN za routerem. DNAT jest po to, aby na przykład z sieci Internet/WAN dostać się do usługi uruchomionej na komputerze w sieci LAN za routerem. Translacji podlega docelowy adres IP, i ewentualnie port TCP, w pakiecie otrzymanym z sieci publicznej.



Rysunek 1.6: Schemat działania SNAT.

1.3.3 Wady i zalety

Podstawowy zysk wynikający z zastosowania technologii NAT polega na tym, że dzięki translacji adresów IP mamy możliwość podłączenia wielu komputerów z naszej sieci LAN do Internetu poprzez jeden adres IP ustawiony na routerze z NAT.

Druga ważna korzyść to ukrycie całej sieci LAN za routerem. Potencjalny atak z sieci publicznej może być jedynie wycelowany w sam router lub przemapowane poprzez DNAT pojedyncze porty TCP hostów w sieci LAN. Atakujący nie wie jednak, że te przemapowane porty nie należą do routera.

Kolejna zaleta to ułatwienie kontroli i zarządzania siecią. Przejście przez centralny punkt całego ruchu z sieci LAN do Internetu i na odwrót umożliwia monitorowanie tego ruchu i zarządzanie nim bez potrzeby inwestowania w drogie, w pełni zarządzalne przełączniki.

NAT zapewnia anonimowość użytkownikom hostów w sieci LAN. To, z punktu widzenia tych użytkowników, traktowane jest jako zaleta, ale utrud-

niona jest identyfikacja i namierzenie w przypadku różnego rodzaju naruszeń.

Dużym minusem dla użytkowników korzystających z komputera w sieci lokalnej jako serwera plików będzie utrudniony dostęp do tego serwera z zewnątrz. To ukrycie za routerem uniemożliwia wymianę plików, na przykład podczas aktualizacji systemu Windows, poprzez połączenia P2P (peer to peer) z sieci publicznej do prywatnej.

Niestety dla osób grających w gry NAT nie przyniesie korzyści ponieważ, aby utworzyć sieć LAN z komputerem z poza swojej sieci trzeba będzie użyć dodatkowego oprogramowania np. Game Ranger czy też Tunngle.

1.4 Zapory sieciowe

1.4.1 Rodzaje zapór

Zapora sieciowa jest to jeden z wielu sposobów na zabezpieczenie sieci przed atakami z zewnątrz. Może ona być realizowana na komputerze wyposażonym w odpowiednie oprogramowanie lub na dedykowanym urządzeniu przeznaczonym wyłącznie do tego zadania. W drugim wypadku, podobnie jak w przypadku sprzętowego routera, mamy do czynienia z komputerem z zainstalowanym Linuxem lub innym systemem tylko o ograniczonych możliwościach ze względu na wymiary i moc obliczeniową.

Zadaniem zapory sieciowej jest generalnie analiza ruchu w sieci oraz wyłapywnie i blokowanie podejrzanych transmisji. Zwykle odbywa się to na łączach WAN, ale tam gdzie bezpieczeństwo danych ma najwyższy priorytet może być również na w miarę zaufanych łączach LAN.

Z uwagi na sposób filtrowania danych oraz poziom w modelu OSI na jakim działają zapory sieciowe można sklasyfikować w następujący sposób:

Filtr pakietów

Zadaniem zapory filtrującej pakiety danych jest monitoring pakietów wchodzących do sieci i wychodzących z sieci. Każdy pakiet przechodzi przez zestaw reguł. Na podstawie informacji takich jak źródłowy adres i port, docelowy adres i port, wartości flag TCP zawartych w nagłówkach pakietu reguła określa czy dany pakiet pasuje do niej czy nie. Jeśli tak to wykonywana jest jedna z dwóch akcji: przepuść lub odrzuć pakiet. Dodatkowo każda akcja może być zapisana w dzienniku systemowym.

Filtr pakietów jest chyba najszerzej stosowanym rodzajem zapory gwarantującym dość wysoki poziom bezpieczeństwa, ale zależy to w dużej mierze od doświadczenia administratora. Generalnie filtr pakietów jest dość trudny w konfiguracji bo wymaga bardzo dobrej znajomości sieci TCP/IP. Sama składnia reguł jest dość trudna, ale jeśli się wie jak działają sieci TCP/IP to można ją szybko opanować.

Przykładowe implementacje filtra pakietów to: IPFilter (Solaris, FreeBSD, NetBSD), IPTables i NetFilter (Linux), Packet Filter (OpenBSD).

NAT

Translacja adresów mimo, że została wymyślona w innym celu, stanowi swego rodzaju zaporę. Wykorzystuje się tutaj fakt, że host znajdujący się w sieci prywatnej, za routerem z NAT, gdy komunikuje się z publiczną siecią Internet to jego rzeczywisty adres jest zamieniany na adres routera. Ukrywa się w ten sposób tożsamość i uniemożliwia monitorowanie hosta z sieci publicznej.

Proxy

Serwery pośredniczące w komunikacji klient-serwer przy użyciu protokołu HTTP, trochę podobnie jak NAT, nie zostały wymyślone jako zapory, lecz miały za zadanie skrócić czas oczekiwania na dane z odległych serwerów, gdy Internet raczkował i nie było tak wiele sieci szerokopasmowych. Serwery proxy znajdowały się w strategicznych węzłach komunikacyjnych i buforowały ogromne ilość danych "przyspieszając" działanie WWW.

Obecnie, gdy większość z nas ma szerokopasmowe łącze w domu, nie ma potrzeby buforowania danych ze stron WWW. Wykorzystuje się za to możliwości serwerów proxy do analizy pobieranych danych i ewentualnego ich blokowania. W ten sposób realizowana jest kontrola rodzicielska i innego rodzaju kontrola zawartości (ang. content control). Tego rodzaju filtra nie uzyskamy filtrując pakiety na niskim poziomie modelu OSI, bo nie możemy wniknąć w strukturę przesyłanych danych. Zapora proxy działa natomiast w najwyższej warstwie modelu OSI, w warstwie aplikacji.

Filtr adresów MAC

Adres MAC to unikalny w skali globalnej adres karty sieciowej. Jego zmiana nie jest możliwa, można go tylko zamaskować wprowadzając inny adres, ale nie jest to łatwe. Dzięki filtracji adresów MAC możemy uniknąć problemów związanych z dostępem do sieci osób nie wpisanych na tak zwaną "zaufaną listę". Takie rozwiązanie zapewnia bezpieczeństwo dla osób będących w sieci przed wykradzeniem poufnych danych. Filtracja adresów MAC może odbywać się poprzez połączenie przewodowe jak i bezprzewodowe. Aby rozpocząć konfigurację takiego zabezpieczenia najważniejszym krokiem jest wyłączenie DHCP, które przydziela adresy IP dla każdego urządzenia podłączającego się do sieci bez sprawdzania jego adresu MAC (por. [11]).

Mówiąc o zaporach sieciowych warto też wspomnieć o systemach IDS (ang. Intrusion Detection Systems). Taki system w odróżnieniu od wymienionych

zapór dokonuje znacznie szerszej analizy ruchu w sieci. Używane są tutaj i dopasowywane wzorce niepożądanych zachowań w sieci oraz metody statystyczne do wykrywania zagrożeń.

1.4.2 IPFilter

Spośród wymienionych wcześniej różnych zapór sieciowych najbardziej interesować będzie nas IPFilter bo mamy go uruchomić w ramach tej pracy na maszynie z Solaris 11.

Filtr pakietów, aby mógł wykonać swoją robotę musi mieć pełny dostęp do wszystkich danych przepływających przez wszystkie interfejsy sieciowe maszyny. Oprogramowanie działające na tak niskim poziomie w systemie operacyjnym musi być modułem jądra systemu. I tak też jest w przypadku IPFiltra, o czym możemy się przekonać wołając `modinfo`:

```
indiana$ modinfo | grep ipf
253 ffffffff8c38000  38678 105   1  ipf (IP Filter: v4.1.9)
```

W Solaris IPFilter działa jako usługa o nazwie:

```
svc:/network/ipfilter:default
```

Na świeżo zainstalowanym systemie usługa ta jest wyłączona:

```
indiana$ svcs ipfilter
STATE          STIME     FMRI
disabled       18:51:30  svc:/network/ipfilter:default
```

Konfiguracja polega na edycji plików:

```
/etc/ipf/ipf.conf
/etc/ipf/ipnat.conf
```

Ten drugi plik odpowiada wprawdzie za NAT nie za filtr pakietów IPF, ale stanowią one integralną całość.

Składnia reguł IPF początkowo wydaje się bardzo skomplikowana, ale to dlatego, że jedna reguła może składać się z mnóstwa elementów i na początku ciężko zapamiętać kolejność oraz terminologię. Przyjrzyjmy się kilku przykładom.

```
block in on e1000g0 all
```

Pierwsze słowo kluczowe, tutaj `block`, oznacza akcję do wykonania. Słówko `in` odnosi się do pakietów wchodzących (ang. inbound) na interfejs podany dalej jako `e1000g0`. Tutaj warto powiedzieć, że w Solaris interfejsy sieciowe mają nazwy związane z odpowiadającymi im sterownikami, czyli modułami jądra. Tutaj `e1000g` to popularne, gigabitowe karty sieciowe Intela. Liczba `0` na końcu oznacza kolejny numer interfejsu. Ostatnie słowo kluczowe `all`

oznacza wszystkie pakiety. Reguła ta wydaje się być bardzo restrykcyjna bo blokuje cały ruch wejściowy. Brak słowa kluczowego `quick` powoduje jednak, że przetwarzanie reguł trwa dalej i jeśli pakiet nie zostanie dopasowany do kolejnej reguły, która go przepuści, to zostanie zablokowany. Tak więc, podsumowując, tę regułę możnaby czytać następująco: *odrzuć wszystkie pakiety wchodzące na interfejs `e1000g0`, chyba że zdecydujemy inaczej.* Jeśli chcemy być restrykcyjni i przepuszczać tylko wybrane pakiety to dobry pomysł, aby taką regułę umieścić jako pierwszą. W IPTables odpowiada to przyjęciu jako domyślnej polityki odrzucania (ang. policy drop).

```
block in quick on e1000g0 from 59.11.18.208
```

Słowo kluczowe `quick` oznacza wykonaj natychmiast i zakończ procedowanie dla tego pakietu. Natomiast `from` określa adres źródłowy. Tak więc tę regułę tłumaczymy jako: *odrzuć natychmiast pakiety wchodzące na interfejs `e1000g0` z adresu `59.11.18.208`.*

```
block in quick on e1000g0 from 10.0.0.0/8 to any
```

Jako adres źródłowy możemy wskazać całą podsieć, tutaj `10.0.0.0/8` to prywatna podsieć klasy A. Słówko `to` oznacza adres docelowy i jak się łatwo domyśleć `any` to dowolny adres. Zatem *odrzuć natychmiast pakiety wchodzące na interfejs `e1000g0` z sieci `10.0.0.0/8` skierowane gdziekolwiek.*

```
pass in quick on e1000g0 proto tcp from any
        to 212.33.73.197/32 port = 25 flags S keep state
```

Akcja `pass` oznacza przepuszczenie pakietu. Słowo kluczowe `proto` oznacza, że do tej reguły dopasowane będą wyłącznie pakiety w warstwie transportowej TCP. Ponieważ `port` wsytępuje po `to`, więc określa port docelowy (25 to port SMTP). Zaklęcie `flags S` oznacza pakiety z ustawioną flagą `SYN`, natomiast `keep state` mówi, aby IPF zapamiętał stan połączenia i przepuszczał kolejne pakiety w ramach tego połączenia.

```
pass out quick on e1000g0 proto tcp all keep state
```

Słowo `out` oznacza ruch wychodzący (ang. outbound). Tak więc tę regułę czytamy następująco: *przepuść natychmiast pakiety wychodzące z interfejsu `e1000g0` w protokole TCP i zapamiętaj stan połączeń.*

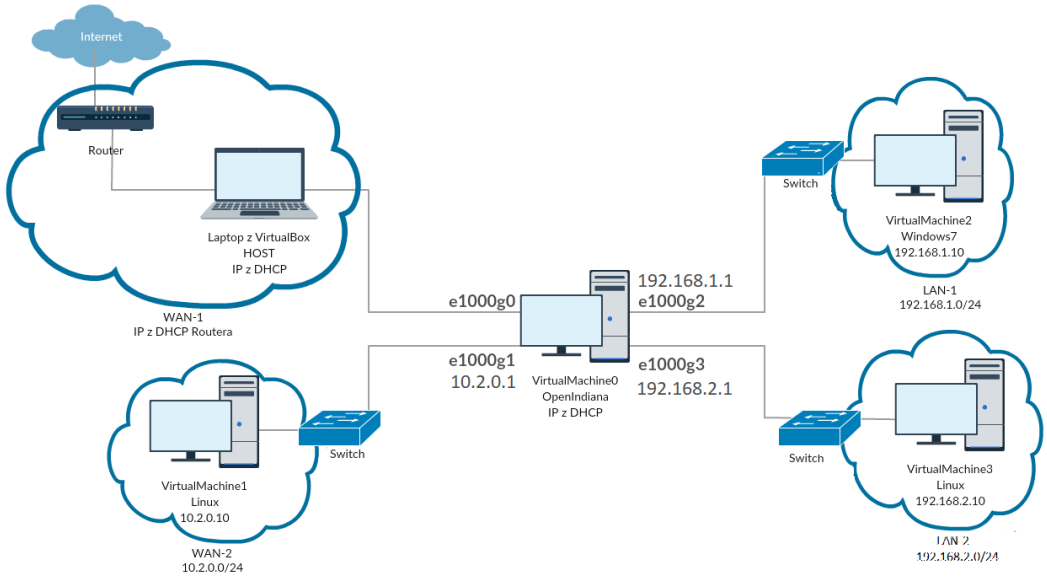
Tych kilka przykladów nie wyczerpuje możliwości IPF, ale pokazuje z czym mamy do czynienia.

Rozdział 2

Przykładowe wdrożenie

2.1 Instalacja i konfiguracja maszyn wirtualnych

Tak jak to zostało opisane we wstępie pracy konfigurujemy router na serwerze Solaris 11, który znajduje się w dwóch sieciach rozległych WAN oraz dwóch sieciach lokalnych LAN. Na tym routerze ma być uruchomiony NAT, tak aby z obu sieci LAN można było korzystać z sieci Internet, filtr pakietów do ochrony serwera i sieci lokalnych oraz co najważniejsze DNS, który udziela odpowiedzi zależnie od tego z jakiej sieci pochodzi zapytanie. Dodatkowo, dla wygody użytkowników, ale i administratora, uruchamiamy na serwerze DHCP.

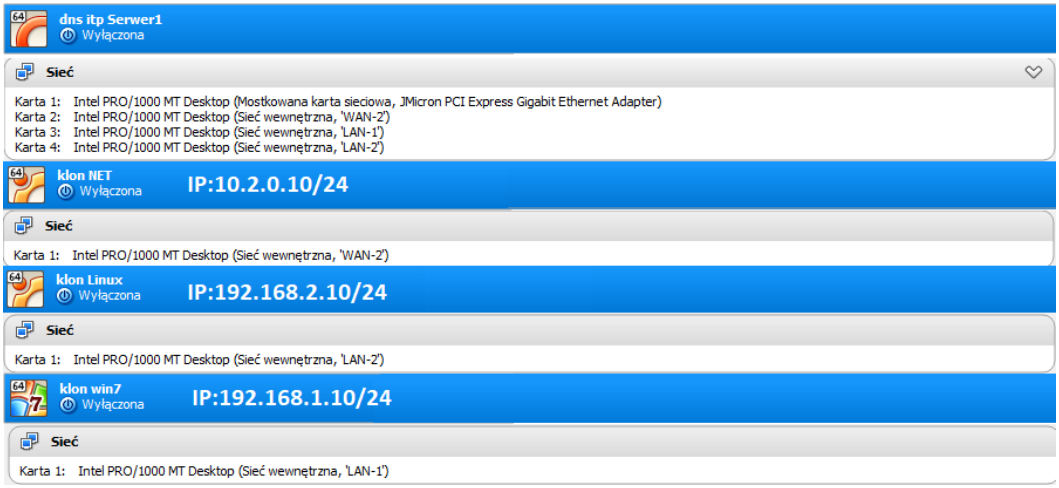


Rysunek 2.1: Poglądowy diagram sieci.

Zadanie realizuję na laptopie z zainstalowanym systemem Windows 10 i oprogramowaniem VirtualBox. Ten system będzie hostem dla pozostałych maszyn wirtualnych. Zakładam, że mam tutaj dostęp do sieci Internet i dostaję adres dynamicznie z DHCP.

Jako podstawowy serwer instaluję najnowszy system OpenIndiana Hipster 2017.04. Sieć WAN-1 będzie reprezentowana przez hosta z Windows 10, WAN-2 przez maszynę wirytualną z Linuxem Ubuntu, sieć LAN-1 przez maszynę wirtualną z Windows 7 oraz LAN-2 przez maszynę wirtualną z Linux Ubuntu. Na rysunku 2.1 w poglądowy sposób przedstwiamy jak ma wyglądać nasza sieć, jakie są połączenia między poszczególnymi maszynami i jakie jest miejsce serwera.

Aby ukazać konfigurację maszyn wirtualnych umieszczam zrzut ekranu 2.2, na którym widać jak każda z maszyn jest skonfigurowana oraz jak są nazwane jej karty sieciowe w VirtualBox.



Rysunek 2.2: Konfiguracja sieci w VirtualBox.

Po odpowiednim ustawieniu opcji w VirtualBox i instalacji wirtualnych maszyn należało skonfigurować na nich ustawienia sieciowe. Maszyny z sieci lokalnych pobierają adresy IP dynamicznie z serwera DHCP na OpenIndiana, natomiast maszyna w WAN-2 ma ustawiony adres statyczny. Dokładniej opiszę konfigurację na serwerze.

2.2 Konfiguracja ustawień sieciowych serwera

Konfigurację naszego serwera należy rozpocząć od ustawień sieciowych. Zaczynamy od:

```
dladm show-phys
```

Dzięki temu poleceniu zobaczymy listę kart sieciowych jakie są aktualnie zainstalowane w komputerze, czy są aktywne oraz jakie oznaczenie posiadają. W mojej konfiguracji na OpenIndiana z czterema interfejsami sieciowymi wygląda to tak:

```
indiana# dladm show-phys
LINK         MEDIA                STATE      SPEED  DUPLEX    DEVICE
e1000g0      Ethernet             up         1000   full      e1000g0
e1000g1      Ethernet             up         1000   full      e1000g1
e1000g2      Ethernet             up         1000   full      e1000g2
e1000g3      Ethernet             up         1000   full      e1000g3
```

Na serwerze konfigurujemy adresy statyczne. W Solaris 11 pojawiło się do tego specjalne polecenie:

```
ipadm
```

W celu przypisania adresu do danego interfejsu wystarczy użyć polecenia:

```
ipadm create-addr -T static -a X.X.X.X/M YYYYYY/v4
```

W miejscu gdzie są `X` należy wpisać adres IP jaki ma zostać na stałe przypisany do danej karty sieciowej, w miejscu `M` należy podać maskę sieci w postaci prefixu, natomiast w miejscu `YYYYYY` należy podać nazwę interfejsu sieciowego jaki będziemy konfigurować. W moim przypadku polecenie odnośnie konfiguracji interfejsu dla sieci LAN-1 wygląda następująco:

```
ipadm create-addr -T static -a 192.168.1.1/24 e1000g2
```

Dzięki temu na interfejsie `e1000g2` wpiętym do sieci LAN-1 uzyskałem adres IP `192.168.1.1` z maską sieciową o prefixie `24`, czyli `255.255.255.0`. Analogiczne zaklęcie należy powtórzyć dla każdego interfejsu sieciowego jaki zostanie użyty w systemie.

Po konfiguracji wypadałoby sprawdzić, czy każdy z interfejsów ma przydzielony poprawny adres IP oraz czy jest aktywny. W tym celu należy wpisać polecenie:

```
ipadm show-addr
```

Jeżeli wszystko jest dobrze skonfigurowane oraz nie ma żadnych problemów polecenie powinno zwrócić listę interfejsów wraz z przypisanymi do nich adresami sieci:

```
indiana# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
e1000g0/v4        static   ok           10.18.0.124/23
e1000g1/v4        static   ok           10.2.0.1/24
e1000g2/v4        static   ok           192.168.1.1/24
e1000g3/v4        static   ok           192.168.2.1/24
lo0/v6            static   ok           ::1/128
```

Natomiast jeżeli nie powiedzie się konfiguracja jakiegokolwiek interfejsu możemy usunąć adres poleceniem:

```
ipadm delete-addr YYYYYY/v4
```

Tak samo jak w przypadku dodawania adresu, tak i tu `YYYYYY` oznacza nazwę interfejsu, którego konfiguracja zostanie usunięta.

Stare dobre polecenie `ifconfig` dalej jest dostępne w Solaris 11:

```
indiana# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL>
mtu 8232 index 1 inet 127.0.0.1 netmask ff000000
e1000g0: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4>
mtu 1500 index 3 inet 10.18.0.124 netmask fffffe00 broadcast 10.18.1.255
 ether 8:0:27:2c:b7:81
e1000g1: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4>
mtu 1500 index 2 inet 10.2.0.1 netmask ffffff00 broadcast 10.2.0.255
 ether 8:0:27:bb:c5:32
e1000g2: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4>
mtu 1500 index 4 inet 192.168.1.1 netmask ffffff00 broadcast 192.168.1.255
 ether 8:0:27:ba:dc:ab
e1000g3: flags=1100843<UP,BROADCAST,RUNNING,MULTICAST,ROUTER,IPv4>
mtu 1500 index 5 inet 192.168.2.1 netmask ffffff00 broadcast 192.168.2.255
 ether 8:0:27:92:9c:32
```

W Solaris 11 polcenie `ifconfig -a` pokazuje zawsze wszystkie interfesy z przypisanymi lub nie adresami IPV4 oraz IPv6. Ponieważ nie używamy IPv6 tutaj ta część odpowiedzi została usunięta.

Ostatnia rzecz do zrobienia, która jest bardzo ważna, to uzyskanie dostępu do sieci Internet na serwerze. W tym celu musimy utworzyć domyślną bramę dla pakietówy wychodzących. Używamy polecenia:

```
route -p add default 10.18.0.1
```

Opcja `-p` zapewnia, że domyślna trasa pakietów przetrwa restart maszyny. Brama w naszym wypadku to `10.18.0.1`, czyli router na kampusie. Do sprawdzenia, czy wszystkie trasy są poprawne wywołujemy:

```
indiana# netstat -rn
```

```
Routing Table: IPv4
  Destination        Gateway        Flags  Ref     Use     Interface
---------------- ---------------- ----- ----- ---------- ---------
default          10.18.0.1        UG        3        103
10.2.0.0         10.2.0.1         U         2          0 e1000g1
10.18.0.0        10.18.0.124      U         3          0 e1000g0
127.0.0.1        127.0.0.1        UH        2         26 lo0
192.168.1.0      192.168.1.1      U         2          0 e1000g2
192.168.2.0      192.168.2.1      U         2          0 e1000g3
```

2.3 Konfiguracja DHCP

Aby rozpocząć konfigurację DHCP dla systemu Solaris 11 należy zainstalować pakiet:

```
pkg install isc-dhcp
```

Wyżej wymieniony dodatek jest niezbędy do rozpoczęcia konfiguracji naszego własnego DHCP. Gdy już pobierzemy oraz zainstalujemy potrzebne rzeczy warto usiąść na spokojnie i przemyśleć jak ma wyglądać nasza sieć, jakie mamy interfejsy oraz przydzielić im odpowiednie adresy w celu uniknięcia problemów w trakcie jak i po konfiguracji.

Serwer DHCP zainstalował się jako usługa:

```
svc:/network/dhcp/server:ipv4
```

Jest również dostępna wersja `ipv6`, ale ta nas nie insteresuje i pozostaje wyłączona. Zanim ją włączymy należy ją skonfigurować. Konfiguracja usług w Solaris odbywa się przy pomocy polecenia `svccfg`, więc uruchamiamy:

```
svccfg -s dhcp/server:ipv4
```

W ten sposób włączamy interaktywne narzędzie, którym między innymi możemy zmieniać właściwości usług. Poleceniem `listprop` możemy obejrzeć listę własności i ich aktualne wartości:

```
svc:/network/dhcp/server:ipv4> listprop
config                              application
config/config_file                  astring  /etc/inet/dhcpd4.conf
config/debug                        boolean  false
config/lease_file                   astring
/var/db/isc-dhcp/dhcpd4.leases
config/value_authorization          astring  solaris.smf.value.dhcp
config/listen_ifnames               astring
general                             framework
general/enabled                     boolean  true
restarter                           framework NONPERSISTENT
restarter/logfile                   astring
```

```
    /var/svc/log/network-dhcp-server:ipv4.log
restarter/contract                  count    102
restarter/start_pid                 count    697
restarter/start_method_timestamp    time     1504436158.641480000
restarter/start_method_waitstatus   integer  0
restarter/auxiliary_state           astring  dependencies_satisfied
restarter/next_state                astring  none
restarter/state                     astring  online
restarter/state_timestamp           time     1504436158.645375000
```

Nas interesuje własność `config/listen_ifnames` zawierająca listę interfejsów na których serwer DHCP będzie nasłuchiwał. Domyślnie ta lista jest pusta więc DHCP nie działa mimo włączenia usługi. Ustawiamy więc wartość:

```
setprop config/listen_ifnames=("e1000g2""e1000g3")
```

tak aby DHCP nasłuchiwał na inetrfejsach w naszych sieciach lokalnych LAN-1 i LAN-2 (por. rys. 2.1). Wychodzimy przez `CTRL+D` i odświeżamy usługę:

```
svcadm refresh dhcp/server:ipv4
```

Jak już przeszliśmy przez proces konfiguracji usługi DHCP możemy rozpocząć właściwą konfigurację serwera DHCP, w której będziemy mogli określić z jakiej puli adresów w danej sieci komputery będą otrzymywały adresy IP. W tym celu należy edytować plik:

```
/etc/inet/dhcpd4.conf
```

W przypadku moich dwóch sieci LAN-1 oraz LAN-2 konfiguracja wygląda następująco:

```
option domain-name "fabian.net.pl";
ddns-update-style none;
default-lease-time 72000;

subnet 192.168.1.0 netmask 255.255.255.0 {
        range 192.168.1.100 192.168.1.199;
        option broadcast-address 192.168.1.255;
        option subnet-mask 255.255.255.0;
        option routers 192.168.1.1;
        option domain-name-servers 192.168.1.1;
}

subnet 192.168.2.0 netmask 255.255.255.0 {
        range 192.168.2.100 192.168.2.199;
        option broadcast-address 192.168.2.255;
        option subnet-mask 255.255.255.0;
        option routers 192.168.2.1;
        option domain-name-servers 192.168.2.1;
}
```

Z czasem może wystąpić potrzeba, aby dane urządzenie pomimo dynamicznej puli adresów jakie będą przydzielane w podsieci miało swój własny niezmienny adres IP. W tym celu również w pliku konfiguracyjnym należy umieścić wpis:

```
host win7 {
        hardware ethernet 08:00:27:0a:bc:39;
        fixed-address 192.168.1.10;
}
```

Podajemy nazwę urządzenia/maszyny, tutaj to `win7`, adres MAC karty sieciowej oraz adres IP jaki ma zostać mu przydzielony. Dzięki takiemu zabiegowi pomimo dynamicznej puli adresów `192.168.1.100 - 192.168.1.199` maszynie z systemem Windows 7 został przydzielony adres IP `192.168.1.10`.

Jeżeli wszystkie kroki zostały właściwie według nas wykonane należy włączyć usługę DHCP, która jest domyślnie wyłączona. Robimy to poleceniem:

```
svcadm enable dhcp/server:ipv4
```

Po użyciu zaklęć zawierających w sobie `svc...`, wypada sprawdzić czy nie wyskoczyły błędy. Robimy to komendą `svcs -x`. Natomiast jeżeli pojawią się błędy warto sprawdzić je za pomocą komendy `dmesg | tail`.

2.4 Konfiguracja NAT

Zanim zaczniemy konfigurować translację adresów w IPFilter musimy włączyć przekazywanie pakietów pomiędzy interfejsami sieciowymi, czyli *IP forwarding*. W Solaris od wersji 10 służy do tego polecenie `routeadm`. Uruchomione bez opcji wyświetli aktualny stan. Domyślnie, na czysto zainstalowanym systemie, IP forwarding jest wyłączony, natomiast włączony jest routing. Routing wyłączamy, a włączamy forwarding:

```
routeadm -u -d ipv4-routing
routeadm -u -e ipv4-forwarding
```

Parametr `-u` mówi, aby wartość została zapisana na stałe i przetrwała reboot systemu, `-d` wyłącza (ang. disable), natomiast `-e` (ang. enable) włącza opcję. Po tej operacji powinniśmy uzyskać taki oto efekt:

```
indiana# routeadm
           Configuration   Current             Current
                  Option   Configuration       System State
---------------------------------------------------------------
             IPv4 routing   disabled            disabled
             IPv6 routing   disabled            disabled
           IPv4 forwarding  enabled             enabled
           IPv6 forwarding  disabled            disabled
```

```
        Routing services   "route:default ripng:default"

Routing daemons:

                   STATE   FMRI
                disabled   svc:/network/routing/ripng:default
                disabled   svc:/network/routing/legacy-routing:ipv4
                disabled   svc:/network/routing/legacy-routing:ipv6
                  online   svc:/network/routing/ndp:default
                disabled   svc:/network/routing/route:default
                disabled   svc:/network/routing/rdisc:default
```

W celu skonfigurowania NAT należy zmodyfikować plik odpowiedzialny za to:

`/etc/ipf/ipnat.conf`

Nasz plik wygląda tak:

```
map e1000g0 192.168.1.0/24 -> 0/32 proxy port ftp ftp/tcp
map e1000g0 192.168.1.0/24 -> 0/32 portmap tcp/udp 10000:30000
map e1000g0 192.168.1.0/24 -> 0/32

map e1000g0 192.168.2.0/24 -> 0/32 proxy port ftp ftp/tcp
map e1000g0 192.168.2.0/24 -> 0/32 portmap tcp/udp 10000:30000
map e1000g0 192.168.2.0/24 -> 0/32
```

Niemal dosłownie powtórzony jest blok 3 wierszy: pierwszy dla sieci LAN-1, drugi dla LAN-2. Obie sieci wychodzą w świat poprzez interfejs `e1000g0` na serwerze, czyli przez WAN-1. Pierwszy wiersz w tym bloku dotyczy FTP, bo FTP korzysta z dwóch, powiązancyh portów TCP: 20 i 21. Drugi wiersz mówi, że NAT ma tłumaczyć adresy IP oraz porty TCP oraz korzystać z portów TCP z zakresu 10000:30000. Trzeci realizuje właściwą translację adresów i ten jest w sumie wystarczający jeśli nie interesuje nas FTP i chcemy aby tylko adres IP był podmieniany przez NAT.

Ze względu, iż docelowo chciałbym posiadać skonfigurowane alternatywne łącze internetowe stworzyłem również drugi, alternatywny plik konfiguracyjny NAT. Jest on wręcz łudząco podobny do powyższego, gdzie NAT odbywa się na `e1000g0`, jednakże chciałbym, aby alternatywne łącze było na karcie sieciowej `e1000g1`, czyli poprzez WAN-2, więc odpowiedni plik konfiguracyjny wygląda następująco:

```
map e1000g1 192.168.1.0/24 -> 0/32 proxy port ftp ftp/tcp
map e1000g1 192.168.1.0/24 -> 0/32 portmap tcp/udp 10000:30000
map e1000g1 192.168.1.0/24 -> 0/32

map e1000g1 192.168.2.0/24 -> 0/32 proxy port ftp ftp/tcp
map e1000g1 192.168.2.0/24 -> 0/32 portmap tcp/udp 10000:30000
map e1000g1 192.168.2.0/24 -> 0/32
```

Pierwszy plik nazwałem `ipnat-e1000g0.conf`, a drugi `ipnat-e1000g1.conf`. Aby ułatwić sobie życie postanowiłem zastąpić oryginalny plik `ipnat.conf`, linkiem symbolicznym do odpowiedniej wersji konfiguracji NAT. Teraz, aby przełączyć dostawcę internetu na przykład z WAN-1 na WAN-2, należy przejść z linii poleceń do katalogu `/etc/ipf/` oraz użyć poleceń:

```
rm ipnat.conf
ln -s ipnat-e1000g1.conf ipnat.conf
```

Aby włączyć NAT należy włączyć usługę IPFilter:

```
svcadm enable ipfilter
```

Po każdej zmianie w pliku `ipnat.conf` ustawienia aktualizujemy wpisując:

```
ipnat -F -C -f /etc/ipf/ipnat.conf
```

gdzie poszczególne opcje oznaczają:

`-F` czyszczenie tablicy NAT (czyszczenie połączeń)

`-C` czyszczenie zestawu reguł

`-f` ścieżka do pliku konfiguracyjnego

Po całym zabiegu warto przejść do maszyn w sieci LAN-1 i LAN-2 oraz sprawdzić czy mamy łączność z Internetem. Na serwerze warto sprawdzić listę połączeń używając zaklęcia:

```
ipnat -l
```

Po wpisaniu tej komendy otrzymamy taki wynik:

```
indiana# ipnat -l
List of active MAP/Redirect filters:
map e1000g0 192.168.1.0/24 -> 0.0.0.0/32 proxy port ftp ftp/tcp
map e1000g0 192.168.1.0/24 -> 0.0.0.0/32 portmap tcp/udp 10000:30000
map e1000g0 192.168.1.0/24 -> 0.0.0.0/32
map e1000g0 192.168.2.0/24 -> 0.0.0.0/32 proxy port ftp ftp/tcp
map e1000g0 192.168.2.0/24 -> 0.0.0.0/32 portmap tcp/udp 10000:30000
map e1000g0 192.168.2.0/24 -> 0.0.0.0/32

List of active sessions:
MAP 192.168.1.10    123   <- -> 10.18.0.124   14173 [13.79.239.69 123]
MAP 192.168.1.10    49179 <- -> 10.18.0.124   27898 [204.79.197.200 80]
```

oraz co jakiś czas warto monitorować statystki:

```
indiana# ipnat -s
mapped  in      3915    out     2099
added   98      expired 0
no memory       0       bad nat 0
inuse   2
orphans 0
rules   6
wilds   0
```

Dzięki zastosowaniu dwóch różnych konfiguracji NAT, w razie awarii wystarczy tylko przełączyć zawartość pliku. Gdy już problemy z dostawcą internetu w sieci WAN-1 ustąpią i chcemy wrócić do oryginalnej sieci, należy postępować tak samo jak w przypadku zmiany z WAN-1 do WAN-2, z tą różnicą że zamiast w poleceniu tworzącym link symboliczny użyć `ipnat-e1000g0.conf` zamiast `ipnat-e1000g1.conf`.

2.5 Konfiguracja DNS

Tak samo jak w przypadku konfiguracji DHCP, tak i tu potrzebujemy zainstalować dodatki do systemu. Instalujemy oprogramowanie serwera DNS zaimplementowane jako ISC BIND:

```
pkg install dns/bind
```

To wystarczy, aby bezproblemowo oraz bezkonfliktowo przeprowadzić konfigurację własnego DNS. Usługa pod którą zainstalował się serwer DNS nazywa się:

```
svc:/network/dns/server:default
```

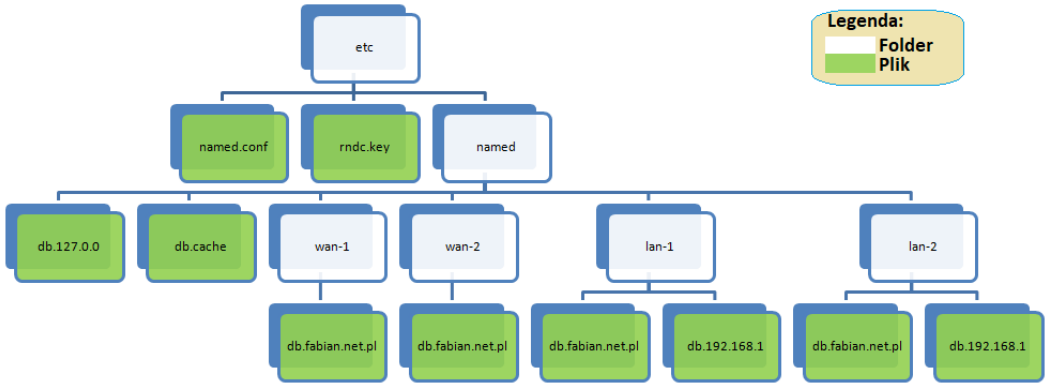
Uruchamianie serwera DNS zaczynamy od wygenerowania kluczy dla narzędzia do zarządzania serwerem nazywającego się `rndc`. Gotowy fragment pliku konfiguracyjnego wygeneruje nam `rndc-confgen`. W pliku `/etc/rndc.conf` umieszczamy:

```
key "rndc-key" {
    algorithm hmac-md5;
    secret "CqbfTbRv9+qyhlnCWji7HA==";
};
```

i tę samą deklarację umieszczamy też w głównym pliku konfiguracyjnym serwera DNS `/etc/named.conf`.

Domena, którą konfigurujemy w omawianym przykładzie to `fabian.net.pl`. Na dzień 3 września 2017 r. jest ona niezarejestrowana.

Aby pokazać jak wygląda konfiguracja serwera DNS z wykorzystaniem widoków chciałbym najpierw zaprezentować odpowiednie drzewo katalogów na moim serwerze:



Rysunek 2.3: Struktura plików i katalogów konfiguracji serwera DNS.

Konfiguracja praktycznie każdego oprogramowania, w tym serwera DNS, jest wieloetapowa i czasem stosuje się metodę prób i błędów. Pokażę jednak efekt końcowy. Rozpoczynamy od najważniejszego pliku `/etc/named.conf`, który wygląda następująco:

```
options {
     directory "/etc/named";
     listen-on { 10.18.0.124; 10.2.0.1; 192.168.1.1; 192.168.2.1; };
     auth-nxdomain yes;
     forward first;
     forwarders { 8.8.8.8; 8.8.4.4; };
};

key "rndc-key" {
    algorithm hmac-md5;
    secret "CqbfTbRv9+qyhlnCWji7HA==";
};

controls {
     inet * allow { any; } keys { "rndc-key"; };
};

view "lan-1" {
    match-clients { 192.168.1.0/24; };
    recursion yes;
    zone "." {
        type hint;
        file "db.cache";
    };
    zone "0.0.127.in-addr.arpa" {
        type master;
        notify no;
```

```
        file "db.127.0.0";
    };
    zone "1.168.192.in-addr.arpa" in {
        type master;
        file "lan-1/db.192.168.1";
    };
    zone "fabian.net.pl" {
        type master;
        notify no;
        file "lan-1/db.fabian.net.pl";
    };
};

view "lan-2" {
    match-clients { 192.168.2.0/24; };
    recursion yes;
    zone "0.0.127.in-addr.arpa" {
        type master;
        notify no;
        file "db.127.0.0";
    };
    zone "2.168.192.in-addr.arpa" in {
        type master;
        file "lan-2/db.192.168.2";
    };
    zone "fabian.net.pl" {
        type master;
        notify no;
        file "lan-2/db.fabian.net.pl";
    };
};

view "wan" {
    match-clients { any; };
    recursion no;
    zone "fabian.net.pl" {
        type master;
        notify no;
        file "wan/db.fabian.net.pl";
    };
};
```

Jak widać znajdują się tutaj 3 widoki: `lan-1`, `lan-2` i `wan`. Nie ma rozbicia na WAN-1 i WAN-2 bo jedno z przyłączy traktowane jest jako zapasowe. Tak więc w danym momencie w sieci publicznej nasz serwer ma jedno IP w zależności, który WAN jest podstawowym. W razie awarii link symboliczny `wan` będzie przełączany na katalog `wan-1` lub `wan-2`.

Kolejność jest istotna bo jako pierwsze powinny być dopasowywane zapytania z sieci lokalnych, a dopiero potem z sieci WAN jako pozostałe. Zgodnie z ogólnymi zasadami zapytania rekurencyjne możliwe są tylko w sieciach lokalnych. Jako `forwarders`, czyli DNSy, które będą wykorzystane dla wszystkich zapytań oprócz naszej własnej domeny, wziąłem DNSy Google: `8.8.8.8` oraz `8.8.4.4`. Powinny one działać w każdej sieci. Jest to uproszczenie bo serwer jest na laptopie kursującym między domem i kampusem. Najlepiej jest użyć DNSów proponowanych przez dostawcę Internetu.

Poza strefą `fabian.net.pl` wyjaśnienia wymagają pozostałe strefy:

- `.` – wymagana w zapytaniach rekurencyjnych, zawiera serwery nazw domen TLD,
- `0.0.127.in-addr.arpa` – wymagana do rozwiązywania `localhost`,
- `1.168.192.in-addr.arpa` – odwrotny DNS dla LAN-1,
- `2.168.192.in-addr.arpa` – odwrotny DNS dla LAN-2.

Tak skonfigurowany DNS nakazuje nam stworzenie podkatalogów dla sieci LAN-1, LAN-2 oraz WAN. W tych podkatalogach dla każdej z sieci będą umieszczane odpowiednie wersje deklaracji stref. Wyglądają one następująco.

Strefa `lan-1/db.fabian.net.pl`

```
$TTL 2h
@ IN SOA indiana.fabian.net.pl. sysadm.indiana.fabian.net.pl. (
                                        1       ; serial
                                        1h      ; refresh
                                        5m      ; retry
                                        2w      ; expire
                                        2h)     ; minimum
            IN NS   ns.fabian.net.pl.
localhost   IN A    127.0.0.1
loghost     IN A    127.0.0.1
@           IN A    192.168.1.1
indiana     IN A    192.168.1.1
ns          IN A    192.168.1.1
win7        IN A    192.168.1.10
www         IN CNAME    indiana
ftp         IN CNAME    indiana
mail        IN CNAME    indiana
@           IN MX   10 indiana.fabian.net.pl.
```

Strefa `lan-2/db.fabian.net.pl`

```
$TTL 2h
@ IN SOA indiana.fabian.net.pl. sysadm.indiana.fabian.net.pl. (
                                        1       ; serial
                                        1h      ; refresh
                                        5m      ; retry
                                        2w      ; expire
                                        2h)     ; minimum
            IN NS   ns.fabian.net.pl.
localhost   IN A    127.0.0.1
loghost     IN A    127.0.0.1
@           IN A    192.168.2.1
indiana     IN A    192.168.2.1
ns          IN A    192.168.2.1
win7        IN A    192.168.2.10
www         IN CNAME    indiana
ftp         IN CNAME    indiana
mail        IN CNAME    indiana
@           IN MX   10 indiana.fabian.net.pl.
```

Strefa `wan-1/db.fabian.net.pl`

```
$TTL 2h
@ IN SOA indiana.fabian.net.pl. sysadm.indiana.fabian.net.pl. (
                                        1       ; serial
                                        1h      ; refresh
                                        5m      ; retry
                                        2w      ; expire
                                        2h)     ; minimum
            IN NS   ns.fabian.net.pl.
localhost   IN A    127.0.0.1
loghost     IN A    127.0.0.1
@           IN A    10.18.0.124
indiana     IN A    10.18.0.124
ns          IN A    10.18.0.124
failover    IN A    10.2.0.1
www         IN CNAME    indiana
ftp         IN CNAME    indiana
mail        IN CNAME    indiana
@           IN MX   10 indiana.fabian.net.pl.
@           IN MX   20 failover.fabian.net.pl.
```

Strefa `wan-2/db.fabian.net.pl`

```
$TTL 2h
@ IN SOA indiana.fabian.net.pl. sysadm.indiana.fabian.net.pl. (
                                        1       ; serial
                                        1h      ; refresh
```

```
                                        5m      ; retry
                                        2w      ; expire
                                        2h)     ; minimum
            IN NS   ns.fabian.net.pl.
localhost   IN A    127.0.0.1
loghost     IN A    127.0.0.1
@           IN A    10.2.0.1
indiana     IN A    10.2.0.1
ns          IN A    10.2.0.1
failover    IN A    10.18.0.124
www         IN CNAME    indiana
ftp         IN CNAME    indiana
mail        IN CNAME    indiana
@           IN MX   10 indiana.fabian.net.pl.
@           IN MX   20 failover.fabian.net.pl.
```

Różnice pomiędzy poszczególnymi strefami dotyczą tylko adresacji IP. W przypadku konkretnej sieci takie pliki należałoby dostosować do własnych potrzeb, które jak wiadomo każdy ma inne. Tutaj pokazujemy tylko mały przykład.

Gdy już zakończymy konfigurację naszego DNS, to warto z niego korzystać na serwerze. W tym celu edytujemy plik `/etc/resolv.conf`:

```
domain fabian.net.pl
search fabian.net.pl
nameserver 10.18.0.124
```

Tak więc, na serwerze będziemy mieli adresację dla domeny `fabian.net.pl` taką, jak dla sieci publicznej.

2.6 Konfiguracja IPF

Aby rozpocząć konfigurację IPFilter należy przejść do katalogu `/etc/ipf` oraz odnaleźć plik `ipf.conf`, który jest odpowiedzialny za konfigurację filtra pakietów IPF.

Na zaporze IPF przyjmujemy restrykcyjną politykę dla obu sieci WAN i odrzucamy wszystkie pakiety, które nie zostaną przepuszczone przez konkretne reguły. Z sieci LAN przepuszczamy wszystko z jednym wyjątkiem: blokujemy połączenia do poczty SMTP (port 25 TCP) z sieci LAN-2. Zatem na interfejsie `e1000g0` w sieci WAN-1 mamy następujące reguły:

1. Natychmiast odrzucamy pakiety z sieci prywatnych.
2. Blokujemy protokół IDENT.
3. Przepuszczamy traceroute'a odpowiadając ICMP: port unreachable na portach UDP powyżej 33400.

4. Przepuszczamy ruch wychodzący bez ograniczeń.
5. Przepuszczamy żądanie echa PING.
6. Otwieramy porty
   (a) SSH – TCP 22,
   (b) SMTP – TCP 25,
   (c) DNS – TCP 53 i UDP 53,
   (d) HTTP – TCP 80,
   (e) SMTP z TLS – TCP 587,
   (f) HTTPS – TCP 443,
   (g) IMAPS – TCP 993,
   (h) POP3S – TCP 995

Na interfejsie `e1000g1` mamy ten problem, że odpowiedzi na pakiety przychodzące z sieci WAN-1 będą kierowane do domyślnej bramy `10.18.0.1`, czyli do sieci WAN-1 i nikt z nami się nie dogada po IP `10.2.0.1`. Problem rozwiązuje tak zwane *policy routing*. Tak więc, dla interfejsu `e1000g1` w sieci WAN-2 mamy następujące reguły:

1. W ramach policy routing pakiety z adresem źródłowym `10.18.0.124` przepychamy do bramy sieci WAN-1 na interfejsie `e1000g0`.
2. Natychmiast odrzucamy pakiety z sieci prywatnych.
3. Blokujemy protokół IDENT.
4. Przepuszczamy traceroute'a odpowiadając ICMP: port unreachable na portach UDP powyżej 33400.
5. Przepuszczamy ruch wychodzący bez ograniczeń.
6. Przepuszczamy żądanie echa PING z opcją `reply-to` tak, aby odpowiedź wyszła portem `e1000g1`, a nie bramą domyślną `10.18.0.1`.
7. Otwieramy wyłącznie port SSH – TCP 22 z opcją `reply-to` jak wyżej.

Ostateczna wersja pliku `/etc/ipf/ipf.conf` wygląda tak:

```
# WAN-1
block in on e1000g0 all
#
# Antispoofing
block in quick on e1000g0 from 192.168.0.0/16 to any
block in quick on e1000g0 from 172.16.0.0/12 to any
block in quick on e1000g0 from 127.0.0.0/8 to any
```

```
block in quick on e1000g0 from 0.0.0.0/8 to any
block in quick on e1000g0 from 169.254.0.0/16 to any
block in quick on e1000g0 from 192.0.2.0/24 to any
block in quick on e1000g0 from 204.152.64.0/23 to any
block in quick on e1000g0 from 224.0.0.0/3 to any
block in quick on e1000g0 from 195.117.228.65/32 to any
#
# IDENT block by sending RST
block return-rst in quick on e1000g1 proto tcp from any to e1000g0/32
    port = 113
#
# Traceroute
block return-icmp (port-unr) in quick on e1000g0 proto udp from any
    to e1000g0/32 port > 33400
#
# Common outgoing packets
pass out quick on e1000g0 proto tcp all keep state keep frags
pass out quick on e1000g0 proto udp all keep state keep frags
pass out quick on e1000g0 proto icmp all keep state keep frags
#
# ICMP echo request
pass in quick on e1000g0 proto icmp from any to e1000g0/32 icmp-type 8
    keep state
#
# SSH
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 22
    flags S keep state
#
# SMTP
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 25
    flags S keep state
#
# DNS
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 53
pass in quick on e1000g0 proto udp from any to e1000g0/32 port = 53
#
# HTTP
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 80
    flags S keep state
#
# SMTP+TLS
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 587
    flags S keep state
#
# HTTP over SSL
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 443
    flags S keep state
#
# IMAP over SSL
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 993
    flags S keep state
#
```

```
# POP over SSL
pass in quick on e1000g0 proto tcp from any to e1000g0/32 port = 995
    flags S keep state
#
#
# WAN-2
# Policy routing
pass out quick on e1000g1 to e1000g0:10.18.0.1 from 10.18.0.124/32
    to any keep state
#
block in on e1000g1 all
#
# Antispoofing
block in quick on e1000g1 from 192.168.0.0/16 to any
block in quick on e1000g1 from 172.16.0.0/12 to any
block in quick on e1000g1 from 127.0.0.0/8 to any
block in quick on e1000g1 from 0.0.0.0/8 to any
block in quick on e1000g1 from 169.254.0.0/16 to any
block in quick on e1000g1 from 192.0.2.0/24 to any
block in quick on e1000g1 from 204.152.64.0/23 to any
block in quick on e1000g1 from 224.0.0.0/3 to any
block in quick on e1000g1 from 195.117.228.65/32 to any
#
# IDENT block by sending RST
block return-rst in quick on e1000g1 proto tcp from any to e1000g1/32
    port = 113
#
# Traceroute
block return-icmp (port-unr) in quick on e1000g1 proto udp from any
    to e1000g1/32 port > 33400
#
# Common outgoing packets
pass out quick on e1000g1 proto tcp all keep state
pass out quick on e1000g1 proto udp all keep state
pass out quick on e1000g1 proto icmp all keep state
#
# ICMP echo request
pass in quick on e1000g1 reply-to e1000g1:10.2.0.10 proto icmp from any
    to e1000g1/32 icmp-type 8 keep state
#
# SSH
pass in quick on e1000g1 reply-to e1000g1:10.2.0.10 proto tcp from any
    to e1000g1/32 port = 22 flags S keep state
#
#
# LAN-2
block in log quick on e1000g3 from any to any port = 25
```

Ponieważ IPFilter był włączony, gdy uruchamialiśmy NAT, o czym możemy się przekonać:

```
indiana#svcs -p ipfilter
STATE          STIME    FMRI
```

```
online          16:41:14 svc:/network/ipfilter:default
                16:41:14     2571 svc.ipfd
                16:41:14     2573 ipmon
```

wystarczy podczytać reguły z pliku:

```
ipf -F a -f /etc/ipf/ipf.conf
```

Zostaniemy poinformowani o ewentualnych błędach syntaktycznych. Jeśli wynik jest pusty to po Unixowemy wszystko OK. Dla sprawdzenia możemy wylistować zadeklarowane reguły poleceniami:

```
ipfstat -i
ipfstat -o
```

Analogicznie jak dla NAT, czy DNS, tworzymy dwa pliki: `ipf-e1000g0.conf`, `ipf-e1000g1.conf`. Pierwszy to zapisany przed chwilą `ipf.conf` tylko zmieniamy mu nazwę. Drugi plik jest jakby symetryczny względem pierwszego. Następnie tworzymy link symboliczny:

```
ln -s ipf-e1000g0.conf ipf.conf
```

do przełączenia w razie awarii.

2.7 WAN failover

W celu uniknięcia błędów i ułatwienia sobie zadania przełączania konfiguracji w razie awarii u dostawcy internetowego napisałem skrypt. Dzięki prostocie mechanizmu nie zachodzą potrzeby dodatkowej konfiguracji, wystarczy uruchomienie skryptu.

Do przełączenia się na sieć WAN-2 (zastępczą) sktypt wygląda następująco:

```
#!/bin/bash
# wlaczenie polaczenia na WAN-2
#
# Default route
route -p delete default 10.18.0.1
route -p add default 10.2.0.10
# NAT
rm /etc/ipf/ipnat.conf
ln -s /etc/ipf/ipnat-e1000g1.conf /etc/ipf/ipnat.conf
ipnat -F -C -f /etc/ipf/ipnat.conf
ipnat -l
# IPF
rm /etc/ipf/ipf.conf
ln -s /etc/ipf/ipf-e1000g1.conf /etc/ipf/ipf.conf
ipf -F a -f /etc/ipf/ipf.conf
```

```
# DNS
rm /etc/named/wan
ln -s /etc/named/wan-2 /etc/named/wan
rndc reload
echo 'Sieć dostępowa została zmieniona na WAN-2'
```

Został on zapisany w pliku pod nazwą `wan-2.sh`, natomiast aby wrócić do głównego dostawcy Internetu pracującego na karcie sieciowej o oznaczeniu e1000g0 wystarczy uruchomić drugi skrypt pod nazwą `wan-1.sh`, w którym znajdują się analogiczne polecenia:

```
#!/bin/bash
# wlaczenie polaczenia na WAN-1
#
# Default route
route -p delete default 10.2.0.10
route -p add default 10.18.0.1
# NAT
rm /etc/ipf/ipnat.conf
ln -s /etc/ipf/ipnat-e1000g0.conf /etc/ipf/ipnat.conf
ipnat -F -C -d /etc/ipf/ipnat.conf
ipnat -l
# IPF
rm /etc/ipf/ipf.conf
ln -s /etc/ipf/ipf-e1000g0.conf /etc/ipf/ipf.conf
ipf -F a -f /etc/ipf/ipf.conf
# DNS
rm /etc/named/wan
ln -s /etc/named/wan-1 /etc/named/wan
rndc reload
echo 'Sieć dostępowa została zmieniona na WAN-1'
```

Procedurę przełączania można zautomatyzować, ale to nie jest takie banalne zadanie. Nie jest to priorytetem w tej pracy.

Bibliografia

[1] P. Albitz, C. Liu, *DNS and BIND*, ed. 5th, O'Reilly Media, 2009.

[2] `https://www.isc.org/downloads/bind/`
Stan na dzień: 16.08.2017

[3] `https://pl.wikipedia.org/wiki/OpenSolaris`
Stan na dzień: 15.08.2017

[4] `https://wiki.illumos.org/`
Stan na dzień: 10.08.2017

[5] `https://openindiana.org/`
Stan na dzień: 10.08.2017

[6] `https://virtualbox.org/`
Stan na dzień: 5.08.2017

[7] `https://pl.wikipedia.org/wiki/Domena_najwyszego_poziomu`
Stan na dzień: 27.08.2017

[8] `https://roan24.pl/aktualnosc/podstawy-dns-strefa-dns/`
Stan na dzień: 27.08.2017

[9] `https://krowicki.pl/widoki-strefy-domeny/`
Stan na dzień: 29.08.2017

[10] `http://www.zytrax.com/books/dns/ch7/view.html`
Stan na dzień: 29.08.2017

[11] `https://www.poradykomputerowe.pl/bezpieczenstwo-komputera/filtrowanie-adresow-mac-w-sieci-wi-fi.html`
Stan na dzień: 31.08.2017