Dokumentacja

Projekt SKJ 2022 – s22666

Model sieci i topologia

Sieć węzłów tworzona jest zgodnie z następującym schematem:

1. Pierwszy węzeł, który nie ma podanej bramy, rozpoczyna nową sieć.
2. Pierwsze cztery węzły, które łączą się z danym węzłem, łączą się również ze sobą
3. Kolejne cztery węzły, które łączą się z danym węzłem, łączą się również ze sobą ale nie z pierwszymi czterema
4. Każdy węzeł obsługuje maksymalnie osiem połączeń
5. Pierwsza „sieć” nazywa się wewnętrzną (nadsiecią), a druga „sieć” nazywa się zewnętrzną (podsiecią)
6. Próba połączenia przy ośmiu istniejących połączeniach powoduje przekierowanie nowego węzła do jednego z węzłów podsieci. (Każde kolejne połączenie jest przesyłane do kolejnego węzła podsieci, tak, żeby sieć rozrastała się równomiernie) Przekierowany węzeł zapomina o oryginalnym węźle „bramy” i łączy się od nowa z podanym w przekierowaniu węzłem

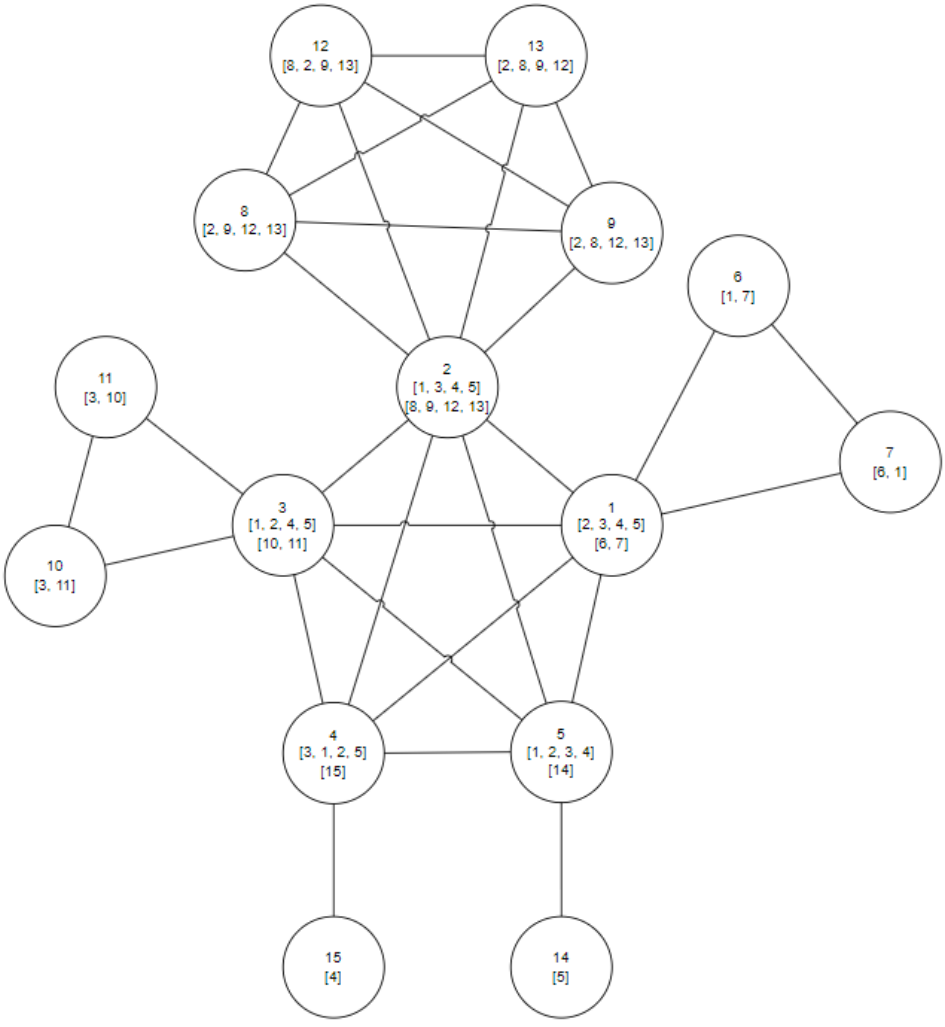


Figure 1: Przykładowa sieć, w nawiasach: połączenia nadsieci, i połączenia podsieci

Algorytm dodawania węzłów

1. Jeżeli tablica wewnętrzna nie jest pełna
    a. Dopisz nowy węzeł do tablicy wewnętrznej
2. Jeżeli tablica zewnętrzna nie jest pełna
    a. Dopisz nowy węzeł do tablicy zewnętrznej
3. W innym wypadku*
    a. Wyślij polecenie przekierowania do kolejnego ze swoich adresów zewnętrznych

*w przypadku dodania węzła przez inny węzeł (patrz punkt niżej) ten przypadek nigdy nie występuje, przekierowania wysyła tylko węzeł obsługujący podłączenie nowego węzła do sieci.

Algorytm przyjmowania połączenia*

1. Wyślij do wszystkich węzłów w swojej tablicy informacje o nowym połączeniu (patrz algorytm dodawania węzłów)
2. Wyślij informacje o węzłach w swojej tablicy do nowo podłączengo węzła
3. Dodaj do swojej tablicy informacje o nowo podłączonym do sieci węźle

*Przyjmowanie połączenia to reakcja na komunikat HONK, a dodawanie węzła to reakcja na komunikat DIR, które są wytłumaczone w dalszej części tego dokumentu.

Każdy węzeł który przy uruchomieniu ma podaną bramę, ma do swojej wewnętrznej tablicy od razu wpisany adres tej bramy.

Model rezerwacji zasobów

Rezerwacja zasobów odbywa się z udziałem węzła, który odbiera i nadaje komunikaty od/do klienta (nazywa się on wtedy Originatorem) oraz innych węzłów, na których są blokowane a potem ewentualnie rezerwowane zasoby.

Algorytm rezerwacji

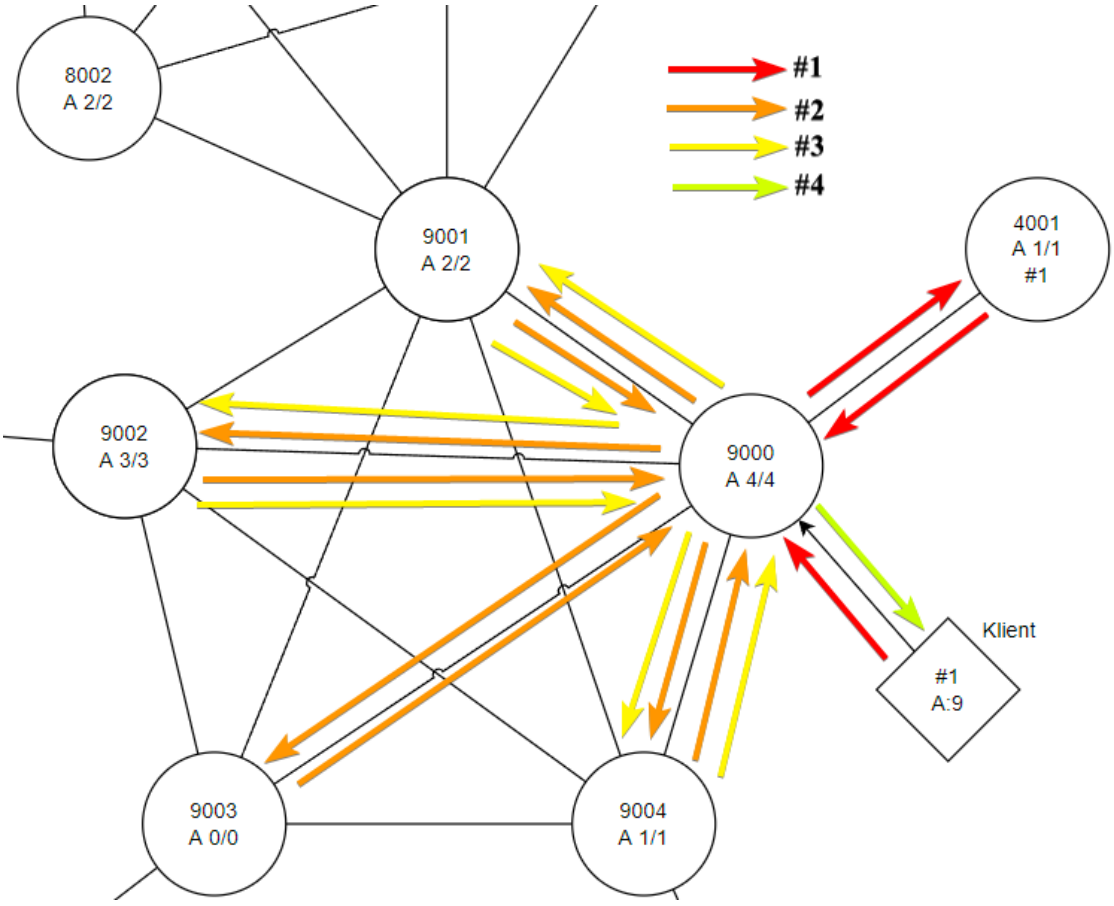
1. Czy mam u siebie wszystkie żądane zasoby?
2. Jeśli tak, od razu je blokuje, następnie wysyłam żądanie rezerwacji do samego siebie, i odpowiadam klientowi
3. Jeśli mam część, lub nie mam żadnych żądanych zasobów lecz jestem połączony z innymi węzłami, to rozpoczynam **wyszukiwanie** i oczekuje na jego rezultat
    a. Po otrzymaniu rezultatu: jeżeli udało się zablokować wszystkie żądane zasoby, i zamówienie jest wypełnione, odpowiadam klientowi zgodnie z miejscem ich alokacji
    b. Po otrzymaniu rezultatu: jeżeli nie udało się zablokować wszystkich żądanych zasobów, to wysyłam klientowi odpowiedź o niepowodzeniu, a następnie zwalniam wszystkie zablokowane zasoby
4. Jeśli nie mam wszystkich żądanych zasobów, i nie jestem połączony z żadnymi węzłami, to odpowiadam klientowi z komunikatem niepowodzenia i odblokowuje u siebie zajęte zasoby (które mogły być zablokowane częściowo)

Algorytm wyszukiwania

1. Jeśli posiadam podsieć (conajmniej jednego członka) to wysyłam do niej żądanie blokady (komunikat LOK), jeśli nie mam podsieci, to wysyłam do swojej nadsieci*
2. Wysyłam żądanie według wzoru: dla każdego zasobu: dzielę ilość potrzebnych (jeszcze nie zablokowanych) zasobów przez ilość członków podsieci (z resztą) następnie oprócz tego przypisuje losowemu z nich resztę z tego dzielenia (nigdy nie wysyłam żądania blokady większej ilości zasobów, niż potrzebuję)
3. Czekam na N komunikatów FIN (o zakończeniu wyszukiwania), po otrzymaniu których jeżeli to było wyszukiwanie w podsieci, przeszukuje podsieć ponownie, z pominięciem węzłów, które nie miały żadnych z żądanych zasobów (nie były w stanie niczego zablokować)
4. Jeżeli to było drugie wyszukiwanie na podsieci, przeszukuje nadsieć, a jeżeli to było przeszukiwanie na nadsieci, to wykonuje analogicznie drugie wyszukiwanie na nadsieci
5. Jeżeli to było ostatnie przeszukiwanie (drugie na nadsieci) to wysyłam odpowiedź klientowi, zgodnie z tym czy udało mi się zablokować wymagane zasoby, czy nie

*węzły nie będące originatorami (którym przekazano LOK z góry) nie prowadzą wyszukiwania na nadsieci, zamiast tego, po drugim przeszukaniu podsieci, odpowiadają węzłowi, który przesłał im LOK, wysyłając mu FIN

Dowolny węzeł, który był w stanie zablokować część potrzebnych zasobów, przekazuje stosowną informację bezpośrednio do originatora, pomijając łańcuch wyszukiwań, dzięki czemu originator od razu wie, jakie zasoby zostały zablokowane.



Komunikaty

Wszystkie komunikaty są prekazywane w formie tekstowej

Wszystkie komunikaty jako separatora między wartościami używają spacji

Komunikaty między węzłami nie muszą używać /n jako zakończenia komunikatu

1. **HONK** – *Trąbi do węzła w celu ogłoszenia swojej obecności, reakcją na ten komunikat jest odpowiednia operacja podłączenia do sieci (patrz Algorytm przyjmowania połączenia)*
2. **ADD** <Adres:Port> [Adres2:Port2 Adres3:Port3 …] – *Informuje dany węzeł o tym, że ma dodać do swoich tablic jakieś adresy (patrz Algorytm dodawania węzłów)*
3. **DIR** <Adres:Port> - *Informuje dany węzęł o tym, że ma rozpocząć łączenie na nowo (wyczyścić tablice i wysłać HONK do adresu wyspecyfikowanego w komunikacie)*
4. **LOK** <AdresOriginatora:PortOriginatora> <ID Zamówienia> <Zasób1:IlośćZas> [<Zasób2:IlośćZas> <Zasób3:IlośćZas> ...] – *Żąda blokady danych zasobów na rzecz* zamówienia ID Zamówienia złożonego Originatorowi, reakcja na ten komunikat odbywa się zgodnie z Algorytmem rezerwacji
5. **LKR** <AdresOriginatora:PortOriginatora> <ID Zamówienia> <Zasób1:IlośćZas> [<Zasób2:IlośćZas> <Zasób3:IlośćZas> ...] – *Informuje orignatora o zajęciu żadanych zasobów lub części żądanych zasobów.*
6. **FIN** <AdresOriginatora:PortOriginatora> <ID Zamówienia> <Zasób1:IlośćZas> [<Zasób2:IlośćZas> <Zasób3:IlośćZas> ...] – *Informuje węzeł który rozpoczął wyszukiwanie o jego zakończeniu, i liczbie zasobów które udało się zająć (może być 0)*
7. **ULK** <Zasób1:IlośćZas> [<Zasób2:IlośćZas> <Zasób3:IlośćZas> ...] – *Każe węzłowi odblokować dane zasoby, reakcją na ten komunikat jest bezwarunkowe odblokowanie zasobów*
8. **RES** <ID Klienta> <Zasób1:IlośćZas> [<Zasób2:IlośćZas> <Zasób3:IlośćZas> ...] – *Każe węzłowi bezwarunkowo zarezerwować zasoby które zostały zablokowane. Teoretycznie jest możliwa odmowa rezerwacji, jeśli blokada była niepoprawnie wysłana lub nie doszła, na takie wydarzenie jednak, węzeł nie odpowiada. Po prostu nie rezerwuje żądanych zasobów.*

Dokumentacja kodu

Dostępna jest w pliku **docs/index.html** w formacie javadoc.

Założenia, ograniczenia i warunki

1. System nie sprawdza poprawności otrzymanych komunikatów ani argumentów wiersza poleceń i zakłada że komunikat zawsze ułożony jest zgodnie z formatem, tak jak i są podane odpowiednie argumenty wiersza poleceń

2. System zakłada że nie występuje utrata pakietów i że **każdy pakiet wysłany, to pakiet dostarczony**. W przypadku utraty, błędu transmisji, itd. system może w nieskończoność oczekiwać na odpowiedź lub poinformować klienta o rezerwacji zasobów, które tak naprawdę nie zostały zarezerwowane. Mogą występować również różne inne błędy.
   **Aczkolwiek implementacja systemu potwierdzeń i przedawnień jest możliwa i zgodna z algorytmami, jedynie nie starczyło na to czasu.*

3. Jeśli klient potrzebuje zasobu, który posiada tylko węzeł, który został dodany w trakcie przetwarzania zapytania przez sieć, to w zależności od etapu na którym znajdowało się wyszukiwanie w momencie dodania tego węzła, odpowiedź na zapytanie może być niezgodna z rzeczywistym stanem sieci (były zasoby, a został odesłany komunikat o niepowodzeniu)

4. Sieć nie była testowana na zapytaniach pochodzących z adresów IPv6, ani na węzłach korzystających z adresów IPv6. W teorii powinno to działać, aczkolwiek ze względu na użycie w paru miejscach „:” jako separatora, jest szansa, że będzie to powodowało problemy

5. Program korzysta z biblioteki https://github.com/vaqxai/java-tcpclientserver. Jest ona napisana przeze mnie (co mogę udowodnić), a jej składowe są zawarte w projekcie jako bezpośrednio skopiowane z biblioteki (ze względu na ograniczenie założeń projektu nie pozwalających na użycie narzędzia Maven)

S22666 (aka Vaqxai), Stanisław Knapiński, 2022-01-23