Zadanie 1

Otwarto: piątek, 28 marca 2025, 00:00
Wymagane do: wtorek, 29 kwietnia 2025, 23:59

Sieciowe synchronizowanie zegarów

Zaimplementuj program do synchronizowania zegarów w modelu *peer-to-peer*. Sieć synchronizowania zegarów składa się z równoprawnych węzłów. Każdy węzeł komunikuje się z innymi węzłami, aby zsynchronizować swój zegar, uwzględniając czas podróży pakietów w sieci.

Działanie sieci synchronizowania zegarów obejmuje następujące czynności:

- dołączenie węzła do sieci poprzez kontakt z innym węzłem,
- wybranie lidera, do którego węzły synchronizują swoje zegary,
- synchronizowanie zegarów uwzględniające opóźnienia transmisji pakietów.

Każdy węzeł utrzymuje swój *naturalny zegar*, który jest liczbą milisekund od jego uruchomienia.

Poziom synchronizacji węzła to wartość `255`, gdy węzeł nie jest zsynchronizowany z żadnym węzłem, wartość `0`, gdy węzeł jest źródłem synchronizacji dla innych węzłów (jest liderem), wartość `1`, gdy węzeł jest zsynchronizowany z liderem, wartość `2`, gdy węzeł jest zsynchronizowany z węzłem, który jest zsynchronizowany z liderem itd., ale maksymalnie `254`.

Każdy węzeł przechowuje swój poziom synchronizacji. Wartość początkowa poziomu synchronizacji węzła to `255`. Węzeł pamięta, z którym węzłem jest zsynchronizowany.

Węzły komunikują się za pomocą UDP i adresowania IPv4.

Parametry programu węzła

Program implementujący funkcjonalność węzła akceptuje w linii poleceń następujące parametry:

- `-b bind_address` – adres IP, na którym węzeł nasłuchuje, opcjonalny, domyślnie węzeł nasłuchuje na wszystkich adresach hosta, na którym jest uruchomiony;
- `-p port` – port, na którym węzeł nasłuchuje, liczba z przedziału od 0 do 65535, zero oznacza wybór dowolnego portu, opcjonalny, domyślnie zero;
- `-a peer_address` – adres IP lub nazwa hosta innego węzła, z którym ten węzeł powinien się skomunikować, opcjonalny;
- `-r peer_port` – port innego węzła, z którym ten węzeł powinien się skomunikować, liczba z przedziału od 1 do 65535, opcjonalny.

Parametry mogą być podawane w dowolnej kolejności. Parametry `-a` i `-r` muszą być podane oba. Zachowanie programu, gdy któryś z parametrów został podany wielokrotnie, powinno być rozsądne.

Komunikaty wymieniane między węzłami

?

W komunikatach wymienianych między węzłami mogą wystąpić następujące pola:

- `message` – 1 oktet, typ komunikatu;
- `count` – 2 oktety, liczba znanych węzłów;
- `peer_address_length` – 1 oktet, liczba oktetów w polu `peer_address`;
- `peer_address` – `peer_address_length` oktetów, adres IP węzła w formacie jak w nagłówku IP;
- `peer_port` – 2 oktety, numer portu, na którym nasłuchuje węzeł;
- `timestamp` – 8 oktetów, czas, wartość zegara;
- `synchronized` – 1 oktet, poziom synchronizacji węzła.

Wartości w binarnych polach wielooktetowych zapisuje się w porządku sieciowym.

Węzły wysyłają komunikaty z portu, na którym nasłuchują. Węzły uzyskują informację o adresie IP i numerze portu nadawcy z warstwy sieciowej. Adres IP i numer portu jednoznacznie identyfikują węzeł.

Dołączanie do sieci

Węzeł może dołączyć do sieci na dwa sposoby:

- jeśli został uruchomiony bez parametrów `-a` i `-r`, nasłuchuje komunikatów i oczekuje na nowego uczestnika;
- jeśli został uruchomiony z parametrami `-a` i `-r`, wysyła komunikat `HELLO` do podanego węzła.

Komunikaty wymieniane w tym etapie komunikacji:

- `HELLO` – `message = 1`;
- `HELLO_REPLY` – `message = 2`, `count` – liczba rekordów zawierających informacje o węzłach, które zna odpowiadający węzeł, po czym następują te rekordy, każdy zawiera `peer_address_length`, `peer_address`, `peer_port`;
- `CONNECT` – `message = 3`;
- `ACK_CONNECT` – `message = 4`.

Komunikat `HELLO` informuje inny węzeł o chęci nawiązania z nim komunikacji.

Komunikat `HELLO_REPLY` jest odpowiedzą na komunikat `HELLO`. Informuje nowy węzeł w sieci o innych aktywnych węzłach w sieci. W przesyłanej liście nie ma nadawcy ani odbiorcy tego komunikatu. Nowy węzeł na podstawie tej listy wysyła komunikat `CONNECT` do każdego poznanego węzła w celu nawiązania komunikacji z tym węzłem.

Komunikat `CONNECT` informuje o chęci nawiązania komunikacji z innym węzłem. Węzeł, który odebrał komunikat `CONNECT`, dodaje do swojej listy węzłów węzeł, od którego dostał ten komunikat.

Komunikat `ACK_CONNECT` jest odpowiedzią na komunikat `CONNECT`. Potwierdza nawiązanie komunikacji. Węzeł dodaje do swojej listy węzłów węzły, które potwierdziły nawiązanie komunikacji.

Węzły, które wymieniły komunikaty `HELLO` i `HELLO_REPLY` nawiązały komunikację i nie wymieniają już komunikatów `CONNECT` i `ACK_CONNECT`.

Synchronizowanie czasu

Proces synchronizowania zegarów węzłów obejmuje wymianę trzech komunikatów:

- `SYNC_START` – `message = 11`, `synchronized`, `timestamp`;
- `DELAY_REQUEST` – `message = 12`;
- `DELAY_RESPONSE` – `message = 13`, `synchronized`, `timestamp`.

?

Komunikat `SYNC_START` jest wysyłany cyklicznie przez wszystkie węzły, których poziom synchronizacji jest mniejszy niż `254`, do wszystkich im znanych węzłów. Komunikat ten zawiera poziom synchronizacji i aktualną wartość zegara `T1` węzła, który jest nadawcą tego komunikatu.

Węzeł podejmuje próbę synchronizacji i odpowiada na `SYNC_START` tylko wtedy, gdy:

- nadawca jest mu znany;
- nadawca ma poziom synchronizacji mniejszy niż `254`;
- poziom synchronizacji nadawcy jest mniejszy od jego poziomu synchronizacji dla nadawcy będącego węzłem, z którym ten węzeł jest zsynchronizowany;
- poziom synchronizacji nadawcy jest o co najmniej dwa mniejszy od jego poziomu synchronizacji dla nadawcy będącego węzłem, z którym ten węzeł nie jest zsynchronizowany.

Węzeł kontynuuje proces synchronizowania tylko z pierwszym węzłem, który spełnia wyżej opisane warunki.

Jeśli przez 20–30 sekund węzeł nie dostanie komunikatu `SYNC_START` od węzła, z którym jest zsynchronizowany, lub dostanie od niego taki komunikat z wartością pola `synchronized` większą od lub równą wartości własnego poziomu synchronizacji, zmienia swój poziom synchronizacji na `255`.

Jeśli węzeł odebrał komunikat `SYNC_START` i kontynuuje proces synchronizowania, to zapisuje wartość swojego zegara `T2` w chwili odebrania tego komunikatu, odsyła komunikat `DELAY_REQUEST` i zapisuje wartość swojego zegara `T3` w chwili jego wysłania.

Węzeł, który rozpoczął proces synchronizowania, po odebraniu komunikatu `DELAY_REQUEST` odsyła komunikat `DELAY_RESPONSE` zawierający ponownie jego poziom synchronizacji i wartość zegara `T4` w chwili otrzymania komunikatu `DELAY_REQUEST`.

Węzeł synchronizowany oblicza

```
offset = (T2 - T1 + T3 - T4) / 2
```

Od tego momentu węzeł synchronizowany jest zsynchronizowany na poziome o jeden większym niż wartość `synchronized` odebrana w komunikatach `SYNC_START` i `DELAY_RESPONSE` (wartości tego pola muszą być identyczne w obu komunikatach).

Węzły powinny wysyłać komunikat `SYNC_START` do wszystkich znanych im węzłów co 5 do 10 sekund. Jeśli węzeł nie otrzyma odpowiedzi `DELAY_REQUEST` lub `DELAY_RESPONSE` w czasie 5 do 10 sekund, to zaprzestaje procedury synchronizowania i ignoruje takie komunikaty odebrane po tym czasie.

Jeśli węzeł wysyłający wartości `T1` i `T4` ma zegar zsynchronizowany z innym węzłem, to wysyła wartości tego zsynchronizowanego czasu.

Wybieranie lidera

Aby rozpoczął się proces synchronizowania węzłów, co najmniej jeden węzeł musi zostać liderem. Służy do tego komunikat:

- `LEADER` – `message = 21`, `synchronized`.

Jeśli wartość pola `synchronized` w komunikacie `LEADER` wynosi `0`, to węzeł staje się liderem (ustawia swój poziom synchronizacji na `0`) i po dwóch sekundach od otrzymania tego komunikatu zaczyna wysyłać komunikaty synchronizujące do innych węzłów, które zna.

Jeśli wartość pola `synchronized` w komunikacie `LEADER` wynosi `255` i węzeł jest

?

liderem, to węzeł ten przestaje być liderem (ustawia swój poziom synchronizacji na `255`) i przestaje wysyłać komunikaty synchronizujące do innych węzłów, przy czym kończy już rozpoczęte wymiany komunikatów synchronizujących.

Inne wartości w polu `synchronized` tego komunikatu są niepoprawne. Niepoprawny jest też komunikat z wartością `255` w polu `synchronized` odebrany przez węzeł niebędący liderem.

Węzeł powinien reagować na każdy komunikat `LEADER` od początku działania bez weryfikowania jego nadawcy.

Udzielanie informacji o aktualnym czasie

Każdy węzeł powinien udzielać informacji o aktualnym czasie, korzystając z komunikatów:

- `GET_TIME` – `message = 31`;
- `TIME` – `message = 32`, `synchronized`, `timestamp`.

Komunikat `GET_TIME` jest żądaniem podania aktualnego czasu.

Komunikat `TIME` jest odpowiedzią na komunikat `GET_TIME` i zawiera poziom synchronizacji i wartość naturalnego zegara odpowiadającego węzła, jeśli węzeł nie jest zsynchronizowany z innym węzłem, a wartość tego zegara ~~skorygowaną~~pomniejszoną o wartość `offset`, jeśli węzeł jest zsynchronizowany z innym węzłem.

Węzeł powinien odpowiadać na każdy komunikat `GET_TIME` od początku działania bez weryfikowania jego nadawcy.

Obsługa błędów

Program powinien dokładnie sprawdzać poprawność parametrów, a dokładną informację o błędzie powinien wypisywać na standardowe wyjście diagnostyczne, kończąc działanie z kodem `1`.

Błędy związane z wywoływaniem funkcji systemowych lub bibliotecznych należy obsługiwać, wypisując na standardowe wyjście diagnostyczne stosowną informację. Jeśli błąd uniemożliwia dalsze działanie programu, należy program zakończyć kodem `1`. Błędy umożliwiające dalsze działanie programu, np. niepoprawne komunikaty, ignorowane komunikaty, nie powinny przerywać jego działania. Komunikat należy uznać za niepoprawny również wtedy, gdy nadawcą jest nieznany węzeł (patrz wyjątki opisane wyżej) albo jest nieoczekiwany w danym stanie komunikacji, albo gdy zawiera niepoprawną lub nieoczekiwaną wartość jakiegoś pola.

Informacje o błędach wypisywane na standardowe wyjście diagnostyczne powinny zaczynać się od słowa `ERROR`. Informacje o niepoprawnych komunikatach powinny zaczynać się od `ERROR MSG`, po czy występuje spacja i co najwyżej 10 początkowych bajtów tego komunikatu zapisanych szesnastkowo, przykładowo:

```
ERROR MSG 7a12c534
```

Dodatkowe wyjaśnienia

1. Rozmiar pola `count` w komunikacie `HELLO_REPLY` ogranicza liczbę węzłów do 65535. Należy przyjąć, że jest to górne ograniczenie i należy ignorować komunikaty, które powodowałyby przekroczenie tego ograniczenia. Jeśli komunikat `HELLO_REPLY` jest zbyt duży, aby mógł być wysłany, należy go porzucić i potraktować jak inne ignorowane komunikaty.
2. Komunikat `HELLO_REPLY` należy uznać w całości za błędny, jeśli nie zawiera `count` rekordów, jeśli któreś z pól `peer_address_length` lub `peer_port` ma błędną wartość, jeśli w liście węzłów jest nadawca lub odbiorca tego komunikatu.

?

Rozwiązanie

Rozwiązanie należy zaimplementować w języku C lub C++, korzystając z interfejsu gniazd. Nie wolno korzystać z żadnych innych bibliotek realizujących komunikację sieciową. Oczekujemy rozwiązania jednowątkowego, mimo to komunikacja z jednym węzłem nie powinna blokować komunikacji z innymi węzłami. Program ma się kompilować i uruchamiać w laboratorium komputerowym.

Programy powinny być napisane zgodnie ze sztuką. Brak oczywistych oczekiwań wobec programów (np. że nie sformatują dysku, czy że wartości zwracane przez funkcje systemowe są sprawdzane) w treści zadania nie oznacza, że program, który ich nie spełnia, będzie uznany za dobry. Również kod programu będzie podlegał ocenie. Protokół będzie testowany rygorystycznie.

Jako rozwiązanie należy przysłać archiwum zawierające pliki niezbędne do zbudowania rozwiązania. Nie wolno załączać plików binarnych i innych zbędnych. Do stworzenia archiwum należy użyć programu `zip`, `rar`, `7z` lub pary programów `tar` i `gzip`. Archiwum powinno mieć odpowiednio rozszerzenie `.zip`, `.rar`, `.7z` lub `.tgz`. Po rozpakowaniu wszystkie pliki powinny znajdować się w katalogu, w którym jest plik archiwum. Archiwum nie może zawierać podkatalogów. Archiwum powinno zawierać plik `makefile` lub `Makefile`. Wykonanie polecenia `make` powinno stworzyć plik wykonywalny `peer-time-sync`. Wykonanie polecenia `make clean` powinno usunąć wszystkie pliki powstałe podczas kompilowania.

opis-testow.md 24 maja 2025, 18:03