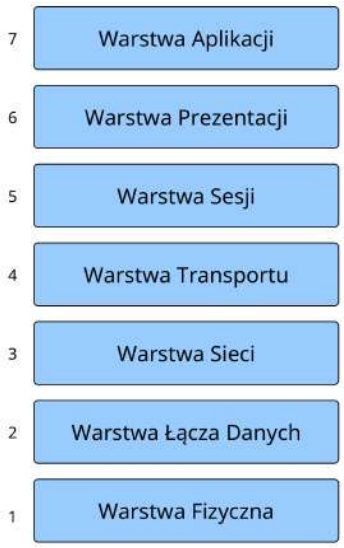
Jak działa Internet
Model OSI, warstwa fizyczna, adresy MAC i IP

Jest to pierwszy z serii artykułów, w których chciałbym wprowadzić czytelnika w podstawy działania „Internetu". Zaczniemy od omówienia modelu OSI, opisując pobieżnie warstwę fizyczną połączeń w sieciach, adresów MAC oraz IP (w obecnym artykule), dojdziemy kolejno do bardziej szczegółowego opisu protokołu TCP, systemu DNS, a skończymy na witrynach internetowych oraz API korzystających z HTTP. Przy okazji poznamy program Wireshark do analizy ruchu sieciowego. Zaczynajmy!

Od razu wyjaśnijmy jedną rzecz: w artykule nie będziemy się skupiali na Internecie jako takim. Internet jest zbiorem połączonych ze sobą sieci, które komunikują się ze sobą. Żeby zrozumieć tę komunikację, będziemy upraszczać sieci do kilku urządzeń lub kilku małych sieci. Nie sposób jednak poruszyć tematu komunikacji bez omówienia modelu OSI.

MODEL OSI

Model OSI (Open Systems Interconnection) to bardzo ogólny model komunikacji. Dzieli on technologiczny stos komunikacyjny na warstwy o pojedynczych odpowiedzialnościach. W rzeczywistości implementacja stosów komunikacyjnych nie jest w pełni zgodna z tym modelem (poszczególne warstwy mogą nie być rozdzielone od siebie w taki sam sposób). Mimo to model jest bardzo popularny i szeroko stosowany w żargonie sieciowym. Często np. będziemy mówili o tym, którą warstwę sieci wspierają urządzenia.



Rysunek 1. Model OSI

Jak widzimy, model OSI składa się z siedmiu warstw, zgodnie z intuicją wyższe warstwy odpowiadają wyższym poziomom abstrakcji.

W tym artykule zajmiemy się głównie warstwami 1-3. Warstwę Fizyczną omówimy bardzo pobieżnie, ponieważ jest ona najmniej ciekawa z „Programisty" punktu widzenia. Z warstwą łącza danych będziemy mieć do czynienia od czasu do czasu, a z warstwą sieciową często.

Warstwa Fizyczna

Tak jak sama nazwa wskazuje, Warstwa Fizyczna opisuje komunikację na najniższym – fizycznym poziomie. Mam tutaj na myśli opis tego, jak wyglądają symbole, jak np. zera i jedynki przy połączeniu przewodem elektrycznym, czy jak synchronizować długość trwania przesyłanego bitu. Podobnie, to właśnie ta warstwa opisuje częstotliwości używane przez połączenia Wi-Fi.

Przykładami implementacji warstw fizycznych są:

- Przewody UTP, FTP, STP – przewody ethernetowe kolejno bez ekranowania, z ekranowaniem całej skrętki i z ekranowaniem poszczególnych par. Różne ekranowania odpowiadają odporności na zakłócenia elektromagnetyczne.
- Opisy fal elektromagnetycznych i ich częstotliwości używanych chociażby w połączeniach bezprzewodowych.
- Urządzenia takie jak huby ethernetowe. (Obecnie raczej nie zobaczymy hubów w praktyce. Są to stare urządzenia. Warto jednak o nich pamiętać przy omawianiu różnych warstw sieci, chociażby po to, żeby porównać ich sposób działania do popularnych switchy, o czym później).

Warstwa Łącza Danych

Jest to warstwa protokołu, która pozwala na komunikację między urządzeniami *znajdującymi się w jednej sieci lokalnej*. Co to znaczy jedna sieć lokalna? Na nasze potrzeby chodzi o komunikację tych urządzeń między sobą, które znajdują się w jednym mieszkaniu i używają tego samego routera do dostępu do Internetu (mimo że nie omawialiśmy routerów, to zakładam, że intuicyjnie wszyscy wiemy, o które dokładnie urządzenie w mieszkaniu chodzi).

Jest to dość ciekawa warstwa, ponieważ często rozdziela się ją na 2 pomniejsze warstwy:

- LLC (ang. *Logical Link Control*), która jest warstwą wyższą oraz
- MAC (ang. *Media Access Control*), będąca warstwą niższą

Do Warstwy Łącza Danych należą przede wszystkim karty sieciowe (ang. NIC – Network Interface Card). To właśnie karty sieciowe

implementują logikę związaną z pakowaniem i rozpakowywaniem danych z tzw. ramek danych (warstwa LLC), adresowaniem danych (również warstwa LLC), a także bezpośrednio obsługują warstwę fizyczną (warstwa MAC).

Ramka danych to określenie, które jest przypisane bezpośrednio do warstwy drugiej modelu OSI. Dane kolejnych warstw będą nazywane kolejno:

- pakietami – warstwa 3,
- segmentami, datagramami – warstwa 4,
- żądaniami i odpowiedziami HTTP – warstwa 7 (w kolejnych częściach tej serii będziemy omawiać właśnie protokół HTTP, natomiast inne protokoły mogą mieć model komunikacji inny niż żądania i odpowiedzi).

Wrócimy jeszcze do tego w dalszej części artykułu.

Z warstwą MAC możemy kojarzyć odpowiadające jej adresy – MAC. Adres MAC to jedno z pól struktury danych znajdujących się w ramce, która odpowiada za identyfikowanie urządzenia w sieci. Jest to tzw. *adres fizyczny*, ponieważ nadawany jest urządzeniom już podczas ich produkcji (można je jednak zmieniać również poprzez oprogramowanie). Konkretniej, adres MAC jest przypisany do karty sieciowej (ang. NIC - Network Interface Card). Komputery, czy inne urządzenia, mogą mieć wiele kart sieciowych, w tym wirtualnych kart sieciowych (vNIC). Adresy te zajmują 48 bitów informacji (6 bajtów). Pierwsze 3 z nich identyfikują firmę, która wyprodukowała urządzenie, a kolejne wskazują na konkretne urządzenie, co przedstawiono na Rysunku 2.

W zapisie często stosuje się zapis szesnastkowy. Przykładowy adres MAC wygląda wtedy następująco: `28:97:C1:38:51:E2`.

W ramach sieci lokalnej urządzenia wysyłają do siebie wiadomości, używając właśnie adresów MAC, możemy również sprawdzić adresy MAC naszego urządzenia. W przypadku systemu operacyjnego Windows możemy wywołać komendę `ipconfig /all`, a w środowiskach typu UNIX odpowiednią komendą może być `ip a` (na macOS znajdziemy komendę `ifconfig`). Przykładowe wyjście komendy możemy zobaczyć poniżej:

```
1: lo: <LOOPBACK,UP,LOWER_UP> …
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: wlp2s0: <BROADCAST,MULTICAST,UP,LOWER_UP>...
    link/ether 74:e5:f9:86:f9:0c brd ff:ff:ff:ff:ff:ff
    inet 10.11.0.75/24 brd 10.11.0.255
       scope global dynamic noprefixroute wlp2s0
       valid_lft 82888sec preferred_lft 82888sec
    inet6 fe80::a494:5de9:603c:9f77/64
       scope link noprefixroute valid_lft forever
       preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP>…
    link/ether 02:42:36:87:00:08 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255
      scope global docker0 valid_lft forever
      preferred_lft forever
```

Powyższa komenda pokazuje nam, że nasza maszyna ma 3 interfejsy sieciowe:

- `lo` (tzw. loopback),
- `wlp2s0` (interfejs Wi-Fi),
- `docker0` (wirtualny interfejs stworzony na potrzeby narzędzia docker).

Adres MAC jest przypisany do każdego z interfejsów osobno i są to kolejno:

- 00:00:00:00:00:00
- 74:e5:f9:86:f9:0c
- 02:42:36:87:00:08

Jak już wspomnieliśmy, adres MAC jest częścią ramki, której obsługą zajmuje się warstwa druga modelu OSI. Uproszczony schemat takiej ramki zamieszczono na Rysunku 3.

Kolejno adres MAC odbiorcy i nadawcy służą do tego, żeby ramka danych dotarła do określonego urządzenia, a także by odbiorca wiedział, do kogo wiadomość odesłać.

Typ danych informuje o formacie danych w polu Dane, co pozwala na ich prawidłową interpretację. W naszym przypadku zazwyczaj będziemy mówić o danych pakietu IP, o którym będzie mowa w dalszej części artykułu.

Ramkę kończy FCS – Frame Check Sequence – matematycznie wyliczony skrót danych, pozwalający określić, czy dane zostały odebrane prawidłowo. Czasami w wyniku zakłóceń elektromagnetycznych lub innych problemów w warstwie fizycznej może dojść do błędów transferu danych. Taki błąd może się objawić chociażby odebraniem złego bitu danych. Wysyłający wylicza wartość FCS, przesyła go do odbiorcy, który jeszcze raz liczy FCS i sprawdza jego poprawność. Jeśli wartości są sobie równe, to z dużym prawdopodobieństwem można stwierdzić, że dane zostały odebrane bez problemu.

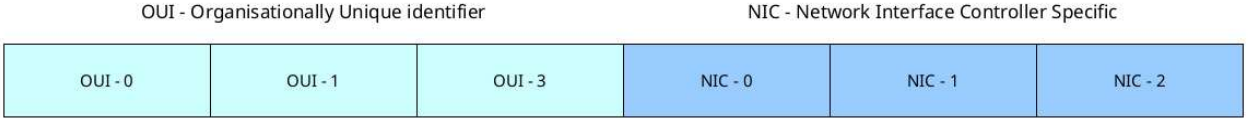
Chociaż istnieje teoretyczna możliwość błędu w przesyle danych, że zarówno dane, jak i FCS ulegną modyfikacji i wartość FCS nadal będzie poprawnie wyliczona dla niepoprawnie przesłanych danych, prawdopodobieństwo takiego zdarzenia jest niezwykle małe (w praktyce nie trzeba się tym przejmować, a protokoły wyższych warstw również implementują algorytmy sprawdzające spójność danych).

Urządzenia warstwy 1 i 2.

Zanim przejdziemy do omawiania kolejnych warstw i ich implementacji, spójrzmy na 2 urządzenia – huby oraz switche. Są to odpowiednio urządzenia warstwy pierwszej i drugiej.

Zarówno do hubów, jak i switchy możemy podłączyć wiele urządzeń, co umożliwi im komunikację ze sobą. Jednak w obu przypadkach ta komunikacja będzie wyglądać nieco inaczej.

Huby to urządzenia warstwy pierwszej. Jako takie nie rozumieją one ramek danych warstwy drugiej (Rysunek 3). Oznacza to, że wszystkie urządzenia podłączone do huba, komunikując się, rozsyłają



Rysunek 2. Opis adresu MAC

ramki po całej sieci (obciążona jest cała sieć, nie tylko połączenie z poszczególnym urządzeniem). Na przykład jeśli do huba podłączonych jest 10 urządzeń, a chcemy ramkę wysłać tylko do jednego z nich, to i tak ramkę tę otrzyma każde urządzenie. Dopiero urządzenia odbierające ramki danych są w stanie stwierdzić, czy ramka danych była zaadresowana do nich, i odpowiednio je filtrują (większość kart sieciowych można skonfigurować tak, aby ramki adresowane do innych urządzeń nie były odrzucane, wówczas możemy spróbować podsłuchiwać całą komunikację należącą do innych urządzeń).

Kolejną wadą podłączania urządzeń w jedną sieć poprzez hub jest występowanie przeciążenia sieci przy większej ilości urządzeń i przy zwiększonym ruchu. Skoro każdy pakiet danych trafia do każdego urządzenia podłączonego do huba, nietrudno sobie wyobrazić, że przy większej ilości przesyłanych danych urządzenia mogą mieć problem z uzyskaniem wyłącznego dostępu do warstwy fizycznej. Jeżeli 2 urządzenia w takiej sieci zaczną wysyłać sobie wiele danych (np. wysyłając duże pliki), zakłócą one komunikację reszty urządzeń w sieci.

Switche jako urządzenia warstwy drugiej nie mają takiego ograniczenia. Rozumiejąc ramki danych i „widząc" ruch na portach, zapamiętują, jakie urządzenia są podłączone do którego portu switcha. Ramki nie są więc rozsyłane po wszystkich przewodach, a tylko do tych, gdzie są urządzenia, które tych danych potrzebują. Zmniejsza to obciążenie na warstwie fizycznej, co zwiększa niezawodność komunikacji sieci.

Ramki danych w Wireshark

Skoro znamy już ramki danych, spójrzmy na nie przez pryzmat aplikacji Wireshark.

Wireshark to analizator sieciowy. Możemy dzięki niemu diagnozować problemy związane z połączeniami, pozwala nam podejrzeć komunikację aplikacji z serwisami, a także komunikację między urządzeniami a routerem. Jeśli pojawią się problemy z komunikacją, powinniśmy je również zobaczyć w aplikacji.

Uruchomienie aplikacji

Po uruchomieniu aplikacji powinno wyświetlić się okienko widoczne na Rysunku 4.

Na głównym planie widzimy listę interfejsów sieciowych, które utworzył system operacyjny. Mogą być to interfejsy bezpośrednio związane z fizyczną siecią (np. z siecią bluetooth czy W-Fi – wlp2s0), inne interfejsy mogą być czysto wirtualne (np. loopback czy docker0).

Interfejs loopback odpowiada za komunikację aplikacji w ramach jednego systemu operacyjnego (aplikacja działająca na jednym komputerze wysyła wiadomość do innej aplikacji działającej na tym samym komputerze, a pakiety danych nigdy fizycznie nie opuszczają urządzenia). Interfejs docker natomiast jest interfejsem dedykowanym dla serwisu docker, którego nie będziemy w tym artykule omawiać.

Na tym etapie możemy dwukrotnie kliknąć na wybrany interfejs. Wireshark natychmiast rozpocznie zbieranie pakietów z wybranego interfejsu.

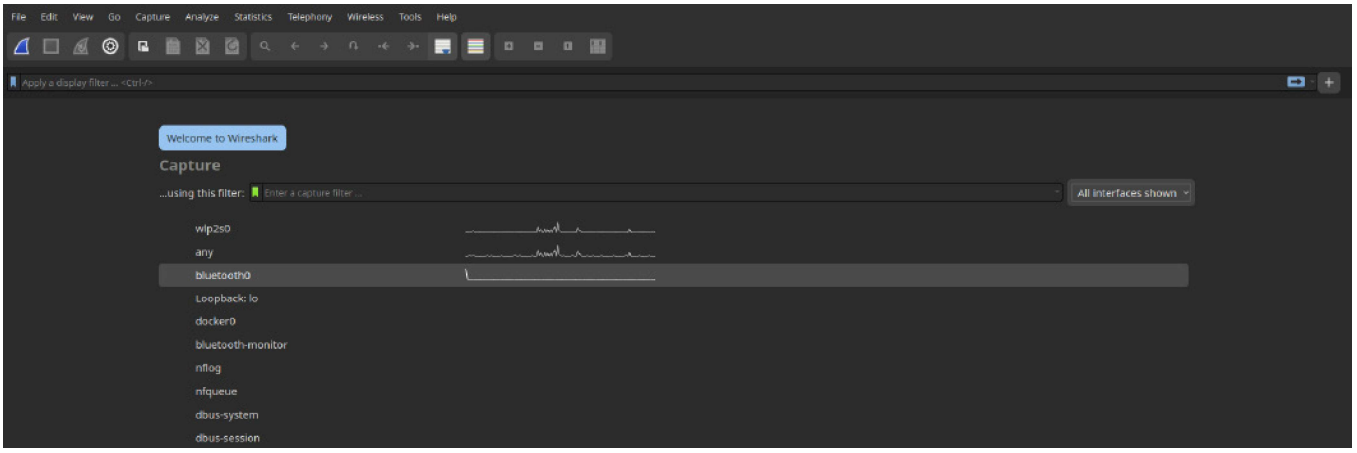
Warto jeszcze zwrócić uwagę na 2 pola – na samej górze przy niebieskiej zakładce – display filter, a także capture filter.

Jak same nazwy wskazują, capture filter (zaraz nad listą interfejsów) odpowiada za pakiety, które aplikacja będzie rejestrować. Jego głównym zadaniem jest redukcja ilości danych przetwarzanych przez program. Ma to szczególne znaczenie dla długich sesji lub rejestrowania zatłoczonych środowisk (gdzie potencjalnie będziemy rejestrować bardzo dużo danych). Niezarejestrowane pakiety są na stałe zgubione. Więcej o tym rodzaju filtra w [5].

Display filter natomiast to filtr, według którego aplikacja będzie nam wyświetlać pakiety, które już zostały zarejestrowane. Ma nam on pomóc w analizie ruchu. W ramach jednego capture filtra będziemy wielokrotnie modyfikować display filter, aby dostosować wyświetlany ruch do tego, co nas interesuje. Więcej na ten temat można przeczytać w [6].

Kolejny ekran będzie już zawierał podgląd wszystkich pakietów, aktualizowany w czasie rzeczywistym, zarejestrowanych z użyciem capture filter i odfiltrowany przez display filter.

Rysunek 3. Opis ramki danych



Rysunek 4. Aplikacja Wireshark zaraz po uruchomieniu

Wireshark na przykładzie ARP (Address Resolution Protocol)

ARP służy do umożliwienia komunikacji między urządzeniami w ramach jednej podsieci. Mimo że nie omawialiśmy jeszcze protokołu IP (to za chwilę), na potrzeby omawiania protokołu ARP zapamiętajmy jedynie, że aplikacje adresują wiadomości nie z użyciem adresu MAC, tylko z użyciem adresu IP. Dopiero niższe warstwy są odpowiedzialne za używanie adresów MAC.

Dzięki protokołowi ARP możemy uzyskać adres MAC, posiadając adres IP – ale tylko w ramach jednej podsieci. Jak już wspomnieliśmy wcześniej, potrzebujemy adresu MAC do tego, aby móc poprawnie zaadresować ramki.

Przyjrzyjmy się kilku ramkom takiej komunikacji. Nie będziemy w szczegółach przyglądać się protokołowi, ale użyjemy go jako przykład, żeby zobaczyć ramkę danych.

W szczególności przyjrzymy się trzem rodzajom komunikatów ARP: ogłoszenie – zapytanie o adres MAC (request) – odpowiedź z adresem MAC (reply).

Komunikaty te przedstawiono na Rysunku 5.

W aplikacji Wireshark możemy odfiltrować wiadomości ARP, wpisując po prostu w pole display filtru nazwę protokołu – `arp`.

Ogłoszenie ARP

Zacznijmy od ogłoszenia ARP. Takie ogłoszenie możemy zaobserwować, kiedy w sieci pojawiają się nowe urządzenia. Możemy je wywołać sztucznie, chociażby wywołując komendy `ip link set <interface> down && ip link set <interface> up` na systemach Linux. Komendy te odpowiednio wyłączają i włączają interfejs sieciowy. Działa to mniej więcej tak, jakbyśmy odłączyli i podłączyli kartę sieciową ponownie do urządzenia.

Na zrzucie ekranu widzimy listę pakietów, zarejestrowanych na interfejsie sieciowym i odfiltrowanym przez display filter. Na Rysunkach 6 i 7 widzimy szczegóły pakietu. Po lewej stronie pakiet jest rozpisany według warstw. Podkreślona część wskazuje właśnie na warstwę drugą – MAC. Po prawej stronie widzimy reprezentację bajtową pakietu. Podkreślona część – warstwa druga – jest również zaznaczona po prawej stronie.

Kolejne części wiadomości możemy klikać, a odpowiadający kod hex zostanie podświetlony po stronie prawej, co również przedstawiono na Rysunkach 6 i 7.

Widzimy tutaj adres odbiorcy, nadawcy, typ wiadomości (ARP). Nie widzimy natomiast segmentu FCS. Segment ten jest wysyłany, ale często urządzenia sieciowe nie podają go do systemu operacyjnego, a ten nie podaje do aplikacji (patrz [13]).

Ogłoszenie ARP jest wysyłane przez urządzenie, kiedy zmienia się jego adres MAC lub IP. Dzięki takiemu ogłoszeniu urządzenia w sieci mają aktualne informacje o adresach pozostałych urządzeń w sieci.

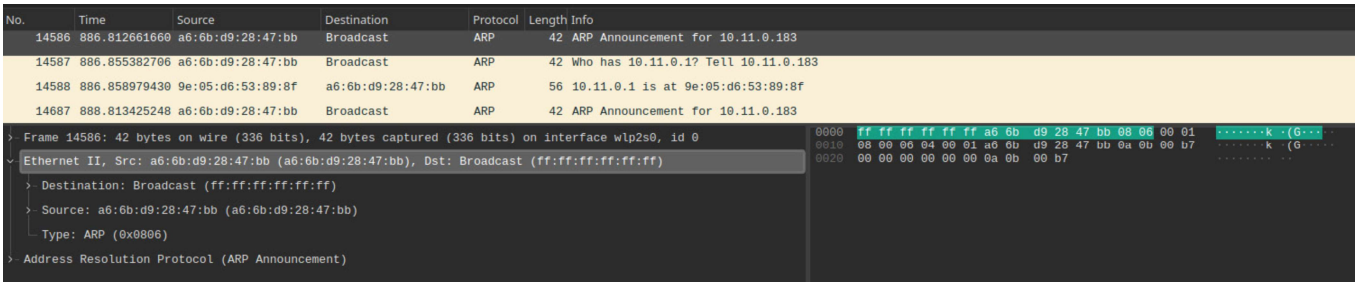
Ogłoszenie ARP ma znanego nadawcę – jest to wspomniane urządzenie, które zmieniło adres, nie ma jednak określonego odbiorcy. Ogłoszenie, jak sama nazwa wskazuje, ma w końcu trafić do wszystkich urządzeń w sieci. Widać to w nietypowym adresie MAC odbiorcy – `ff:ff:ff:ff:ff:ff`. Taki adres MAC oznacza właśnie, że wiadomość jest przeznaczona do wszystkich urządzeń.

Zapytanie i odpowiedź ARP

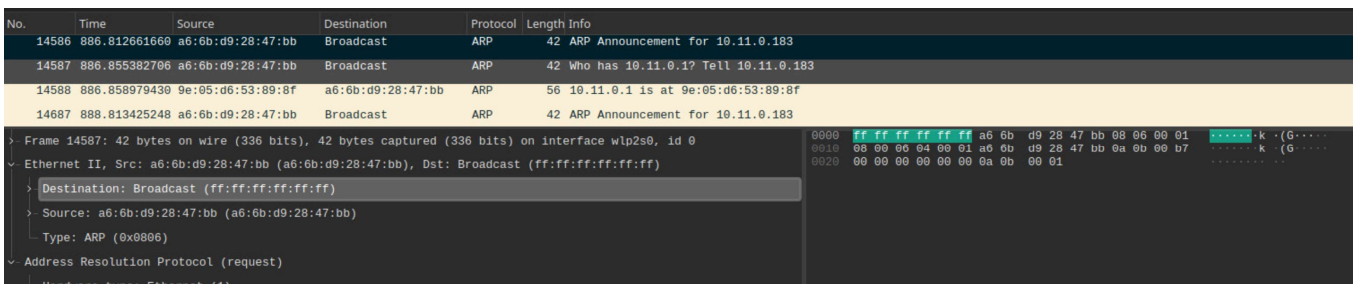
Pozostałe 2 pakiety to zapytanie o adres MAC urządzenia o danym adresie IP.

Ogłoszenia ARP są niewystarczającym sposobem na aktualizację danych o urządzeniach w sieci, ponieważ urządzenia podłączone do sieci po ogłoszeniu nigdy nie otrzymają wiadomości. Jest to jednak efektywny sposób komunikacji (wystarczy jeden komunikat, aby poinformować wszystkie urządzenia w sieci o zmianie).

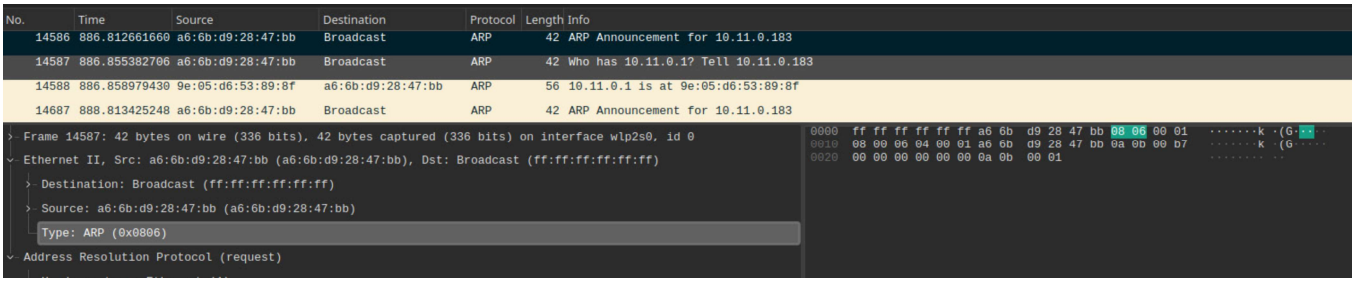
Przeciwieństwem jest komunikacja na zasadzie pytanie-odpowiedź. Jest to wystarczający sposób aktualizacji danych, ale mało efektywny (urządzenia muszą pytać za każdym razem o aktualny adres MAC danego adresu IP).



Rysunek 5. Pakiety ARP w aplikacji Wireshark



Rysunek 6. Podkreślanie adresu MAC odbiorcy w ramce



Rysunek 7. Podkreślenie typu danych w ramce

Zapytanie takie przedstawiono na Rysunku 8.

Widzimy również, że Wireshark w intuicyjny sposób przedstawia nam treść zapytania w kolumnie `Info` – „Who has {ip address}? Tell {my ip address}".

Zapytanie to jest również rozgłoszeniem. Ponieważ nie znamy adresu MAC odbiorcy (gdybyśmy znali, zapytanie nie byłoby w ogóle potrzebne), zapytanie musi trafić do wszystkich urządzeń w sieci. Urządzenie o danym adresie IP, znając już adres MAC nadawcy, może odpowiedzieć bezpośrednio do niego, co widać w następnej paczce danych, gdzie zarówno kolumna `source`, jak i `destination` ma konkretne wartości.

Warto pamiętać, że wszystkie wiadomości „rozgłoszeniowe" – czy to ARP, czy innych protokołów – zajmuje część przepustowości łącza wszystkich podłączonych urządzeń, dlatego ilość urządzeń w jednej sieci powinna być ograniczona, a ilość danych rozgłoszeniowych kontrolowana. W przeciwnym razie wydajność naszej sieci może być niezadowalająca.

Warstwa sieci i adresy IP

Jak już wspomnieliśmy, tworzenie sieci złożonej z dużej ilości urządzeń nie jest najlepszym pomysłem. Ponadto aplikacje komunikujące się ze sobą często nie znają adresów MAC swoich urządzeń.

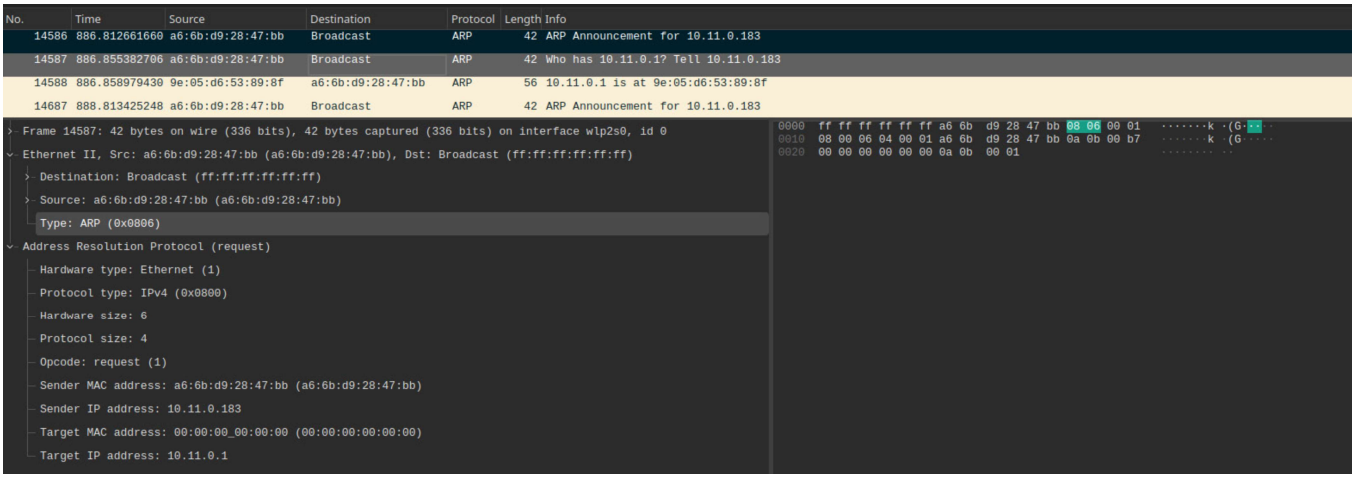
Z pomocą przychodzi właśnie warstwa sieci razem z adresami IP.

Tak jak adresy MAC są fizycznym sposobem adresowania (w uproszczeniu są nadawane urządzeniom na etapie ich produkcji), tak adresy IP są logicznym sposobem adresowania. Analogicznie możemy myśleć o telefonach i numerach telefonów. Każdy aparat ma swój numer seryjny nadany podczas jego produkcji. Dopiero po jego zakupie nadawany jest mu numer telefonu. Co więcej, urządzenie możemy sprzedać, co nie zmieni jego numeru seryjnego, ale zmieni numer telefonu, lub możemy zmienić numer telefonu, nie zmieniając numeru seryjnego. Podobnie ma się adres IP do adresu MAC (w praktyce karty sieciowe oraz sterowniki pozwalają również na softwareową zmianę adresu MAC).

W sieci urządzenia mają nadawane adresy IP przez administratorów, podobnie jak numery telefonów przez operatorów sieci telefonicznej. Oczywiście nie oznacza to manualnego ustawiania adresu IP dla każdego urządzenia (chociaż można i tak). Obecnie w ramach jednej sieci najczęściej urządzenia mają nadawanie adresy IP automatycznie (chociażby przez protokół DHCP, którego nie musimy znać w szczegółach).

Co do formatu danych, podobnie do protokołu ARP, adres IP jest „dorzucony" do ramki danych warstwy drugiej, stając się *pakietem*.

Jego schemat został zaprezentowany na Rysunku 9.



Rysunek 8. Zapytanie ARP

Rysunek 9. Uproszczony pakiet IP w ramce danych

Jak widać, w miejsce na dane w ramce wrzucony jest format pakietu IP (w dużym uproszczeniu). Kolejne warstwy, które będą dokładać więcej danych, będą działać na podobnej zasadzie – w pole Dane warstwy niższej będą wrzucać swoje formaty danych.

Jak nietrudno się domyślić, im więcej warstw protokołów, tym więcej metadanych musimy przesyłać przez sieć. Naturalnym jest więc, że pożądaną cechą protokołów komunikacji jest ich niski rozmiar.

Przejdźmy teraz do nieco praktyczniejszej strony warstwy trzeciej, zaczynając od opisu adresów IP.

Adresy IP

Adresy IP obecnie dostępne są w dwóch wariantach – wariancie 4-bajtowym (IPv4) oraz 16-bajtowym (IPv6). W tym artykule będziemy głównie rozważać starszy wariant – 4-bajtowy.

IPv4 to nadal najbardziej popularny format, którym posługujemy się na co dzień.

Budowa adresu IP

Jak już wspomnieliśmy, adres ten jest zbudowany z czterech bajtów i zapisywany zazwyczaj jako sekwencja czterech liczb od 0 do 255 oddzielonych kropkami. Kolejne liczby odpowiadają kolejnym bajtom adresu. Przykład takiego adresu to: 127.0.0.1.

Adresy IP często są przydzielane zakresami do grup urządzeń. Taka grupa urządzeń dostaje pulę adresów IP. Adres IP dzieli się wtedy niejako na 2 części – identyfikującą sieć i identyfikującą urządzenie w sieci. Żeby wiedzieć, która część odpowiada za adres sieci, a która za konkretne urządzenie, wprowadzono pojęcie maski sieci. Maska sieci ma ten sam format co sam adres IP, często można spotkać zapis typu CIDR (o czym poniżej). Często wygląda ona tak: 255.255.255.0.

Kto zna operacje bitowe, wie, że taki adres odpowiada kolejno binarnie: 11111111.11111111.11111111.00000000.

Każdy włączony bit maski odpowiada bitowi adresu IP odpowiedzialnego za adres sieci. Taka maska razem z adresem IP 192.168.0.10 mówi o tym, że część adresu `192.168.0` identyfikuje sieć, natomiast `10` odnosi się do konkretnego urządzenia (hosta) w tejże sieci.

Jednym ze sposobów alokacji adresów są klasy przypisane poszczególnym bajtom adresu. Organizacje mogły rezerwować/wykupować określone klasy adresów IP. Podział na klasy ilustruje następujący diagram:

Rysunek 10. Klasowy podział adresów IP

W zależności od klasy wykupionych adresów IP ilość adresów IP, które mogły być przypisane urządzeniom, była różna. Na przykład w najwyższej klasie A jedynie pierwszy bajt odpowiadał za adres sieci, pozostałe 3 bajty adresów można wykorzystać do adresowania urządzeń. Kolejno klasa B oferowała jedynie 2 bajty adresów dla urządzeń, a klasa C tylko jeden bajt.

Podział klasowy nie sprawdzał się jednak zbyt dobrze. Okazało się, że w ten sposób wiele adresów IP jest po prostu marnowanych i potrzebny był nowy sposób przydzielania adresów, czyli właśnie CIDR.

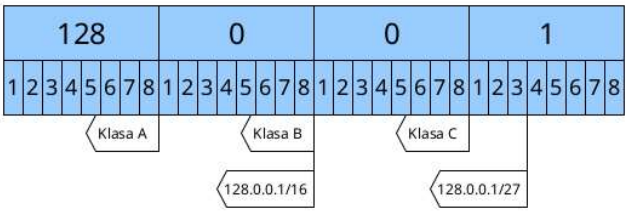
CIDR

CIDR (Classless Inter Domain Routing) to adresy, w których ilość danych odpowiedzialnych za adres sieci, a więc i ilość adresów pozostałych do wykorzystania wewnątrz tej sieci, określa się z dokładnością co do bitu.

Do określenia takiego zakresu podajemy po znaku / liczbę bitów odpowiedzialnych za adres sieci. Skoro każdy bajt to 8 bitów, to możemy łatwo zdefiniować klasy za pomocą adresów CIDR:

- klasa A – /8
- klasa B – /16
- klasa C – /24

Zostało to zilustrowane na poniższym diagramie:



Rysunek 11. Bezklasowy podział adresów IP – CIDR

Zastanawiające może być, co się dzieje z wartością 1 w ostatnim bajcie adresu w przypadku, kiedy podajemy adres sieci taki jak np. `128.0.0.1/16`. Ostatnia wartość (`1`) nie należy w żaden sposób do adresu sieci, ponieważ bajt ten nie należy do maski (wartość nie jest zawarta w pierwszych 16 bitach adresu). W zależności od implementacji bajt ten może być po prostu ignorowany lub powodować błędy (lepiej w takiej sytuacji zerować bity nie należące do adresu sieci).

Zarezerwowane adresy IP

Kolejną funkcjonalnością, którą dostajemy wraz z adresowaniem IP, są zarezerwowane zakresy adresów IP, które nie mogą być używane jako publiczny adres IP (adres IP widoczny przez wszystkich w sieci Internet).

Jest to sposób na ograniczenie zużycia puli adresów. Dzięki temu zarówno mój komputer w mojej domowej sieci, jak i urządzenia czytelnika w jego domowych sieciach mogą mieć takie same adresy wewnętrzne IP, przez co nie wyczerpują puli wszystkich adresów. Oczywiście nasze adresy widziane przez urządzenia w Internecie będą się różniły, bo adresy IP routerów naszych dostawców Internetu będą różne.

Skutkiem takiego podziału adresów jest to, że urządzenia nie mogą ot tak zainicjować wysyłania danych do urządzeń nie mających publicznych adresów IP.

Wszystkie urządzenia w ramach jednej podsieci z prywatnymi adresami IP reprezentuje jeden wspólny publiczny adres IP, o czym dokładniej opowiemy przy omawianiu routerów w sekcji „Routery, LAN i WAN".

Zakresy zarezerwowane dla adresów IP to:

- Class A: 10.0.0.0 to 10.255.255.255
- Class B: 172.16.0.0 to 172.31.255.255
- Class C: 192.168.0.0 to 192.168.255.255

W praktyce spotkamy się jeszcze z kilkoma specjalnymi adresami. Na przykład adres `255.255.255.255` jest adresem broadcastowym, tzn. wysyłanie pakietów na takie adresy kończy się dostarczeniem ich do wszystkich urządzeń w sieci lokalnej.

Adres `127.0.0.1` to tzw. adres loopback. pakiety wysyłane na ten adres trafią z powrotem do tego samego urządzenia (tak naprawdę nigdy tego urządzenia nawet nie opuszczą).

Podczas konfiguracji programów na urządzeniach adres `0.0.0.0` będzie oznaczał nasłuchiwanie przychodzących pakietów dla dowolnego adresu IP nadawcy (jedno urządzenie może mieć wiele adresów). Przy nadawaniu pakietów (przynajmniej w systemach operacyjnych Linux) będzie oznaczało to nadawanie pakietów na adres 127.0.0.1.

DNS

No dobrze, tylko skąd mamy znać adresy IP? W końcu chcąc wyszukać coś w Internecie, nie musimy posługiwać się adresami IP, a jedynie nazwami witryn internetowych jak np. google.com.

W skrócie, odpowiada za to system DNS, który (wracając do analogii z telefonami) działa jak książka telefoniczna (dla młodszych czytelników: dosłownie książka z imionami oraz nazwiskami ludzi z odpowiadającym im numerem telefonu) ze wszystkimi adresami IP (prawie). Szczegóły działania tego systemu opowiemy sobie w innym artykule, na razie wystarczy, że wiemy, że coś takiego istnieje.

Routery, LAN i WAN

To właśnie routery są urządzeniami, które operują na warstwie trzeciej modelu OSI. Przez nie podłączamy się do sieci dostawcy Internetu.

Jeśli chodzi o wygląd zewnętrzny routera, to jest on całkiem podobny do switcha, jednak jego zasada działania się różni. Warto dodać, że routery w domowych warunkach często używane są właśnie jako switche – jako urządzenia trzeciej warstwy nie mają problemu ze zrozumieniem warstwy drugiej. Routery „domowe" to tak naprawdę urządzenie 3 w jednym. Pełnią one funkcje routera, switcha oraz access pointu (access point – punkt dostępu do sieci bezprzewodowej). Jeżeli czytelnik nigdy nie miał okazji obcować z tego typu sprzętem, to gorąco zachęcam do zakupu (jeśli nie posiadasz) jakiegoś budżetowego routera, podpięcie go między router dostawcy a komputer i pobawienie się nim – pogrzebanie w ustawieniach, doczytanie o różnych parametrach, czym jest VLAN itp.

Jak już wspominaliśmy, switche to urządzenia, które odbierają od jednego urządzenia ramkę danych, analizują adres MAC odbiorcy i przekazują ramkę danych na odpowiedni port (przewód podłączony do urządzenia).

Na co w takim razie pozwalają routery, czego nie mogą robić switche? Podpowiedź znajdziemy na obudowie routera. Jeśli spojrzymy na router w swoim domu, jeden z portów najprawdopodobniej jest oznaczony jako WAN (często ma inny kolor). Oczywiście nie jest to regułą, czasami port WAN jest konfigurowany poprzez software i wiele portów może pełnić rolę portu WAN. Jeszcze innym „udziwnieniem" może być posiadanie wielu portów WAN jednocześnie, co zazwyczaj jest cechą specyficzną dla nieco droższych modeli routerów.

WAN (Wide Area Network) to sieć, która składa się z wielu mniejszych sieci. Często mówi się, że sieć WAN jest rozległa geograficznie, ale wcale tak nie musi być. Niejako przeciwieństwem do niej są sieci LAN (Local Area Network), które nie mają podsieci i zazwyczaj ograniczają się do jednego mieszkania.

Port WAN (Wide Area Network) jest to port, który łączy sieć LAN z WAN. Sam router jest niejako mostem między sieciami, który wie, jak przetransportować pakiety z sieci zewnętrznej do wewnętrznej i odwrotnie. Aby to zrobić, musi znać adresy IP sieci wewnętrznej oraz adres sieci zewnętrznej.

Routery często mają wiele adresów IP. Najczęściej jeden adres IP dla sieci LAN i oddzielny adres IP dla sieci WAN.

Przykładowy schemat takiej sieci może wyglądać jak na Rysunku 12.

Widzimy, że mamy 2 przykładowe sieci LAN1 i LAN2. Wszystkie urządzenia w tych sieciach są podłączone do jednego urządzenia switch, które następnie jest podłączone z routerem (w praktyce w sieciach domowych znajdziemy jedno urządzenie, które jest jednocześnie routerem i switchem).

Router ma 2 adresy: jeden adres w sieci „LAN", drugi w sieci „WAN". Warto również zaznaczyć, że z punktu widzenia sieci LAN1 i LAN2 siecią WAN będzie sieć dostawcy Internetu. Router dostawcy Internetu również jest połączony z zewnętrzną siecią – odbiera paczki z „lokalnej sieci" i przekazuje je dalej. Zazwyczaj, kiedy wysyłamy jakikolwiek pakiet do Internetu, przechodzi on co najmniej przez kilka routerów. Można to łatwo sprawdzić chociażby programem traceroute.

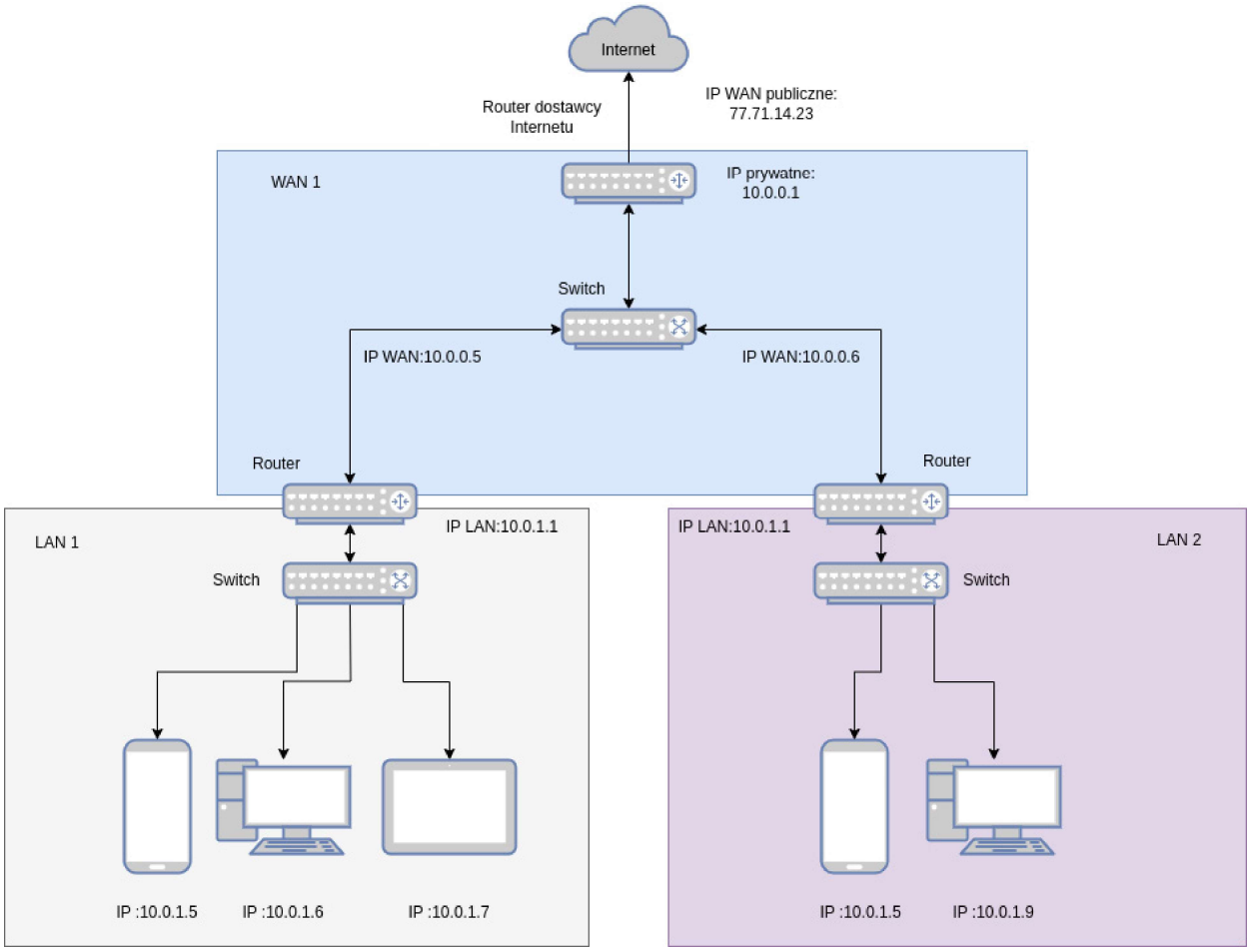
Listing 1. Przykład użycia programu traceroute

```
$ traceroute google.com
traceroute to google.com (142.250.186.206)
 1  _gateway (10.11.0.1)
 2  10.0.2.1 (10.0.2.1)
 3  10.100.0.1 (10.100.0.1)
 4  xxxx
 5  xxxx
 6  155.133.104.168 (155.133.104.168)
 7  72.14.219.142 (72.14.219.142)
 8  142.251.225.169 (142.251.225.169)
 9  142.250.239.81 (142.250.239.81)
10  waw07s05-in-f14.1e100.net (142.250.186.206)
```

Jak widać, aby pakiet dostał się do serwera google.com, pokonuje 10 różnych sieci (dla każdego wypisany jest adres IP). Widzimy więc, że sieć Internet jest mocno hierarchiczna (sieci zagnieżdżone są w sieciach, które znowu zagnieżdżają inne sieci).

Widzimy też, że gdy system operacyjny chce wysłać dane poza sieć lokalną, kieruje ją właśnie do routera (używając jego adresu MAC), a ten szuka odbiorcy wiadomości po zewnętrznej stronie sieci. Adres IP routera w sieci (np. LAN), który łączy ją z szerszą siecią (WAN), nazywany jest domyślna bramką (ang. *default gateway*). Za każdym razem, kiedy pakiet jest adresowany (adresem IP) do urządzenia spoza sieci (co określamy przez adres IP sieci oraz jej maskę), adres MAC odbiorcy w wysyłanej ramce jest ustawiany na adres MAC routera. Jeśli router nie widzi urządzenia o podanym adresie IP w sieci WAN, przesyła pakiet dalej do swojej domyślnej bramki itd. aż ten dotrze na miejsce.

Wspominaliśmy o tym, że tworzenie sieci LAN zbudowanej z wielu urządzeń nie jest dobrym pomysłem (zależy to od wielu czynników, takich jak przepustowość sieci czy intensywność ruchu sieciowego, ale mówimy tutaj o dziesiątkach urządzeń nie kilku). Widzimy też, w jaki sposób routery i adresy IP rozwiązują ten problem przez dzielenie sieci na mniejsze segmenty.



Rysunek 12. Przykład topologii dwóch sieci LAN w sieci WAN dostawcy Internetu

Warto wspomnieć, że schematy działania są tutaj uproszczone. Sieci tworzone przez routery często nie mają wyłącznie prywatnych adresów IP. Prawdopodobnie nasz dostawca pozwala na wykupienie dedykowanego, publicznego adresu IP. W takim przypadku do naszego routera domowego będzie przypisany adres IP i będziemy mogli w dowolnym momencie wysłać do niego pakiet danych, będąc poza siecią LAN.

PODSUMOWANIE

Sieci komputerowe są bardzo skomplikowanym zagadnieniem. Aby wszystko ze sobą „grało", organizacje stworzyły wiele standardów opisujących najróżniejsze aspekty połączeń sieciowych – od warstwy fizycznej opisującej rodzaje przewodów oraz komunikację bezprzewodową do systemowych API pozwalających na tworzenie aplikacji serwerowych i klienckich.

W tym artykule poruszyliśmy warstwę fizyczną, łącza danych i nieco warstwę sieciową. Do pełniejszego zrozumienia musimy jeszcze zapoznać się z systemami DNS, niektórymi z protokołów TCP/IP, a także powinniśmy omówić przynajmniej jeden protokół działający na protokole TCP/IP – w naszym przypadku będzie to HTTP.

Wszystkie te zagadnienia zostaną poruszone w następnych artykułach z serii „Jak działa Internet". Do usłyszenia!

DAWID PILARSKI

dawid.pilarski@panicsoftware.com

Z wykształcenia automatyk i robotyk, a z zawodu i pasji programista. Obecnie Software Engineer, Security Champion i Tech Lead w TomTom. Wolny czas przeznacza na zgłębianie wiedzy o bezpieczeństwie i sieciach. Można się z nim skontaktować poprzez adres e-mail: dawid.pilarski@panicosftware.com (w celach zawodowych) oraz dawid.pilarski@pm.me (w celach prywatnych).

Bibliografia

1. CompTIA Network+ Certification All-in-One Exam guide
2. https://en.wikipedia.org/wiki/Domain_Name_System
3. https://www.geeksforgeeks.org/introduction-of-classful-ip-addressing/
4. https://linux.die.net/man/8/ip
5. https://wiki.wireshark.org/CaptureFilters
6. https://wiki.wireshark.org/DisplayFilters
7. https://en.wikipedia.org/wiki/Address_Resolution_Protocol
8. https://aws.amazon.com/what-is/cidr/
9. https://whatismyipaddress.com/reserved-ip-address-blocks
10. https://www.techtarget.com/searchnetworking/definition/local-area-network-LAN
11. https://www.cisco.com/c/en/us/products/switches/what-is-a-wan-wide-area-network.html#~what-it-is
12. https://www.cisco.com/c/en/us/solutions/small-business/resource-center/networking/network-switch-vs-router.html
13. https://ask.wireshark.org/question/19874/where-is-the-fcs-field-shown-in-the-wireshark-output/
14. https://docs.redhat.com/en/documentation/red_hat_virtualization/4.3/html/technical_reference/virtual_network_interface_controller_vnic

Zbuduj własnego Linuxa z Buildrootem i Raspberry Pi

Własna dystrybucja Linuxa zbudowana od podstaw – brzmi bardzo ambitnie. W końcu to, co potocznie rozumiemy przez „Linuxa", to w rzeczywistości połączenie kilku różnych komponentów, takich jak np. kernel, główny system plików (rootfs), bootloader czy init system (np. systemd). W tym artykule poznamy narzędzie Buildroot oraz krok po kroku stworzymy funkcjonalny obraz systemu na platformę Raspberry Pi 4. Następnie dodamy dodatkowe programy i sprawimy, by WiFi łączyło się automatycznie po starcie systemu.

CZYM JEST BUILDROOT?

Każdy z wymienionych komponentów jest skomplikowany, konfigurowalny na wiele sposobów i potencjalnie dostosowywalny pod dowolne rozwiązanie. Wymaga również włożenia pracy w dostosowanie pod konkretną platformę. Stąd też ciągle rosnąca popularność Linuxa w zastosowaniach, gdzie wcześniej zazwyczaj były używane systemy bare metal. Za cenę większego obrazu systemu i wyższych wymagań sprzętowych dostajemy m.in. dostęp do ogromnej (i darmowej) biblioteki projektów open source, przetwarzania audio/video, gotowych stosów sieciowych. Problem w tym, że dedykowany system trzeba skonfigurować, skompilować i wszystko poukładać zgodnie ze sztuką.

Dlaczego nie skorzystać z istniejących, gotowych systemów np. opartych o Debiana? System operacyjny dla urządzeń embedded musi spełniać nieco odmienne wymagania niż system ogólnego przeznaczenia. Pożądana jest kombinacja komponentów, którą możemy dokładnie dostosować pod konkretne zastosowania i zintegrować zewnętrzne programy klienta, często niedostępne publicznie. Ważne, żeby system zawierał tylko wymagane programy, zajmował jak najmniej zasobów sprzętowych i możliwie szybko się uruchamiał. Niezbędna jest też kontrola nad procesem budowania, możliwość zintegrowania w przyszłości systemu na inne platformy sprzętowe i zapewnienie wieloletniego wsparcia. Należy również pamiętać o konieczności dostosowania całości pod wymagające standardy bezpieczeństwa.

W artykule przedstawione jest rozwiązanie Buildroot, które automatyzuje budowanie dedykowanych dystrybucji Linuxa, w szczególności na urządzenia wbudowane. Przyjmuje określoną konfigurację – listę paczek, konfigurację jądra itp. – i wytwarza gotowy do użycia na docelowym urządzeniu obraz systemowy. W absolutnie minimalistycznym przypadku jest to:

- Bootloader
- Jądro Linuxa (kernel)
- Główny system plików (rootfs)

Bootloader to program, który zarządza początkiem startu systemu. Zostaje załadowany do pamięci przez firmware procesora, następnie inicjalizuje niezbędne zasoby sprzętowe (np. kontroler DRAM, interfejs UART, moduł zarządzania zasilaniem) i ładuje do pamięci kernel razem ze strukturą opisującą podłączony sprzęt (ang. *devicetree*).

Kernel, odpowiadający m.in. za kontrolę sprzętu oraz pamięci, inicjalizuje resztę podzespołów, a następnie uruchamia init system. Jest to pierwszy proces działający w userspace, odpowiedzialny m.in. za zamontowanie wirtualnych systemów plików takich jak /proc i /sys oraz konfigurację plików urządzeń w katalogu /dev.

Powyższy opis jest mocno uproszczony i jest bardzo zależny od konkretnej platformy sprzętowej. Są systemy nie obsługujące bootloadera takiego jak np. U-boot, a jedynie firmware odpowiadający za sprzęt i załadowanie kernela. Niektóre init systemy (np. systemd) mogą być tak ogromne, że praktycznie mogłyby być uznane za osobny system operacyjny, lub też tak małe, że ich rozmiar sprowadzi się do kilkudziesięciu linijek konfiguracji.

W zależności od tego, jaką wybieramy platformę i jak ją skonfigurujemy, poza kluczowymi modułami budowane będą również odpowiednie sterowniki i inne komponenty, np. stos sieciowy. Wynikiem końcowym jest obraz systemu, który jest gotowy do wgrania na urządzenie. W przypadku Raspberry Pi 4 jest to obraz na kartę SD.

Na koniec ważna informacja – **Buildroot nie jest dystrybucją Linuxa**. Jest narzędziem, które pozwala nam takie dystrybucje tworzyć, choć często potocznie mówi się, że dany system jest „zrobiony na Buildroocie" albo „Buildroot-based".

SPRZĘT – RASPBERRY PI 4

Buildroot ma bardzo bogate wsparcie dla różnorodnego sprzętu. Producenci zdają sobie sprawę, że środowiskiem uruchomieniowym ich platform będzie Linux. W związku z tym wsparcie takich projektów jak Buildroot jest zachętą dla kupujących. Buildroot jest również kompatybilny z QEMU, dzięki czemu działać możemy nawet zanim elektronika trafi na nasze biurko.

Do realizacji tego artykułu wybrałem platformę Raspberry Pi 4, ale dla każdej innej Raspberry Pi instrukcje będą zbliżone, o ile nie te same. „Malina" jest bardzo popularna w zastosowaniach hobbystycznych, na tle konkurencji jest również dość tania. Poza tym ma coś, co zdecydowanie wyróżnia ją z tłumu, czyli znakomite wsparcie społeczności. Na większość pytań można szybko znaleźć odpowiedź w Internecie, np. https://raspberrypi.stackexchange.com/ lub https://forums.raspberrypi.com/. Szczególnie bardzo polecam to drugie, udzielają się na nim regularnie developerzy i inni pracownicy Raspberry Pi Foundation.