Zajęcia 1: Wprowadzenie

2024-10-03

Twierdzenie 1 (Fourier)**.** Rozsądne funkcje okresowe wyrażają się szeregiem funkcji trygonometrycznych.

$$f(x) = c_0 + \sum_{i=1}^{\infty} a_i \sin(i \cdot x) + \sum_{i=1}^{\infty} b_i \cos(i \cdot x)$$

$$c_0 = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)dx$$

$$a_i = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin(i \cdot x)dx, \; b_i = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos(i \cdot x)dx$$

Dowód. Trygonometria jest *magiczna*. □

Przedstawiamy tak w punktach które chcemy, w reszcie jest inaczej, ale to nam nie przeszkadza. Intuicja: najpierw rozkładamy pole pod wykresem na całą powierzchnię (c_0), potem przesuwamy funkcjami.

Uwaga. Losowy szum (mała funkcja losowa) ma małe współczynniki w rozkładzie Fouriera, więc mało zmienia przy przekazywaniu sygnału.

Twierdzenie 2 (Nyquist)**.** Jeżeli funkcja f nie ma składowych o częstotliwościach większych niż B Hz i próbkujemy ją z częstotliwością $2B$ Hz, to możemy jednoznacznie odtworzyć f.

W szczególności więcej próbek nie ma sensu. Dlatego większa częstotliwość (np. światłowód) daje lepszą przepustowość niż mniejsza (kabel miedziany).

Wniosek. Maksymalna przepustowość to $2B \log \Sigma$.

Twierdzenie 3 (Shannon)**.** Jeżeli $\frac{S}{N}$ to stosunek mocy sygnału do mocy szumu, to maksymalna przepustowość to $B \log(1 + \frac{S}{N})$, gdzie B jest najwyższą częstotliwością sygnału.

Zajęcia 2: Warstwa łącza danych

2024-10-10

Model ISO-OSI:

- warstwa fizyczna
- warstwa łącza danych (głównie dalej poziom fizyczny, do tego np. rozwiązywanie problemów związanych z jednoczesnym nadawaniem przez różne urządzenia)
- warstwa sieci (tworzymy większe sieci za pomocą niższych warstw, które komunikują się na mniejszą skalę)
- warstwa transportowa
- warstwa sesji (ta i niższe warstwy umożliwiają programiście wykorzystywanie sieci)
- warstwa prezentacji
- warstwa aplikacji

Model TCP/IP:

- warstwa dostępu do sieci
- warstwa internetu
- warstwa transportowa
- warstwa aplikacji

Warstwa łącza danych (dostępu do sieci): chcemy jakiś system nadawania wiadomości, który umożliwi przekazywanie większych komunikatów. Chcemy umieć niwelować błędy w komunikacji, przede wszystkim

wykrywać je, potem usuwać. Często ważna jest możliwość potwierdzenia komunikacji (tego, że odbierający słucha).

Synchronizacja zegara: chcemy wiedzieć, kiedy kończy się jeden komunikat i zaczyna drugi. Komunikujemy się ciągami 0 i 1, więc chcemy zrobić tak, żeby poprawna wiadomość nie miała za długiego ciągu takich samych znaków pod rząd – jeśli odbierany sygnał często się zmienia, to w miarę łatwo jest dostosować się do interwałów, w jakich kolejne sygnały są wysyłane, a gdy jest podtrzymywany ten sam sygnał to nie wiemy, ile razy go liczyć.

System Manchester: każdy bit kodujemy jako dwa znaki, duża strata przepustowości, ale prawie nie ma powtórzeń.

System NRZI (non-return-to-zero inverted): zaczynamy od wysyłania sygnału (czyli przekazywania 1), na wystąpieniu jedynki zmieniamy sygnał (z 1 na 0, z 0 na 1), a 0 nie zmienia sygnału. W takiej sytuacji złe stają się tylko długie ciągi 0, bo 1 zawsze oznacza zmianę.

Można to połączyć w system NRZI + 4B/5B: każde 4 bity zamieniamy na 5 za pomocą tabelki, w której maksymalny ciąg 0 jest krótki.

Używa się też systemu 8B/10B, który dodatkowo sprawia, że przekazywane wiadomości są DC-balanced. Problem (DC-bias) polega na tym, że jeśli odbiornik otrzyma za dużo jedynek (sygnałów z energią) na raz, to może to zmienić stan jego odczytu. Dlatego stosuje się takie ciągi dziesięciobitowe, które mają w miarę równą ilość 0 i 1.

Inny pomysł: random scrambling, czyli xorowanie wiadomości z pseudolosową liczbą i nadanie tego, odbierający xoruje z tym samym i odczytuje.

Historia Ethernetu: na początku kabel kocentryczmy, dopiero potem skrętka. Popularny, bo jest otwartą technologią, do tego łatwo go dostosować: można puścić po światłowodzie, kablach miedzianych, etc.

Nadawanie większych komunikatów Ethernetem:

- nagłówek (zwłaszcza w starych Ethernetach, 8 bajtów na zmianę 1 i 0 – ułatwia synchronizację)
- adres odbiorcy (adres MAC, 6 bajtów)
- adres nadawcy
- typ protokołu / długość komunikatu (2 bajty)
- dane (46-1500 bajtów)
- suma kontrolna (czyli hash, 4 bajty)

Zajęcia 3: Wykrywanie błędów

2024-10-17

Parity bit – dodajemy na koniec bit mówiący, czy liczba jedynek jest nieparzysta (wiec ostatecznie w całej wiadomości jest parzyście wiele jedynek), daje około połowy szans na stwierdzenie, czy wiadomość jest z błędem

Odległość Hamminga – dla dwóch ciągów bitów tej samej długości jest to liczba różniących się bitów.

W celu wykrywania błędów będziemy przekształcać ciągi bitów długości 2^n na 2^m, gdzie $m > n$. Szukamy takiego przekształcenia $2^n \to 2^m$, w którym odległość Hamminga między wartością każdego argumentu to co najmniej $k+1$ – wtedy wykrywamy błąd na k bitach. Jeśli odległość między każdymi dwoma to co najmniej $2k+1$, to można odtworzyć nadany ciąg bez błędów – istnieje tylko jedna poprawna wiadomość, dla której mogła powstać taka wartość.

Ta tabelka musi być łatwo obliczalna i odwracalna, żeby to było praktyczne. Do tego chcemy szybko znajdować najbliższą poprawną wartość.

> **Przykład.** Chcemy korygować 10 błędów. Jak duże musi być n?
>
> **Dowód.** W każdej kuli (w sensie Hamminga) jest co najwyżej $\binom{m}{10} + \binom{m}{9} + \ldots \leq c \cdot m^{10}$ elementów. Zatem na pewno $2^m \geq 2^n \cdot cm^{10}$.

Haszowanie – haszujemy komunikat i doklejamy na koniec. Wykrywamy błąd porównując hash odebrany i wyliczony dla odebranych danych. Do korygowania możemy sprawdzać wszystkie możliwości (czy ich hash się zgadza z odebranym – jest to bardzo niepraktyczne, dlatego hashe zwykle służą tylko do wykrycia błędu).

CRC32 – rozszerzenie parity bita. Dla ciągu 2^n mamy wielomian nad $\mathbb{Z}_2$, gdzie współczynniki są takie jak wartości ciągu, np. $10010 \to x^4 + x$

Idea CRC32: dzielimy przez $x + 1$ i szukamy reszty. Sposób algorytmiczny: xorowanie (odejmowanie) każdej kolejnej pary bitów z naszego ciągu z 11. Resztą będzie parzystość jedynek w naszym ciągu, czyli parity bit.

W samym CRC32 będziemy rozważać wielomiany większe niż $x + 1$ i reszty z dzielenia przez nie. Dość dobrze działa $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$, on jest w Ethernecie (stąd nazwa hashowania – stopień wielomianu). Dla takiego wielomianu jesteśmy w stanie wykryć błędy do 3 bitów (zakładając ciąg bitów długości ramki Ethernet), dla większych błędów nie ma aż tak dużo pokrywających się wiadomości. Ethernet nie naprawia błędów, tylko je wykrywa i jest przy tym dość stabilny (bo jest na kablu), więc nie ma sensu robić silnego wykrywania i korekcji błędów.

Dzielenie kanału komunikacji. Nadawanie różnych komunikatów na raz powoduje, że nakładają się na siebie. Rozwiązaniem jest system (slotted) ALOHA.

Wysyłamy wiadomość, jak nie dostaniemy odpowiedzi, to był jakiś konflikt. Czekamy losową ilość czasu (żeby się znowu nie zderzyć) i powtarzamy. Jeśli słyszymy jakiś komunikat, to nie nadajemy – czekamy, aż będzie wolne. W kablu Ethernet to ma sens, ale przy sieciach radiowych niekoniecznie – możemy dostawać inne sygnały niż nasz odbiorca. Czas, jaki czekamy jest uzależniony od długości przekazywanych komunikatów (to jest właśnie *slotted* w nazwie systemu – każdy komunikat jest podobnie długi). Dlatego długość ramki Ethernet jest ustalona. Im dłużej nie udaje się nadać odebranego komunikatu, tym bardziej wydłużamy zakres z jakiego losujemy czas czekania (exponential back-off) – jak mamy konflikt z wieloma nadawcami, to małe czasy czekania nie wystarczą.

W Ethernecie wysłanie preambuły zajmuje tak długo, że każdy ją usłyszy. Dlatego jeśli każdy używa tego samego algorytmu, to po nadaniu preambuły jest pewność, że nikt inny nie nadaje.

W Ethernecie odbiorca jest ustalany po adresie MAC. Jest adres Broadcast – same jedynki, oznacza nadawanie do wszystkich.

Kabel miedziany umożliwia przesyłanie z częstotliwością ok. 125 MHz. Po zastosowaniu 4B/5B daje to ok. 100 Mb/s, czyli dobrze. W skrętce są 4 pary kabelków, bierzemy sobie parę i wysyłamy komunikację po niej, odbieramy po jakiejś innej. To powoduje, że nie ma problemów z dzieleniem kanału – mamy osobne kanały do nadawania i odbierania, a kabel łączy tylko 2 urządzenia, więc nie ma więcej problemów. Dwie pary kabelków pozostają nieużywane.

W Ethernecie właściwe możliwe są stany $-1, 0, 1$, bo napięcie może być w drugą stronę. W nowych Ethernetach stosuje się stany $-2, -1, 0, 1, 2$ (możemy puścić połowę napięcia). Gdyby puścić takie stany w jedną stronę po wszystkich kabelkach, to mamy około 1Gb/s. Mamy jednostronną komunikację, ale między dwoma urządzeniami – możemy jednocześnie nadawać i czytać, zależnie od tego, jakie jest odchylenie od naszego sygnału możemy stwierdzić, co powinniśmy odczytać. Daje nam do złącze gigabitowe po tych samych kabelkach. Jednocześnie daje nam to pewien rodzaj zabezpieczenia komunikacji – ciężko jest odtworzyć sygnał nie mając pewności, co nadaje jedna ze stron.

Zajęcia 4: Warstwa sieci

2024-10-24

Numer MAC to numer fizyczny karty sieciowej, jest unikatowy, nowy nadawany każdej karcie, choć teoretycznie można zmienić sobie numer na poziomie software'u a systemy operacyjne same nadają adresy MAC wirtualnym kartom sieciowym, więc niekoniecznie są unikatowe.

Broadcast to adres MAC z samymi jedynkami. Służy do wysłania wiadomości do każdego. Karta sieciowa filtruje wiadomości i przekazuje do kernela te adresowane do niej i te broadcastowane. Można zmienić zachowanie karty sieciowej, by nie filtrowała.

Protokół ARP służy do tłumaczenia adresów IP na adresy MAC. Wysyłamy broadcast z pytaniem, czy jakaś maszyna ma dany adres IP, którego szukamy.

Sieci robią się duże, dlatego urządzenia są łączone switchami, które łączą się ze sobą dalej. Switche przekazują komunikaty (nie kopiują sygnału, tylko rozpoznają ramki i nadają je dalej – dzięki temu mogą same decydować o tym, kiedy wyślą wiadomość do każdego odbiorcy i np. robić współdzielenie łącza).

Algorytm switcha: nie odbija wiadomości (nie ma sensu przekazywać wiadomości maszynie, od której się ją dostało), broadcast wysyła wszędzie (poza źródłem). Ramka jest wysyłana w kierunku jej adresata (zapamiętuje, na jakich portach byli obserwowani dani nadawcy, potem wysyła do nich po tych portach) jeśli switch zna ten kierunek, inaczej do wszystkich. Takie ścieżki są pamiętane jakiś czas (rzędu minut), po zapomnieniu uczy się na nowo.

Problem pojawia się, gdy switche zaczną tworzyć cykl. Ramki mają Time To Live – ilość przekazań przez switche, po której mają przestać być nadawane. Jeśli powstanie cykl, to ramka w końcu w nim zniknie. Do tego stosuje się Spanning Tree Protocol – switche uczą się sieci i eliminują cykle. W sieci wybierany jest lider (switch z najmniejszym MAC'iem), który na podstawie informacji o sieci decyduje, które kable nie będą używane.

VLANy – wirtualne sieci, mówimy switchom, po których kabelkach przesyłać daną komunikację, część maszyn traktować jak w jednej sieci, a część jak w drugiej. Połączenie pomiędzy switchami jest uznane za należące do obu sieci. W ramce trzymamy informację o tym, w jakiej sieci została wysłana. Zmieniamy ramkę tak, by było wiadomo, z której sieci idzie wiadomość. W polu z długością komunikatu trzymamy też tryb protokołu, wpisujemy długość dłuższą niż maksymalna możliwa, wtedy wiemy, że dalej będzie numer sieci wirtualnej i dopiero potem długość.

Większość z tych algorytmów działa na poziomie karty sieciowej, ARP w kernelu.

Zajęcia 5: WiFi i sieci komórkowe

2024-10-30

Kanał komunikacji w sieciach WiFi: fale radiowe wysyłane w przestrzeń. W przeciwieństwie do kabli faktycznie występuje współdzielenie kanału komunikacji, do tego nie można stwierdzić, czy ktoś inny odbiera sygnał od kogoś innego – nadawca może być daleko od nas. Dlatego ciężko jest przeciwdziałać kolizjom. Błędy zdarzają się dużo częściej niż w Ethernecie, bo sygnał nie jest po kablu tylko leci przez świat.

Problemy z sieciami WiFi rozwiązuje się poprzez wprowadzenie punktów dostępowych (routery, przekaźniki), które mają sterować siecią. Access pointy łączą się osobnymi kanałami z każdym osobnym klientem, to pomaga w uniknięciu kolizji. Dzięki access pointom można łatwo łączyć sieci WiFi z sieciami Ethernet – można łączyć access pointy ze switchami Ethernetowymi, które mogą być połączone w większe sieci.

Urządzenia w sieciach WiFi są mobilne – regularnie się przemieszczają, zmieniają swoje access pointy. Do tego urządzenia mobilne z reguły mają mały dostęp do energii, więc trzeba ją oszczędzać.

W Ethernecie nie dbaliśmy o bezpieczeństwo, w sieciach komórkowych jest to dużo ważniejsze, bo przesyłany sygnał jest ogólnodostępny i łatwy do przechwycenia.

W sieciach WiFi sygnały słabną z odległością, występują zakłócenia (w szczególności sygnał może sam siebie zakłócić), często nie widać innych uczestników komunikacji – problem ukrytej stacji (nie widzimy, że ktoś inny nadaje do naszego rozmówcy, nie wiemy o zakłóceniu) i eksponowanej stacji (widzimy, że ktoś nadaje, a nasz rozmówca nie, wydaje nam się, że wystąpiło zakłócenie).

Metoda QAM (quadrature amplitude modulation) – przesyłanie informacji poprzez zmianę wysyłanej częstotliwości lub amplitudy (głośności), np. jeśli zobaczymy zakłócenia, to wyciszymy nasz sygnał.

Access pointy w WiFi muszą w jakiś sposób przekazywać informację o tym, że istnieją. Broadcastują taką informację co jakiś czas, dzięki temu nowe urządzenia wiedzą, jakie sieci WiFi są dostępne. Po połączeniu następuje uwierzytelnianie, potem powstaje połączenie kryptograficzne. Access point potrafi rozwiązywać konflikty między urządzeniami, daje im informację, czy wolno im nadawać. Dzięki temu obciążone sieci są w stanie działać. Do tego access pointy przekazują sobie informacje przy zmianie access pointa przez urządzenie.

Błędy w sieciach WiFi są obsługiwane za pomocą kodów CRC (jak w Ethernecie), do tego pojawiają się też kody korygujące LDPC – heurystyka, która potrafi naprawić małe błędy. Potrzebujemy też potwierdzenia, że odbiorca dostał naszą wiadomość, inaczej nie wiemy, czy ramka została uszkodzona.

Aby współdzielić łącze uczestnicy sieci mogą wysłać do access pointa sygnały RTS (request to send), gdy chcą przekazać wiadomość. Wtedy access point może wysłać sygnał CTS (confirmation to send), który znaczy, że teraz będzie słuchał tego uczestnika komunikacji i inni nie powinni im przeszkadzać.

Szyfrowanie kluczem symetrycznym – metoda szyfrowania, kodujemy informację kluczem, potem odbiorca może odkodować informację tym samym kluczem. W tej chwili często stosuje się klucze AES. Szyfrujemy ustaloną liczbę bitów za pomocą klucza o ustalonej długości. Chcemy przesyłać dłuższe wiadomości niż nasze szyfrowanie szyfruje, możemy dzielić na bloki i szyfrować osobno, ale to może ułatwić złamanie szyfru – mamy lepsze metody.

W domowych sieciach WiFi często stosuje się uwierzytelnianie WPA-PSK (pre-shared key) – znamy hasło, na podstawie tego hasła generujemy klucz, którym szyfrujemy. Bierzemy hasło, losowe bity od access pointa i użytkownika, ich adresy MAC, to wszystko hashujemy i dostajemy klucz.

Następuje 4-way handshake:

1. Access point wysyła swoje losowe bity na początku nawiązywania połączenia.
2. Wtedy użytkownik ma już wszystko potrzebne bo szyfrowania, generuje klucz PTK, wysyła swoje losowe bity, a potem wysyła je zahashowane.
3. Wtedy access point może wygenerować klucz PTK i odkodować otrzymane bity. Jeśli się zgadza, to znaczy, że klient podał dobre hasło. Access point wysyła klucz GTK (jeden, znany przez wszystkich użytkowników sieci, wykorzystywany do broadcastowania) zaszyfrowany kluczem PTK.
4. Użytkownik potwierdza nawiązanie połączenia.

Dzięki dodaniu losowych bitów do klucza mamy pewność, że klucz zmieni się przy każdym nawiązaniu połączenia, czyli dość często. Dzięki temu długie zbieranie danych nie pomoże w odszyfrowaniu komunikacji.

Każdy inny użytkownik sieci może podsłuchać wysyłane losowe bity i zna hasło, więc może wygenerować klucz PTK związany z komunikacją innego użytkownika. Dlatego nie można zakładać, że osoby z wnętrza sieci nie widzą komunikacji. W nowszych WiFi używa się WPA3 – lepszy schemat generowania klucza, chroni też przed innymi uczestnikami sieci.

Kiedyś było WPA1 – klucz tylko na podstawie hasła, wtedy bardzo dużo informacji szyfrowane tym samym kluczem (nie zmienia się bardzo długo), daje więcej informacji atakującemu.

Aby WiFi było kompatybilne z Ethernetem stosuje się te same adresy MAC. Do tego ramki muszą być między sobą konwertowalne. Ramki WiFi są dłuższe niż Ethernet (bo muszą przekazać dużo więcej informacji), ale są tak zbudowane, że można po prostu wsadzić ramkę Ethernet w ramkę WiFi. W szczególności ramki WiFi mają 4 pola na adres – adresy nadawcy i odbiorcy faktycznej wiadomości oraz adresy access pointów, które przesyłają sobie wiadomość (w komunikacji radiowej jest to potrzebne).

Aby oszczędzać energię klienta sieci, przez większość czasu wyłącza się kartę sieciową. Aby jednak brać udział w komunikacji karta sieciowa włącza się co jakiś czas. W tym celu przekazuje się informację o tym access pointowi, który buforuje ramki adresowane do danego klienta aż on włączy swoją kartę sieciową i będzie mógł mu je przekazać. Wraz z broadcastem informującym o jego adresie MAC access point wysyła też adresy MAC klientów, dla których ma ramki. Wtedy klient wie, że powinien włączyć kartę sieciową i odebrać te ramki.

Zajęcia 6: Sieć IP

2024-11-07

Rozważamy warstwę internetu, czyli tworzenie sieci, które są bardzo duże. Potrzebujemy w jakiś sposób adresować nasze maszyny, mieć sposób znajdowania tras dla naszych pakietów. Małe sieci, które będziemy łączyć, będą bardzo różne, kolejne metody transportu danych będą się różnić, musimy mieć komunikat, który będzie pasował do wszystkich. Chcielibyśmy też mieć możliwość wysyłania do wielu adresatów, w jakiś sposób związanych ze sobą. Musimy też zajmować się buforowaniem danych.

Na niższych poziomach mamy nadane losowo adresy MAC, pojawiają się też różne dziwne pomysły (np. adresowanie sieci za pomocą klucza publicznego, którego używa kryptografia sieci).

Mamy adresy IP – krótkie (4 bajty), prefix adresu ma sygnalizować lokalizację – adresy o tym samym prefixie powinny znajdować się blisko siebie (geograficznie). Mamy też adresy DNS – adresujemy czytelnym

dla człowieka ciągiem znaków.

Komputery łączące się w sieć są ze sobą bezpośrednio połączone za pomocą różnych technologii. Jeden komputer generuje pakiet IP i adresuje go do innego komputera. Chcemy znaleźć ścieżkę między nimi w sieci, która jest jak najkrótsza i daje najmniejsze opóźnienie. Jednocześnie chcemy to robić szybko. Właściwie sprowadza się to do znalezienia najkrótszej ważonej ścieżki w grafie. Problem jest taki, że sieć cały czas się zmienia, bo np. jedne połączenia są bardziej zajęte niż inne. Do tego komputery mogą łączyć się i odłączać.

Naszą sieć mogą modelować sieci przepływowe, gdzie pakiety płyną ze źródeł do ujść. Przy ustalonych ujściach dla każdego źródła (multi commodity flow) problem jest NP-trudny, więc nie da się tego zrobić optymalnie w sensownym czasie. Podobnie trudne jest stwierdzenie, które pakiety należy w danej chwili przesłać po jakich krawędziach mając zadaną pełną listę komunikatów.

Pomysł na efektywny routing jest taki, że tworzymy hierarchię sieci, dzielimy ją na fragmenty, które odpowiadają prefixom w numerach IP. Znacznie ogranicza to rozmiar problemu – jeśli dzielimy po bajtach, to na najwyższym poziomie mamy 256 wierzchołków, więc nawet trzymanie wszystkich tras działa sensownie. Następnie wewnątrz tych fragmentów znowu dzielimy na fragmenty i powtarzamy cały pomysł.

W tej chwili dzielimy nie na cztery jednostki, tylko na dwie. Mamy Autonomous Systems (AS), które odpowiadają pierwszej połowie adresu IP. Wewnątrz nich rozchodzimy się na faktyczne adresy IP maszyn.

Podczas przesyłania danych pojawia się problem, gdy wiele pakietów chce wykorzystać to samo połączenie. Wtedy nie mamy miejsca, żeby zmieścić wszystkie na raz, musimy buforować pakiety. Dane są przesyłane po równo (wolniej niż przychodzą), reszta jest zapamiętywana, czeka w kolejce do wysłania. Jeśli bufor jest nieograniczony, to wraz ze wzrostem kolejki opóźnienia mogą bardzo rosnąć.

Dlatego ograniczamy bufor, jak przychodzi niemieszczący się pakiet, to możemy wyrzucać ten, który jest najstarszy – wtedy mamy zerową przepustowość, bo nawet jak wyślemy pakiet, to następny router go odrzuci, bo już będzie stary. Możemy wyrzucać losowe pakiety, co powoduje, że połączenia wysyłające więcej są karane (bo mają większą szansę, że ich pakiet zostanie odrzucony). To powoduje, że komputery będą chciały dzielić się połączeniami po równo.