Technologie Sieciowe - lista trzecia

Bartosz Rajczyk

15 kwietnia 2019

1 Wstęp

Sprawozdanie to porusza kwestie techniczne zaprojektowanych programów dotyczących tzw. "bit stuffingu" oraz protokołu CSMA/CD. Oba programy zostały napisane w języku Javascript i są załączone do tego sprawozdania w formie folderu w repozytorium na Githubie.

2 Opis programów

2.1 frame

`frame.js` jest napisanym przeze mnie na potrzeby listy zadań modułem Node.js służącym do realizowania podstawowych założeń bit stuffinu. Składa się on z dwóch funkcji - `encode` oraz `decode`, które służą odpowiednio do przygotowywania danych oraz ich rozszyfrowywania (są one swoimi funkcjami odwrotnymi). Funkcje te przyjmują dwa argumenty, `data` określający wejściowego stringa do zakodowania (wartość domyślna - pusty string) oraz `config` określający sposób kodowania, czylu obiekt (dla pythonowców - słownik) o polach `escSeq` oraz escNum (wartość domyślna - `escSeq: "01111110", escNum: 5`) określających kolejno sekwencję oznaczającą początek i koniec ramki oraz liczbę jedynek, po których zostaje zawsze wstawiane zero. Psuedokod algorytmu przedstawia Algorytm 1.
Jak widać, algorytm ten nie jest skomplikowany, łatwo też sobie wyobrazić algorytm odkodowujący. Jest on na tyle prosty, że możemy zapisać go w formie praktycznie jednej linijki w Javascripcie (za czytelność takiego kodu nie odpowiadamy).

2.2 CSMA

`CSMA.js` jest napisaną przeze mnie klasą służącą do tworzenia obiektów umożliwiających symulowanie protokołu CSMA/CD. Jej konstruktor przyjmuje jeden parametr - `size` - który określa rozmiar symulowanej "szyny danych". Potem na utworzonym obiekcie możemy użyć metody addNode przyjmującej trzy argumenty, `id`, `position` oraz `probability` (wartość domyślna - 0.05) określające kolejno unikalny identyfikator nadajnika, unikalną pozycję nadajnika na

Algorithm 1 Encode

```
1: data ← data + crc(data)
2: result ← escSeq
3: ones ← 0
4: for each char of data do
5:     if char='1' then
6:         ones ← ones + 1
7:     if ones=escNum then
8:         result ← result +′ 0′
9:         ones ← 0
10:    result ← result + char
11: results ← result + escSeq return result
```

szynie oraz prawdopodobieństwo, że podczas dowolnego kroku nadajnik zacznie nadawanie.
Przedstawmy teraz podstawowe założenia tej symulacji:

- najmniejszą jednostką czasu jest `step` wyliczany przy użyciu metody `step` na symulowanym obiekcie
- metoda `step` najpierw oblicza propagację sygnałów na szynie, następnie kolejne wyniki otrzymywane przez nadajniki
- nadajnik posiada kilka parametrów, podstawowym z nich jest `timeout` decydujący o tym, kiedy rozpocznie się kolejne nadawanie albo zakończy aktualne, timeout wyraża liczbę w `stepach`
- kolizja wykrywana jest, kiedy nadajnik podczas swojej emisji wykryje na swojej pozycji więcej niż jeden sygnał
- po wykryciu kolizji emisja wysyłana jest do samego końca, a następnie losowany jest jeden `timeout` z listy składającej się z zera oraz kolejnych potęg dwójki aż do liczby kolizji z rzędu (nazywanych `mult`) pomnożonych przez długość szyny
- `timeout` transmisji wynosi zawsze dwukrotność długości szyny w `stepach`
- po udanej transmisji losowana jest nowa wartość `timeoutu` uzależniona od `probability`

W ten sposób nasza symulacja oddaje całkiem poprawnie założenia protokołu.

3 Eksperymenty

3.1 Frame

Z `frame` nie można było przeprowadzić zbyt wielu eksperymentów. Za to załączona w repozytorium strona internetowa pozwala na dowolne kodowanie i

konfiguracja	próby	sukcesy	kolizje	czekanie
1/4, 3/4	48	30	20	50
0, SIZE - 1	48	30	18	40
i, i+1	48	45	4	160

Tabela 1: odległości

konfiguracja	próby	sukcesy	kolizje	czekanie
3	52	24	34	90
5	64	16	50	211
13	105	4	98	751

Tabela 2: liczba nadajników

odkodowywanie dowolnych ciągów.

3.2 CSMA

Na symulatorze sieci przeprowadziłem szereg eksperymentów. Rozmiarem szyny było 100 oraz wykonywane było 10 000 kroków. Wyniki przedstawiam w odpowiednich tabelkach w zależności od:

- odległości - dwa nadajniki w różnych odległościach
- liczby (nie ilości!) - równomiernie rozmieszczone nadajniki
- częstotliwości nadawania - trzy równomiernie rozmieszczone nadajniki z różnymi p r o b a b i l i t y

4 Wnioski

Z analizy wyników otrzymanych w tabelkach (oraz eksperymentów nieudokomentowanych) można wysnuć następujące hipotezy:

- zwiększanie odległości pomiędzy nadajnikami nie wpływa w znacznym stopniu na ogólna skuteczność takiej sieci, zmniejszenie ich i przybliżenie do siebie za to znacznie poprawia liczbę sukcesów i zmniejsza liczbę kolizji, ale zwiększa wyraźnie czas, który nadajniki spędzają na oczekiwaniu przy zajętej szynie

konfiguracja	próby	sukcesy	kolizje	czekanie
0.01	49	19	31	73
0.1	55	29	25	168
0.5	55	36	13	317

Tabela 3: częstotliwość

- liczba nadajników znajdujących się na szynie znacznie wpływa na jej skuteczność, przy zwiększaniu jej dramatycznie cierpi skuteczność sieci i widać to już przy niewielkich zmianach pokroju pomiędzy trzema a pięcioma nadajnikami
- częstotliwość nadawania wpływa na skuteczność sieci pozytywnie, lecz w prawdziwym środowisku nie można tak precyzyjnie jej kontrolować, bo momenty nadawania są w prawdziwym świecie prawie całkowicie losowe

Przeprowadzony eksperyment rzucił światło na działanie, zalety i wady protokołu CSMA/CD. Najważniejszym wnioskiem jest ten płynący z dużej "gęstości" nadajników w sieci, która to znacząco obniża jej używalność.

5 Dodatkowo

W załączonym repozytorium znajduje się przeglądarkowy symulator małej sieci w protokole CSMA/CD z możliwością dodawania własnych nadajników i oglądania propagacji sygnału krok po kroku. W celu uruchomienia go należy mieć zainstalowanego Node.js, sklonować repozytorium i w katalogu `lista3` wywołać poleceni `npm install`, dzięki któremu zostaną pobrane biblioteki odpowiedzialne za lokalny serwer http oraz obliczanie CRC. Następnie aplikację można uruchomić poleceniem `node main`, a następnie - jeżeli wszystko pójdzie dobrze, jest to bardzo podstawowy serwer - wejść w przeglądarce na wydrukowany w konsoli link.
Pierwsza sekcja strony poświęcona jest kodowaniu i dekodowaniu ramkowanych stringów, a druga jest wspomnianym symulatorem. W konsoli przeglądarki (zwykle CTRL+SHIFT+J) drukowany jest aktualny przebieg symulacji wraz z graficzną reprezentacją jej w tabelce. Sama strona ma bardzo podstawową oprawę graficzną i najpewniej nie zmieni się to znacząco, ponieważ CSS jest bolesny i nie mam dobrych pomysłów na szatę graficzną.