Algorytmy plotkujące

Marek Dzikiewicz

Uniwersytet Warszawski
Wydział Matematyki, Informatyki i Mechaniki
Seminarium Systemów Rozproszonych

22 stycznia 2009

Plan prezentacji

1. Wstęp
2. Przynależność do systemu
3. Świadomość topologii sieciowej
4. Zarządzanie buforem danych
5. Filtrowanie komunikatów
6. Inne problemy

Co to są algorytmy plotkujące?

- Algorytmy plotkujące (ang. *epidemic algorithms* lub *gossip algorithms*) naśladują zachowanie chorób zakaźnych.

Co to są algorytmy plotkujące?

- Algorytmy plotkujące (ang. *epidemic algorithms* lub *gossip algorithms*) naśladują zachowanie chorób zakaźnych.
- Zapewniają efektywny sposób wymiany informacji w dużych, zdecentralizowanych sieciach (np. peer-to-peer).

Jak działają algorytmy plotkujące?

- System rozproszony składa się ze zbioru komunikujących się ze sobą procesów.

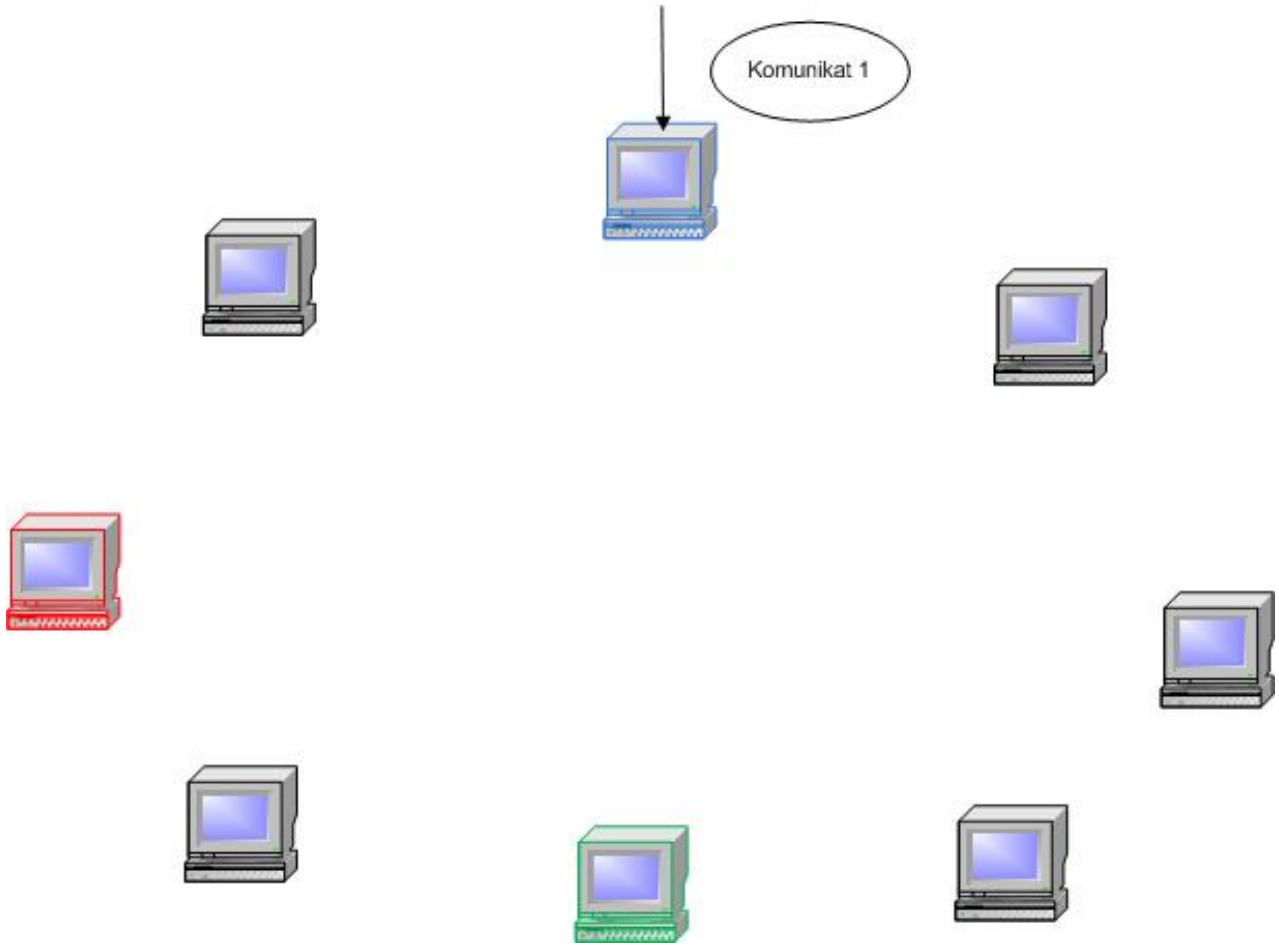
Jak działają algorytmy plotkujące?

- System rozproszony składa się ze zbioru komunikujących się ze sobą procesów.
- Każdy proces systemu rozproszonego przekazuje informacje do losowych procesów, z którymi ma kontakt.

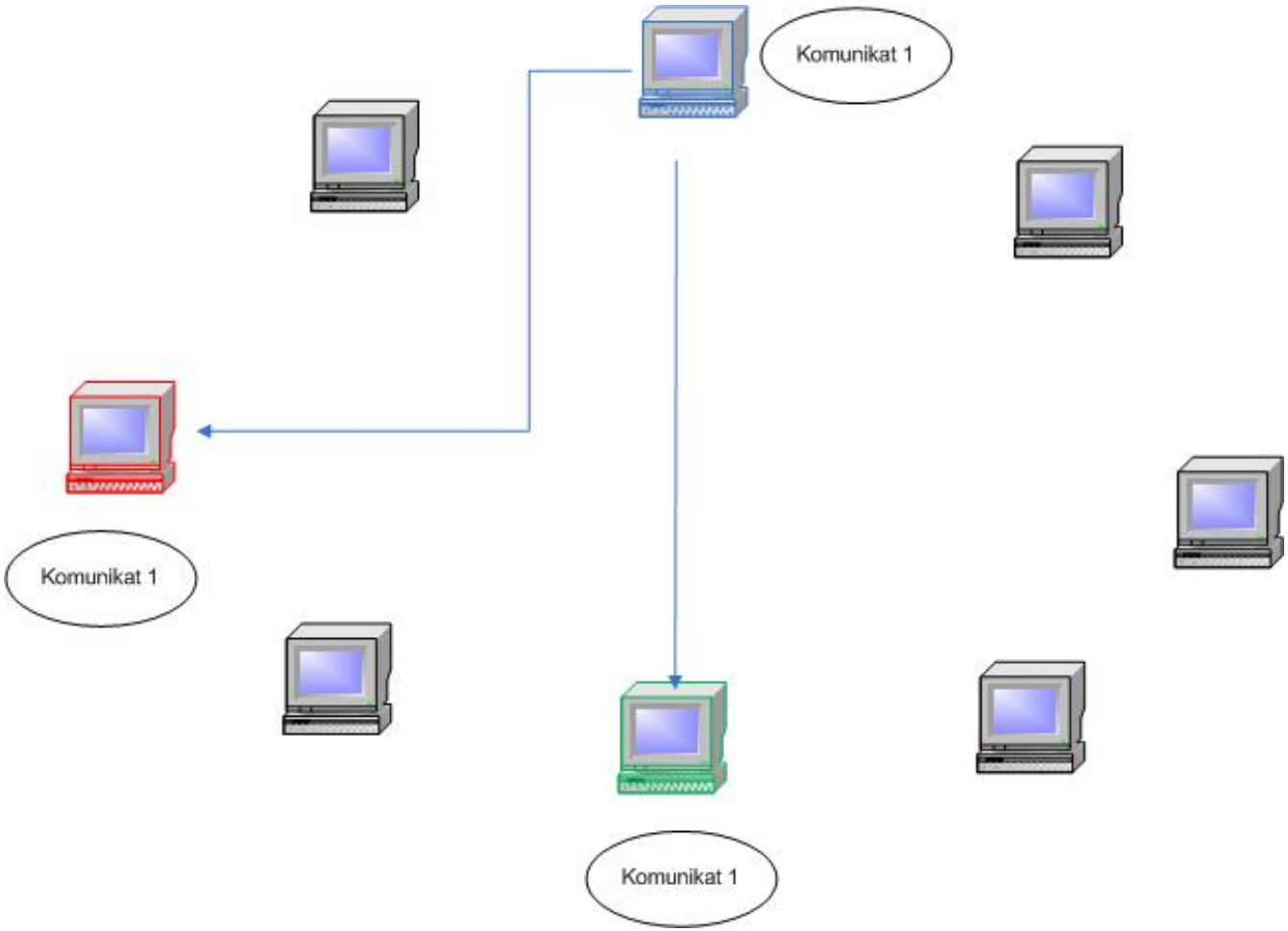
Jak działają algorytmy plotkujące?

- System rozproszony składa się ze zbioru komunikujących się ze sobą procesów.
- Każdy proces systemu rozproszonego przekazuje informacje do losowych procesów, z którymi ma kontakt.
- Każdy proces przechowuje (przez pewien czas) otrzymane komunikaty w buforze, oraz przekazuje je innym procesom.

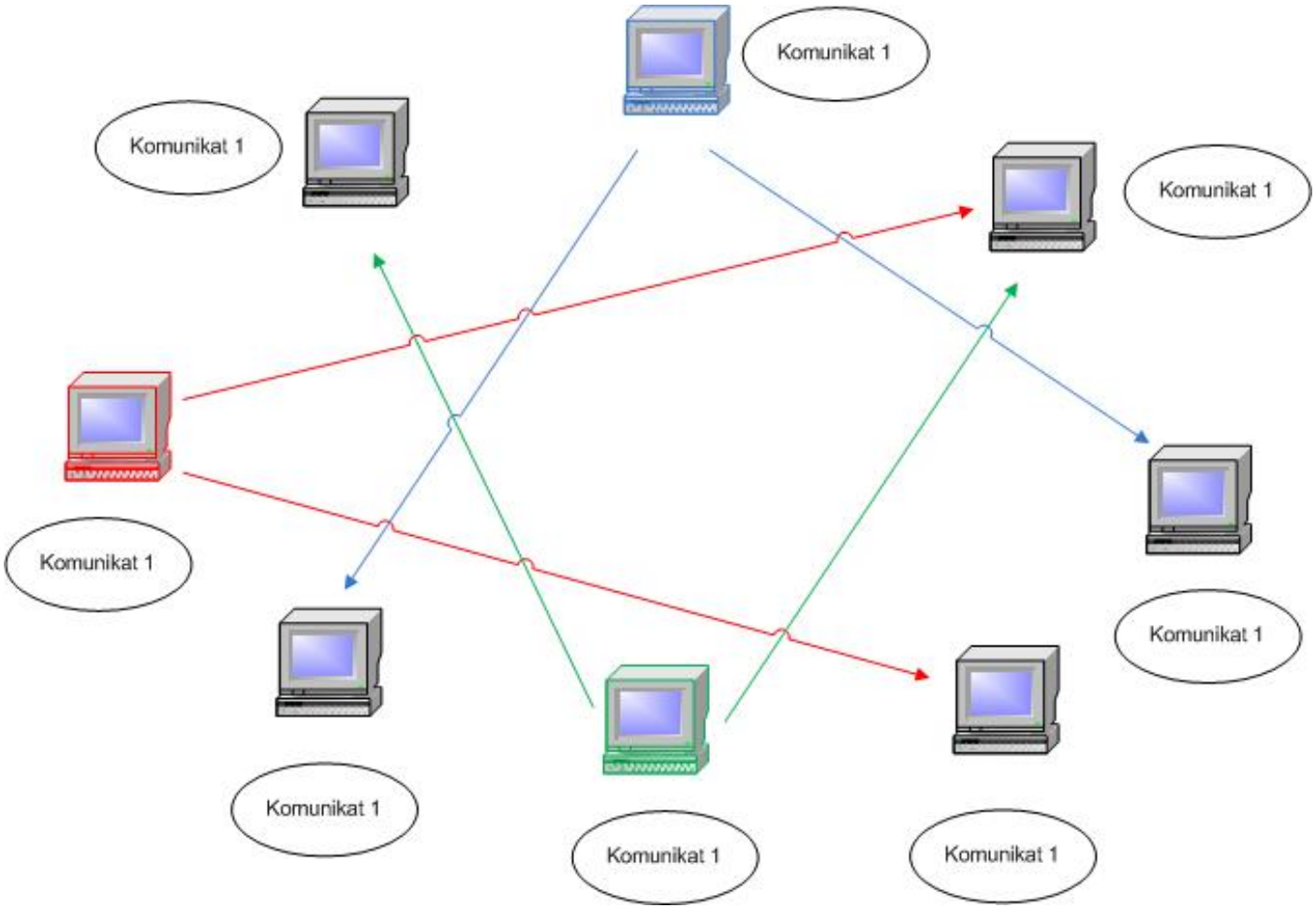
Jak działają algorytmy plotkujące?



Jak działają algorytmy plotkujące?



Jak działają algorytmy plotkujące?



Parametry rozprzestrzeniania danych

- n - liczba procesów w systemie

Parametry rozprzestrzeniania danych

- n - liczba procesów w systemie
- b - pojemność bufora danych

Parametry rozprzestrzeniania danych

- n - liczba procesów w systemie
- b - pojemność bufora danych
- t - ile razy każdy komunikat jest przesyłany do innych procesów

Parametry rozprzestrzeniania danych

- n - liczba procesów w systemie
- b - pojemność bufora danych
- t - ile razy każdy komunikat jest przesyłany do innych procesów
- f - do ilu procesów jest przesyłany za każdym razem komunikat, (tzw. *fan-out*)

Zalety algorytmów plotkujących

- Są bardzo skuteczne

Zalety algorytmów plotkujących

- Są bardzo skuteczne
- Są elastyczne i skalowalne (niezawodność a zużywane zasoby)

Zalety algorytmów plotkujących

- Są bardzo skuteczne
- Są elastyczne i skalowalne (niezawodność a zużywane zasoby)
- Są odporne na awarie (brak *single point of failure*)

Zalety algorytmów plotkujących

- Są bardzo skuteczne
- Są elastyczne i skalowalne (niezawodność a zużywane zasoby)
- Są odporne na awarie (brak *single point of failure*)
- Są odporne na częste przyłączanie i odłączanie procesów (*churn*)

Zalety algorytmów plotkujących

- Są bardzo skuteczne
- Są elastyczne i skalowalne (niezawodność a zużywane zasoby)
- Są odporne na awarie (brak *single point of failure*)
- Są odporne na częste przyłączanie i odłączanie procesów (*churn*)
- Są proste w implementacji

Zastosowania algorytmów plotkujących

- Replikacja baz danych

Zastosowania algorytmów plotkujących

- Replikacja baz danych
- Wykrywanie/monitorowanie zasobów

Zastosowania algorytmów plotkujących

- Replikacja baz danych
- Wykrywanie/monitorowanie zasobów
- Wykrywanie awarii

Zastosowania algorytmów plotkujących

- Replikacja baz danych
- Wykrywanie/monitorowanie zasobów
- Wykrywanie awarii
- Agregacja danych

Zastosowania algorytmów plotkujących

- Replikacja baz danych
- Wykrywanie/monitorowanie zasobów
- Wykrywanie awarii
- Agregacja danych
- Równoważenie obciążenia

Zastosowania algorytmów plotkujących

- Replikacja baz danych
- Wykrywanie/monitorowanie zasobów
- Wykrywanie awarii
- Agregacja danych
- Równoważenie obciążenia
- Ostatnio także w ogólnych systemach rozproszonych opartych o sieć Internet (np. Tribler)

Zastosowania algorytmów plotkujących

- Replikacja baz danych
- Wykrywanie/monitorowanie zasobów
- Wykrywanie awarii
- Agregacja danych
- Równoważenie obciążenia
- Ostatnio także w ogólnych systemach rozproszonych opartych o sieć Internet (np. Tribler)
- Sieci bezprzewodowe

Podstawowe problemy

1. Przynależność do systemu (*membership*)
2. Świadomość topologii sieciowej (*network awareness*)
3. Zarządzanie buforem danych (*buffer management*)
4. Filtrowanie komunikatów (*message filtering*)

Plan prezentacji

1 Wstęp
2 Przynależność do systemu
3 Świadomość topologii sieciowej
4 Zarządzanie buforem danych
5 Filtrowanie komunikatów
6 Inne problemy

Przynależność

Skąd procesy wiedzą o innych procesach należących do systemu?

Rozwiązanie 1

Każdy proces zna wszystkie inne procesy w systemie.

Rozwiązanie 1

Każdy proces zna wszystkie inne procesy w systemie.

- Jak odnaleźć wszystkie procesy w systemie?

Rozwiązanie 1

Każdy proces zna wszystkie inne procesy w systemie.

- Jak odnaleźć wszystkie procesy w systemie?
- Koszt pamięci wzrasta liniowo wraz z wzrostem wielkości systemu.

Rozwiązanie 1

Każdy proces zna wszystkie inne procesy w systemie.

- Jak odnaleźć wszystkie procesy w systemie?
- Koszt pamięci wzrasta liniowo wraz z wzrostem wielkości systemu.
- Duży narzut komunikacyjny w sieci w czasie uaktualniania informacji.

Rozwiązanie 2

Każdy proces zna tylko pewną część procesów systemu.

Możliwe realizacje

Protokół, który okresowo wymienia informacje o aktywnych procesach.

Możliwe realizacje

Protokół, który okresowo wymienia informacje o aktywnych procesach.

- Zapewnia dużą elastyczność

Możliwe realizacje

Protokół, który okresowo wymienia informacje o aktywnych procesach.

- Zapewnia dużą elastyczność
- Zwiększa obciążenie sieci

Możliwe realizacje

Protokół, który okresowo wymienia informacje o aktywnych procesach.

- Zapewnia dużą elastyczność
- Zwiększa obciążenie sieci
- Wymaga implementacji protokołu

Możliwe realizacje

Przekazywanie informacji o procesach razem z rozprzestrzenianą informacją.

Możliwe realizacje

Przekazywanie informacji o procesach razem z rozprzestrzenianą informacją.

- Dynamiczna wymiana najświeższych informacji o procesach

Możliwe realizacje

Przekazywanie informacji o procesach razem z rozprzestrzenianą informacją.

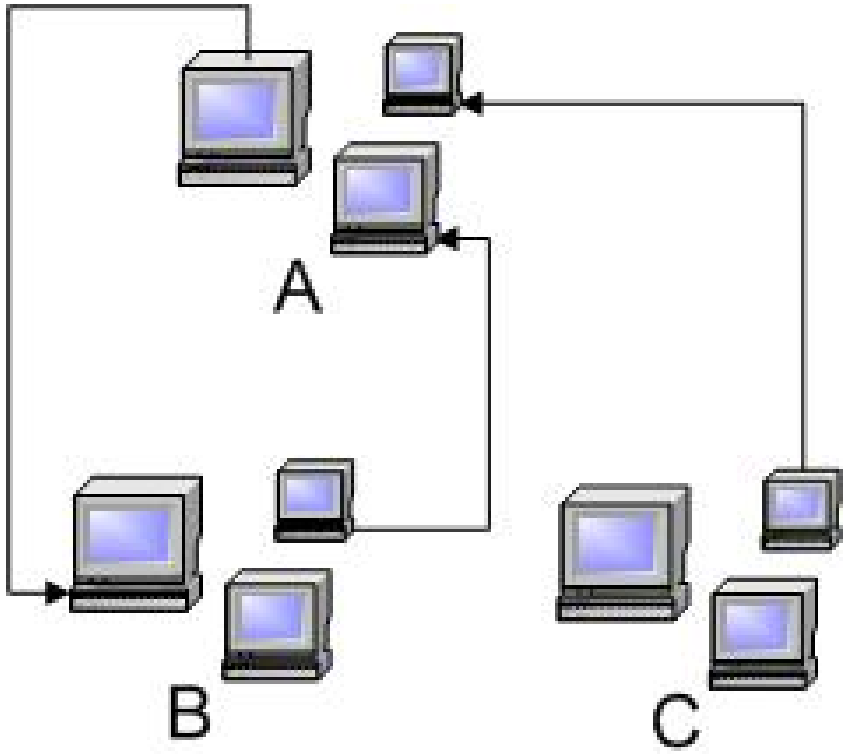
- Dynamiczna wymiana najświeższych informacji o procesach
- Niski narzut komunikacyjny

Możliwe realizacje

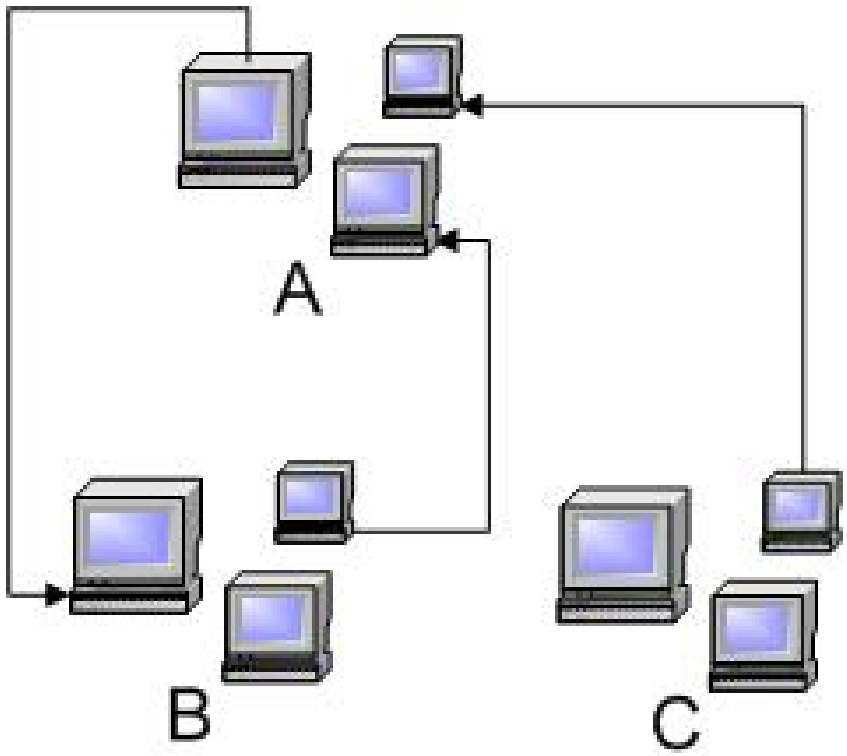
Przekazywanie informacji o procesach razem z rozprzestrzenianą informacją.

- Dynamiczna wymiana najświeższych informacji o procesach
- Niski narzut komunikacyjny
- Prosta implementacja

Problem spójności



Problem spójności



Komunikaty od procesów z grupy A oraz B nie dochodzą do procesów z grupy C!

Problem skalowalności

Wielkość okna znanych procesów (l) musi być zależna od wielkości systemu żeby zapewnić skalowalność.

Problem skalowalności

Wielkość okna znanych procesów (l) musi być zależna od wielkości systemu żeby zapewnić skalowalność.

- l oraz f rośnie wraz z n

Problem skalowalności

Wielkość okna znanych procesów (l) musi być zależna od wielkości systemu żeby zapewnić skalowalność.

- l oraz f rośnie wraz z n
- l oraz t rośnie wraz z n

Problem skalowalności

Wielkość okna znanych procesów (l) musi być zależna od wielkości systemu żeby zapewnić skalowalność.

- l oraz f rośnie wraz z n
- l oraz t rośnie wraz z n

Ale jak poznać n?

Problem podłączania do systemu

Jak proces podłącza się do systemu?

Problem podłączania do systemu

Jak proces podłącza się do systemu?

Potrzeba jakiegoś zewnętrznego mechanizmu, który spowoduje że proces pozna adresy innych procesów w systemie.

Zwykle nazywa się go z ang. *bootstrap*.

Co to są sieci Ad Hoc

- Sieci Ad Hoc to sieci, które zmieniają się bardzo dynamicznie (urządzenia są często przyłączane i odłączane).

Co to są sieci Ad Hoc

- Sieci Ad Hoc to sieci, które zmieniają się bardzo dynamicznie (urządzenia są często przyłączane i odłączane).
- Urządzenia wchodzące w skład sieci są dostępne tylko na czas sesji lub jeśli są w określonej odległości od siebie.

Co to są sieci Ad Hoc

- Sieci Ad Hoc to sieci, które zmieniają się bardzo dynamicznie (urządzenia są często przyłączane i odłączane).
- Urządzenia wchodzące w skład sieci są dostępne tylko na czas sesji lub jeśli są w określonej odległości od siebie.
- Dobrym przykładem sieci Ad Hoc są sieci oparte o połączenia bezprzewodowe.

Sieci MANET

- Samokonfigurująca się, zdecentralizowana sieć oparta o połączenia bezprzewodowe.

Sieci MANET

- Samokonfigurująca się, zdecentralizowana sieć oparta o połączenia bezprzewodowe.
- Przyłączone urządzenia mobilne mogą pełnić zarówno funkcję klienta jak i routera.

Sieci MANET

- Samokonfigurująca się, zdecentralizowana sieć oparta o połączenia bezprzewodowe.
- Przyłączone urządzenia mobilne mogą pełnić zarówno funkcję klienta jak i routera.
- Brak infrastruktury sieciowej.

Sieci MANET

- Samokonfigurująca się, zdecentralizowana sieć oparta o połączenia bezprzewodowe.
- Przyłączone urządzenia mobilne mogą pełnić zarówno funkcję klienta jak i routera.
- Brak infrastruktury sieciowej.
- Brak punktów zarządzających siecią.

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Urządzenia w sieciach bezprzewodowych mogą się komunikować ze sobą tylko pod warunkiem, że znajdują się w określonej odległości od siebie.

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Urządzenia w sieciach bezprzewodowych mogą się komunikować ze sobą tylko pod warunkiem, że znajdują się w określonej odległości od siebie.
- Komunikacja między procesami może być możliwa tylko przy pomocy procesów pośrednich.

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Algorytmy plotkujące bardzo dobrze się nadają do rozprzestrzeniania informacji w systemach Ad Hoc.

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Algorytmy plotkujące bardzo dobrze się nadają do rozprzestrzeniania informacji w systemach Ad Hoc.
- Procesy mogą przechowywać i uaktualniać strukturę danych zawierającą procesy, z którymi jest możliwa komunikacja.

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Algorytmy plotkujące bardzo dobrze się nadają do rozprzestrzeniania informacji w systemach Ad Hoc.
- Procesy mogą przechowywać i uaktualniać strukturę danych zawierającą procesy, z którymi jest możliwa komunikacja.
- Procesy mogą także przechowywać całe trasy do znanych procesów.

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Przekazywanie danych między procesami

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Przekazywanie danych między procesami
- Wyszukiwanie tras w sieci

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Przekazywanie danych między procesami
- Wyszukiwanie tras w sieci
- Zarządzanie sieciami bezprzewodowymi (np. *sleep*, *wake up*, zmiana parametrów sieci)

Zastosowania alg. plotkujących w sieciach Ad Hoc

- Przekazywanie danych między procesami
- Wyszukiwanie tras w sieci
- Zarządzanie sieciami bezprzewodowymi (np. *sleep*, *wake up*, zmiana parametrów sieci)
- Synchronizacja (np. czasu)

Plan prezentacji

1 Wstęp

2 Przynależność do systemu

3 Świadomość topologii sieciowej

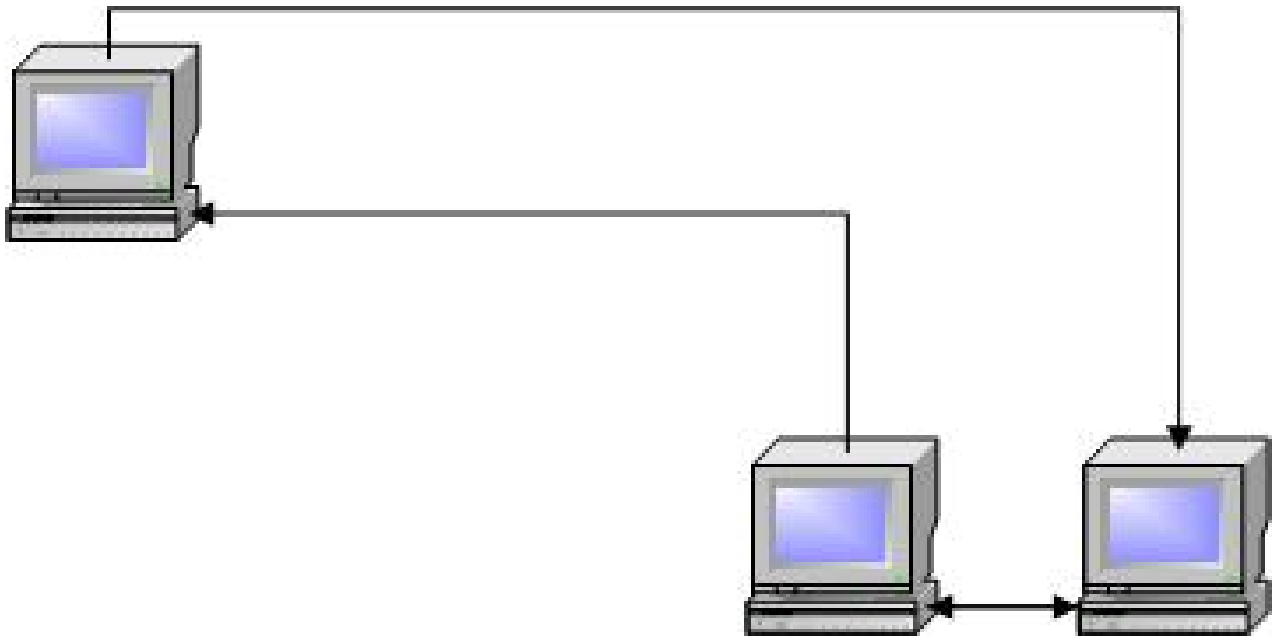
4 Zarządzanie buforem danych

5 Filtrowanie komunikatów

6 Inne problemy

Świadomość topologii sieciowej

Procesy nie wiedzą nic o fizycznej topologii sieci, stąd zakładają równy koszt dostępu do każdego procesu w systemie.



Rozwiązanie

Należy zorganizować procesy w hierarchię, która odzwierciedla topologię sieci oraz zaprogramować algorytm tak, aby ograniczał liczbę wysyłanych komunikatów między procesami z różnych gałęzi hierarchii.

Rozwiązanie

Należy zorganizować procesy w hierarchię, która odzwierciedla topologię sieci oraz zaprogramować algorytm tak, aby ograniczał liczbę wysyłanych komunikatów między procesami z różnych gałęzi hierarchii.

Budowanie dynamicznej i zdecentralizowanej hierarchii procesów jest jednak skomplikowane.

Wprowadzenie serwisów administracyjnych

- Wprowadzenie serwisu administracyjnego, który zna topologię sieci.

Wprowadzenie serwisów administracyjnych

- Wprowadzenie serwisu administracyjnego, który zna topologię sieci.
- Serwis administracyjny przydziela nowe procesy do hierarchii.

Wprowadzenie serwisów administracyjnych

- Wprowadzenie serwisu administracyjnego, który zna topologię sieci.
- Serwis administracyjny przydziela nowe procesy do hierarchii.
- Wymaga to jednak (przynajmniej częściowej) centralizacji.

Wprowadzenie serwisów administracyjnych

- Wprowadzenie serwisu administracyjnego, który zna topologię sieci.
- Serwis administracyjny przydziela nowe procesy do hierarchii.

- Wymaga to jednak (przynajmniej częściowej) centralizacji.
- Skomplikowana implementacja w dynamicznych systemach rozproszonych.

Wybór na podstawie parametrów dynamicznych

- Każdy proces oblicza dla pozostałych procesów ich wagi (np. czas przesłania komunikatu do procesu).

Wybór na podstawie parametrów dynamicznych

- Każdy proces oblicza dla pozostałych procesów ich wagi (np. czas przesłania komunikatu do procesu).
- Przy wyborze procesu, do którego ma być wysłany komunikat, preferowane są procesy z mniejszą wagą.

Organizacja drzewiasta

- Procesy są organizowane w drzewo (np. na podstawie ich adresów sieciowych).

Organizacja drzewiasta

- Procesy są organizowane w drzewo (np. na podstawie ich adresów sieciowych).
- Drzewo to automatycznie ustanawia hierarchię (np. algorytm preferuje przesyłanie komunikatów do procesów z tej samej podsieci).

Organizacja drzewiasta

- Procesy są organizowane w drzewo (np. na podstawie ich adresów sieciowych).
- Drzewo to automatycznie ustanawia hierarchię (np. algorytm preferuje przesyłanie komunikatów do procesów z tej samej podsieci).
- Można także przy okazji implementować filtrowanie danych.

Organizacja drzewiasta

- Procesy są organizowane w drzewo (np. na podstawie ich adresów sieciowych).
- Drzewo to automatycznie ustanawia hierarchię (np. algorytm preferuje przesyłanie komunikatów do procesów z tej samej podsieci).
- Można także przy okazji implementować filtrowanie danych.

- Adresacja procesów musi zawierać informacje o topologii sieciowej.

Sieci Ad Hoc

W sieciach Ad Hoc znajomość topologii sieciowej jest wymagana, aby możliwa była jakakolwiek komunikacja (nie jest to tylko kwestia wydajności).

Plan prezentacji

1 Wstęp

2 Przynależność do systemu

3 Świadomość topologii sieciowej

4 Zarządzanie buforem danych

5 Filtrowanie komunikatów

6 Inne problemy

Wielkość bufora danych

- Bufor komunikatów może mieć wielkość statyczną lub dynamiczną.

Wielkość bufora danych

- Bufor komunikatów może mieć wielkość statyczną lub dynamiczną.
- Bufor nie może być zbyt duży ze względu na efektywność (zużycie pamięci).

Wielkość bufora danych

- Bufor komunikatów może mieć wielkość statyczną lub dynamiczną.
- Bufor nie może być zbyt duży ze względu na efektywność (zużycie pamięci).
- Wszystkie komunikaty otrzymane przez proces muszą być buforowane (do momentu zapełnienia bufora) oraz przesłane t razy do losowego zbioru procesów.

Wielkość bufora danych

- Bufor komunikatów może mieć wielkość statyczną lub dynamiczną.
- Bufor nie może być zbyt duży ze względu na efektywność (zużycie pamięci).
- Wszystkie komunikaty otrzymane przez proces muszą być buforowane (do momentu zapełnienia bufora) oraz przesłane t razy do losowego zbioru procesów.

- W zależności od przepustowości sieci, wielkość bufora może nie być wystarczająca do przesłania każdego komunikatu t razy!

Usuwanie komunikatów z bufora

Kiedy proces odbiera komunikat i nie ma miejsca w buforze to musi usunąć jeden z komunikatów w buforze. Który?

Odrzucenie otrzymanego komunikatu

- Można odrzucić otrzymany komunikat (nie buforować go).

Odrzucenie otrzymanego komunikatu

- Można odrzucić otrzymany komunikat (nie buforować go).
- Może to spowodować "zapchanie" sieci, w skutek czego nowe informacje nie będą przekazywane między procesami!

Rozwiązanie oparte o priorytety

- Dla każdego komunikatu definiujemy jego wiek – tzn. ile razy został przesłany.

Rozwiązanie oparte o priorytety

- Dla każdego komunikatu definiujemy jego wiek – tzn. ile razy został przesłany.
- Przesyłając komunikat przesyłamy także jego wiek.

Rozwiązanie oparte o priorytety

- Dla każdego komunikatu definiujemy jego wiek – tzn. ile razy został przesłany.
- Przesyłając komunikat przesyłamy także jego wiek.
- W przypadku braku miejsca w buforze usuwamy najstarszy komunikat (czyli ten który został przesłany najwięcej razy).

Rozwiązanie oparte o priorytety

- Dla każdego komunikatu definiujemy jego wiek – tzn. ile razy został przesłany.
- Przesyłając komunikat przesyłamy także jego wiek.
- W przypadku braku miejsca w buforze usuwamy najstarszy komunikat (czyli ten który został przesłany najwięcej razy).
- Rozwiązanie to może zapewnić niezawodność ograniczając jednocześnie ilość używanych zasobów (pamięci).

Rozwiązanie oparte o semantykę danych

- Aplikacja wybiera, które komunikaty nie są jej potrzebne na podstawie przechowywanych w nich danych.

Rozwiązanie oparte o semantykę danych

- Aplikacja wybiera, które komunikaty nie są jej potrzebne na podstawie przechowywanych w nich danych.
- Zależne od logiki aplikacji – programista musi zaprojektować algorytm usuwający komunikaty z bufora.

Rozwiązanie oparte o semantykę danych

- Aplikacja wybiera, które komunikaty nie są jej potrzebne na podstawie przechowywanych w nich danych.
- Zależne od logiki aplikacji – programista musi zaprojektować algorytm usuwający komunikaty z bufora.
- Metoda ta może być łączona z rozwiązaniem opartym o priorytety.

Redukcja przepływu informacji

Innym sposobem zapewnienia skalowalności zasobów utrzymując niezawodność algorytmu, jest ograniczenie ilości informacji przepływanych między procesami.

Redukcja przepływu informacji

- Każdy proces oblicza średnią zajętość buforów procesów, z którymi się komunikuje (i przesyła tę informację w czasie komunikacji z innymi procesami).

Redukcja przepływu informacji

- Każdy proces oblicza średnią zajętość buforów procesów, z którymi się komunikuje (i przesyła tę informację w czasie komunikacji z innymi procesami).
- Kiedy jeden z procesów wykryje że średnia zajętość buforów jest zbyt wysoka, to ogranicza ilość wysyłanych komunikatów.

Redukcja przepływu informacji

- Każdy proces oblicza średnią zajętość buforów procesów, z którymi się komunikuje (i przesyła tę informację w czasie komunikacji z innymi procesami).
- Kiedy jeden z procesów wykryje że średnia zajętość buforów jest zbyt wysoka, to ogranicza ilość wysyłanych komunikatów.

- Główną wadą tego rozwiązania jest możliwość ograniczania przepustowości przez procesy z najbardziej zajętymi buforami.

Plan prezentacji

1 Wstęp

2 Przynależność do systemu

3 Świadomość topologii sieciowej

4 Zarządzanie buforem danych

5 Filtrowanie komunikatów

6 Inne problemy

Filtrowanie komunikatów

Czy system ma dostarczać każdy komunikat do każdego procesu?

Filtrowanie komunikatów

Czy system ma dostarczać każdy komunikat do każdego procesu?

Jeśli nie, to można ograniczyć ilość przesyłanych informacji poprzez filtrowanie – czyli dostarczanie komunikatów tylko do procesów zainteresowanych nimi.

Filtrowanie komunikatów

Czy system ma dostarczać każdy komunikat do każdego procesu?

Jeśli nie, to można ograniczyć ilość przesyłanych informacji poprzez filtrowanie – czyli dostarczanie komunikatów tylko do procesów zainteresowanych nimi.

Skąd proces ma wiedzieć jakie komunikaty interesują inne procesy?

Wpływ procesów na otrzymywaną treść

- Każdy proces w systemie określa jaką treścią jest zainteresowany.

Wpływ procesów na otrzymywaną treść

- Każdy proces w systemie określa jaką treścią jest zainteresowany.
- System stara się zwiększyć prawdopodobieństwo, że proces otrzyma komunikaty o treści, którą jest zainteresowany.

Implementacja

- Każdy proces przechowuje w jakiejś strukturze danych zainteresowania znanych procesów.

Implementacja

- Każdy proces przechowuje w jakiejś strukturze danych zainteresowania znanych procesów.
- Zanim komunikat zostanie przesłany do procesu, jego treść jest sprawdzana pod kątem zgodności z zainteresowaniami danego procesu.

Podział na grupy

- Każdy proces należy do 0 lub więcej grup, które reprezentują zainteresowania procesów.

Podział na grupy

- Każdy proces należy do 0 lub więcej grup, które reprezentują zainteresowania procesów.
- Każdy proces posiada informacje, do których grup należą znane mu procesy.

Podział na grupy

- Każdy proces należy do 0 lub więcej grup, które reprezentują zainteresowania procesów.
- Każdy proces posiada informacje, do których grup należą znane mu procesy.
- Każdemu komunikatowi można przyporządkować grupy, do których należy.

Podział na grupy

- Każdy proces należy do 0 lub więcej grup, które reprezentują zainteresowania procesów.
- Każdy proces posiada informacje, do których grup należą znane mu procesy.
- Każdemu komunikatowi można przyporządkować grupy, do których należy.
- Komunikaty są przesyłane do procesów, które należą do grup przyporządkowanych tych komunikatom.

Podział na grupy

- Każdy proces należy do 0 lub więcej grup, które reprezentują zainteresowania procesów.
- Każdy proces posiada informacje, do których grup należą znane mu procesy.
- Każdemu komunikatowi można przyporządkować grupy, do których należy.
- Komunikaty są przesyłane do procesów, które należą do grup przyporządkowanych tych komunikatom.

- Podział na grupy na powyższych zasadach może nie być prosty.

Użycie drzewiastej organizacji procesów

- Procesy są zorganizowane w drzewiastej strukturze danych (np. wg podsieci).

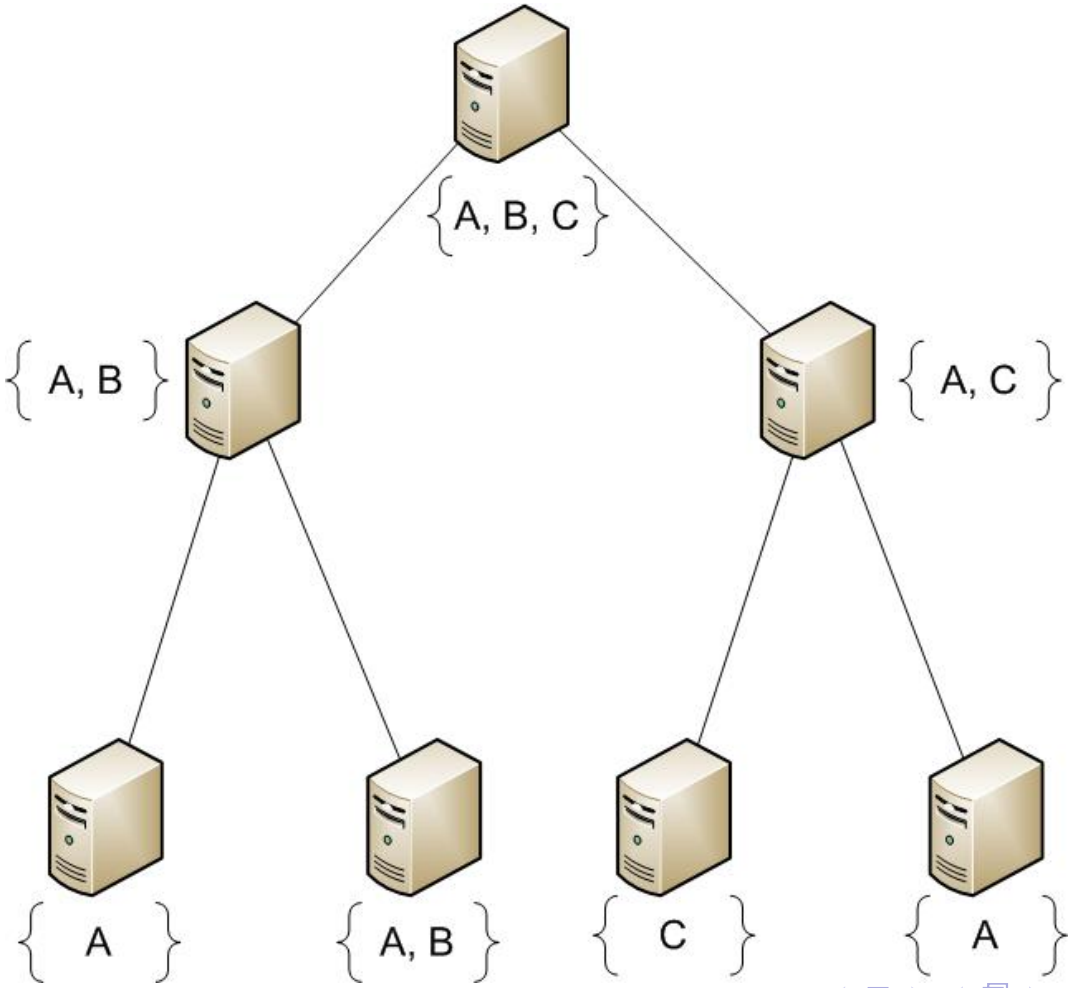
Użycie drzewiastej organizacji procesów

- Procesy są zorganizowane w drzewiastej strukturze danych (np. wg podsieci).
- W każdym węźle drzewa utrzymywane są zainteresowania procesów w poddrzewie.

Użycie drzewiastej organizacji procesów

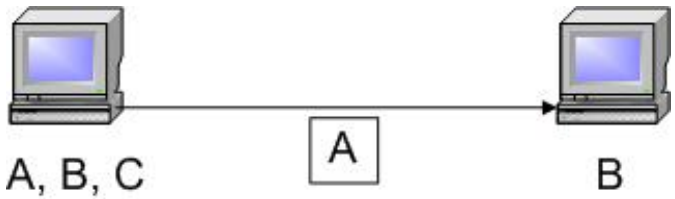
- Procesy są zorganizowane w drzewiastej strukturze danych (np. wg podsieci).
- W każdym węźle drzewa utrzymywane są zainteresowania procesów w poddrzewie.
- Przed przesłaniem komunikatu do poddrzewa, jest on weryfikowany z zainteresowaniami poddrzewa.

Użycie drzewiastej organizacji procesów



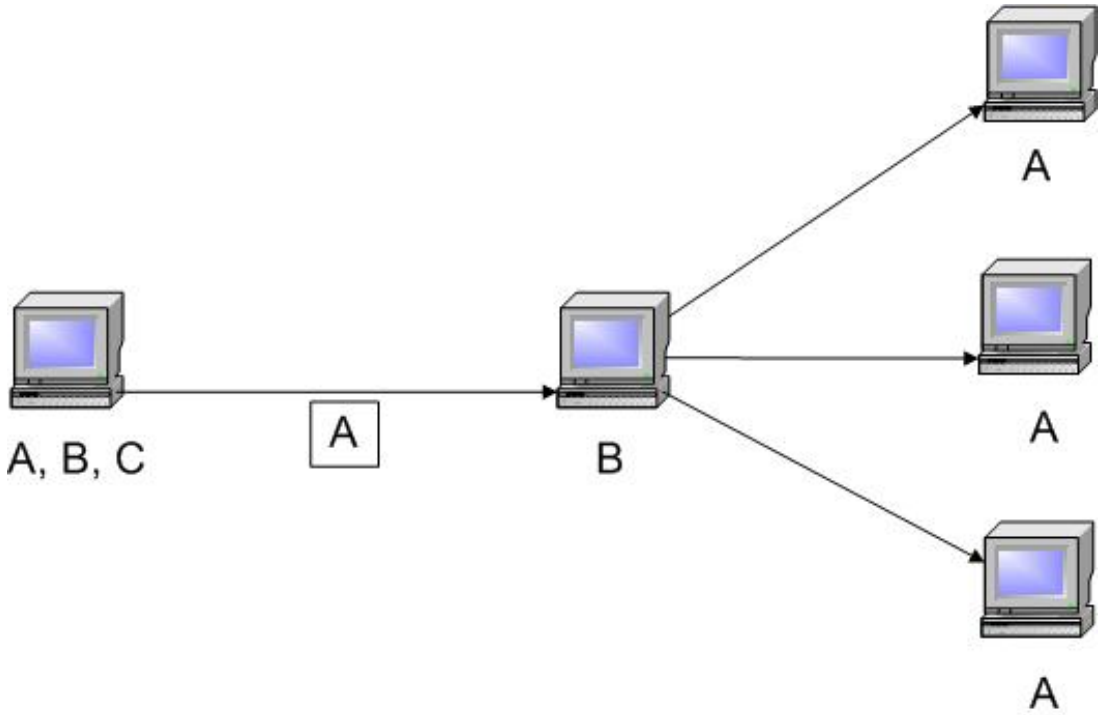
Procesy niezainteresowane

Jeśli dany proces nie jest zainteresowany pewnym komunikatem, to czy inne procesy mają mu go po prostu nie przesyłać?



Procesy niezainteresowane

Nie przesłanie komunikatu do procesu niezainteresowanego może uniemożliwić przesłanie tego komunikatu do innych procesów zainteresowanych!



Uwagi

- Filtrowanie jest dużo łatwiej zaimplementować jeśli każdy proces zna wszystkie inne procesy.

Uwagi

- Filtrowanie jest dużo łatwiej zaimplementować jeśli każdy proces zna wszystkie inne procesy.
- W systemie z częściową znajomością procesów, jakość filtrowania jest zależna od implementacji przynależności do systemu.

Uwagi

- Filtrowanie jest dużo łatwiej zaimplementować jeśli każdy proces zna wszystkie inne procesy.
- W systemie z częściową znajomością procesów, jakość filtrowania jest zależna od implementacji przynależności do systemu.
- Filtrowanie komunikatów zazwyczaj zmienia losowy wybór odbiorców komunikatów na wybór heurystyczny.

Plan prezentacji

1 Wstęp
2 Przynależność do systemu
3 Świadomość topologii sieciowej
4 Zarządzanie buforem danych
5 Filtrowanie komunikatów
6 Inne problemy

Integralność danych

- Komunikacja między procesami systemu jest często zawodna.

Integralność danych

- Komunikacja między procesami systemu jest często zawodna.
- Jak zapewnić, że dane przesyłane pomiędzy procesami nie zostaną po drodze uszkodzone?

Integralność danych

Potencjalne rozwiązanie: Używanie cyfrowych podpisów komunikatów i obliczanie sum kontrolnych

Integralność danych

Potencjalne rozwiązanie: Używanie cyfrowych podpisów komunikatów i obliczanie sum kontrolnych

- Duży koszt obliczeniowy

Integralność danych

Potencjalne rozwiązanie: Używanie cyfrowych podpisów komunikatów i obliczanie sum kontrolnych

- Duży koszt obliczeniowy
- Obciąża sieć nawet jeśli wiadomości są poprawnie przesyłane

Złośliwe procesy

- Sieci Ad Hoc są często heterogeniczne – procesy należące do systemu mogą zachowywać się różnie

Złośliwe procesy

- Sieci Ad Hoc są często heterogeniczne – procesy należące do systemu mogą zachowywać się różnie
- Brak centralnej usługi monitorującej procesy systemu

Złośliwe procesy

- Sieci Ad Hoc są często heterogeniczne – procesy należące do systemu mogą zachowywać się różnie
- Brak centralnej usługi monitorującej procesy systemu
- Jak radzić sobie z procesami, które nie przestrzegają protokołu komunikacji (przez co zmniejszają wydajność systemu)?

Złośliwe procesy

Potencjalne rozwiązanie: Monitorowanie procesów, z którymi odbywa się komunikacja

Złośliwe procesy

Potencjalne rozwiązanie: Monitorowanie procesów, z którymi odbywa się komunikacja

- Skalowalność wymaga używania głównie lokalnej komunikacji z procesami

Agregacja danych

Jak odczytywać dane globalnych oraz łączyć dane z pojedynczych procesów?

Podsumowanie

- Algorytmy plotkujące umożliwiają implementacje efektywnej wymiany informacji w dużych, zdecentralizowanych systemach rozproszonych.

Podsumowanie

- Algorytmy plotkujące umożliwiają implementacje efektywnej wymiany informacji w dużych, zdecentralizowanych systemach rozproszonych.
- Dzięki lokalnym interakcją ze znanymi procesami algorytmy plotkujące są skalowalne oraz odporne na awarie i dynamiczne zmiany w sieci (*churn*).

Podsumowanie

- Algorytmy plotkujące umożliwiają implementacje efektywnej wymiany informacji w dużych, zdecentralizowanych systemach rozproszonych.
- Dzięki lokalnym interakcją ze znanymi procesami algorytmy plotkujące są skalowalne oraz odporne na awarie i dynamiczne zmiany w sieci (*churn*).
- Algorytmy plotkujące mogą się przystosowywać do ilości dostępnych zasobów oraz obciążenia sieci zapewniając jednocześnie niezawodność.

Podsumowanie

- Efektywność algorytmu plotkującego zależy w dużej mierze od jakości informacji o przynależności procesów do systemu.

Podsumowanie

- Efektywność algorytmu plotkującego zależy w dużej mierze od jakości informacji o przynależności procesów do systemu.
- Można znacznie poprawić efektywność algorytmów plotkujących poprzez implementacje filtrowania oraz uświadamiania topologii sieciowej.

Podsumowanie

- Algorytmy plotkujące są coraz częściej wykorzystywane w ogólnych sieciach rozproszonych (np. Tribler).

Podsumowanie

- Algorytmy plotkujące są coraz częściej wykorzystywane w ogólnych sieciach rozproszonych (np. Tribler).
- Można się spodziewać znacznego wzrostu popularności algorytmów plotkujących wraz z wzrostem popularności sieci bezprzewodowych oraz Ad Hoc.

Pytania?

Pytania?

Bibliografia

- Patrick T. Eugster, Rachid Guerraoui, Anne-Marie Kermarrec, Laurent Massoulie, *Epidemic Information Dissemination in Distributed Systems*
- Deepak Ganesan, Bhaskar Krishnamachari, Alec Woo, David Culler, Deborah Estrin, Stephen Wicker, *An Empirical Study of Epidemic Algorithms in Large Scale Multihop Wireless Networks*

Bibliografia

- Daniela Gavidia, Maarten van Steen, *Enforcing Data Integrity in Very Large Ad Hoc Networks*
- Mark Jelasity, Spyros Voulgaris, Rachid Guerraoui, Anne-Marie Kermarrec, Maarten van Steen, *Gossip-based Peer Sampling*

Koniec!

Koniec!