Agregacja oparta na algorytmach plotkujących

Piotr Skowron

March 12, 2009

Spis treści

Najogólniej o agregacji

Definicja

Agregacja, w ujęciu systemów rozproszonych, jest to możliwość podsumowywania informacji. W przypadku systemów sensorowych bywa także naywana fuzją danych (ang. *data fusion*)

O tym nie będzie

Istnieje wiele innych definicji z innych dziedzin informatyki - najbardziej popularne to te z programowania obiektowego

Motywacja

Agregacja znajduje wiele zastosowań we współczesnej informatyce

Zastosowania agregacji

1. Agregacja adresów internetowych powala na skalowalność sieci
   - Bez agregacji tablice routowania potrzebowałyby osobnego wpisu dla każdego adresu
   - Problem pamięci
   - Propagacja tablic byłaby niemożliwa
   - Serwery DNS używają agregacji i pozwalają aby mapowanie nazw na adresy zostało przeprowadzone w kilku krokach
2. Popularna synchronizacja oparta na głosowaniu wymaga agregacji - głosy muszą zostać policzone
3. Agregacja jest używana w bazach danych - użytkownik może zbierać dane z różnych tabelek

Problemy związane z agregacją

1. Mechanizmy te wymagają dużej konfiguracji
2. Konfiguracja ta nie jest współdzielona
3. Konfiguracja jest statyczna
   - słabo adaptuje się do dynamicznie zwiększających się sieci
   - źle reaguje na sytuacje błędne

Przykłady funkcji agregacyjnych

1. wyznaczenie rozmiaru sieci
2. wyznaczenie całkowitego rozmiaru wolnej przestrzeni dyskowej
3. wyznaczenie maksymalnego ruchu w sieci
4. wyznaczenie średniego czasu nieprzerwanej pracy serwera (ang. *uptime*)
5. znalezienie hotspotów
6. elementy bardziej wyszukanych protokołów - np. znajdowanie średniego obciążenia w systemie może być wykorystywane w algorytmach znajdowania optymalnego zbalansowania obciążenia (ang. *near-optimal load-balancing schemes*)

Najogólniejszy podział protokołów używanych do wyznaczania funkcji agregacyjnych

1. reaktywne (ang. *reactive*)
   1. Węzeł 1 wysyła do węzła 2 zapytanie
   2. Węzeł 2 wysyła do węzła 1 odpowiedź
   3. Reszta węzłów w sieci może nie być świadoma odpowiedzi
2. zapobiegawcze (ang. *proactive*)
   - informacja jest w ciągły sposób przekazywana do **wszystkich** węłów
   - informacja jest przekazywana w sposób adaptywny

Model systemu

- duża liczba węzłów z których każdy ma przypisany unikalny identyfikator i które komunikują się poprzez wymianę komunikatów
- sieć jest dynamiczna
  1. nowe węzły mogą dołączać do sieci w sposób spontaniczny :)
  2. węzły mogą opuszczać sieć dobrowolnie lub w wyniku awarii
  3. nie zmniejszając ogólności możemy ograniczyć się do awarii
- węzły są połączone w spójną sieć

Model systemu

- aby umożliwić komunikację każdy węzeł musi znać pewną liczbę innych węzłów nazywanych *sąsiadami*
  1. relacja sąsiedztwa definiuje topologię sieci
  2. sąsiedztwo jest zazwyczaj ograniczone do małych podzbiorów sieci
  3. zbiór sąsiadów zmienia się dynamicznie
- komunikacja
  1. komunikaty mogą się gubić
  2. połączenia między węzłami mogą być zrywane
  3. możliwe są opóźnienia komunikatów
- węzły mają dostęp do lokalnych zegarów które mogą mierzyć czas z sensowną dokładnością

Zadania protokołu

1. Każdy węzeł trzyma pewną numeryczną wartość charakteryzującą jego stan
   - obciążenie węzła
   - dostępna przestrzeń dyskowa
   - zmierzona temperatura (przypadek systemów sensorowych)
2. rozważamy protokół zapobiegawczy (*proactive*)
3. zadaniem protokołu jest ciągłe dostarczanie do wszystkich węzłów przybliżonej wartości funkcji agregującej

Schemat komunikacji

do exactly once in each consecutive
δ time units at a radomly
picked time
q $<-$ GETNEIGHBOUR()
sends s_p to q
$s_q <-$ receive(q)
$s_p <-$ UPDATE(sp, sq)

aktywny wątek

do forever
$s_q <-$ receive(*)
send s_p to sender(s_q)
$s_p <-$ UPDATE(s_p, s_q)

pasywny wątek

Ogólnie o protokole

1. Protokół jest wykonywany w interwałach czasowych o długości δ nazywanych cyklami
2. GETNEIGHBOUR - *underlying service to the protocol* - najczęściej zaimplementowane za pomocą losowania spośród aktualnie aktywnych sąsiadów
3. UPDATE - funkcja zależna od funkcji agregacyjnej którą chcemy wyznaczyć
4. W naszej analizie ograniczymy się do wyznaczania średniej arytmetycznej
   - Każdy węzeł trzyma pojedynczą wartość numeryczną która jest przybliżeniem ostatecznej wartości funkcji agregacyjnej
   - UPDATE$(s_p, s_q) = (s_p + s_q) \,/\, 2$

Poprawność protokołu

1. Wymiana komunikatów nie zmienia wartości globalnej średniej w systemie
2. Wymiana komunikatów zmniejsza wariancję w systemie
3. Wartość trzymana przez poszczególne węzły zbiega do rzeczywistej wartości którą chcemy wynaczyć
   - rozważmy minimalną wartość w systemie
   - minimalna wartość w systemie wzrasta lub liczba węzłów posiadających tę minimalą wartość maleje (chyba że wszystkie węzły mają tę wartość)
   - jeżeli istnieje choć jedna wartość różna od minimalnej (czyli większa) w systemie to istnieje dodatnie prawdopodobieństwo że węzeł posiadający tę wartość zostanie wybrany przez funkcję GETNEIGHBOUR

Zaczynamy matematykę

1. będziemy badać szybkość zbieżności
2. na początku założymy że komunikacja nie zawiera błędów i protokół jest zsynchronizowany - później rozluznimy te założenia
3. protokół będziemy traktować jako iteracyjne zmniejszanie wariancji w wektorze wartości numerycznych
4. inicjalny stan będziemy reprezentować jako wektor: $w_0 = (w_{0,1} \ldots w_{0,N})$
5. $w_{0,1}$, ..., $w_{0,N}$ oznaczają początkowe wartości węzłów w systemie - zakładamy że są to niezależne zmienne losowe z identycznymi wartościami oczekiwanymi

Zaczynamy matematykę

1. założenie o identycznych wartościach oczekiwanych nie jest zbyt restrykcyjne - po permutacji dystrybuanta rozkładu węzła i wygląda następująco:

$$P(b_i < x) = \frac{1}{N} \sum_{j=1}^{N} P(w_j < x)$$

2. rozkłady zmiennych $b_0, ..., b_N$ mają teraz identyczny rozkład

Uwaga

Zmienne $b_0, ..., b_N$ nie są teraz niezależne

Jak wygląda algorytm w naszym modelu

// vector **w** is the input
do N times
    (i,j) = GETPAIR()
    // perform elementary reduction step
    $w_i = w_j = (w_i + w_j)/2$
return w

funkcja AVG

Kontynuujemy matematykę

1. rezultatem algorytmu będzie ciąg wartości $w_1, ..., w_N$, gdzie $w_{i+1} = AVG(w_i)$
2. zachowanie naszego protokołu zależy od implementacji GETPAIR

Hobbistyczna uwaga

Istnieją implementacje GETPAIR, które nie odwierciedlają zachowania rozproszonego, protokołu. Są jednak interesujące ze względów teoretycznych :)

3. będziemy analizować różne implementacje GETPAIR

Jeszcze trochę pojęć

1. $\bar{w}_i$ jest oczekiwanym wynikiem

$$\bar{w}_i = \frac{1}{N}\sum_{k=1}^{N} w_{i,k}$$

2. σ_i^2 - variance-like measure of homogenity :) - określa jakość naszego algorytmu

$$\sigma_i^2 = \sigma_{w_i}^2 = \frac{1}{N}\sum_{k=1}^{N}(w_{i,k} - \bar{w}_i)^2$$

Kontynuujemy matematykę

1. w pojedynczym kroku jeżeli do inicjalnych wartości dodamy stałą C, to otrzymany wynik będzie większy o C
2. bez straty ogólności możemy założyć że wartość oczekiwana elementów wektora w_0 wynosi 0
3. $$E(\sigma_w^2) = \frac{1}{N} \sum_{k=1}^{N} E(w_k^2)$$
4. Pojedynczy krok algorytmu nie zmienia sumy elementów wektora, zatem: $\bar{w}_i = \bar{w}_0$ dla każdego i
5. Powyższa własność gwarantuje poprawność algorytmu
6. Możemy się skupić na zbadaniu zbieżności σ_w^2 do 0

Pierwszy lemat

Niech w' będzie wektorem otrzymanym poprzez zamianę wartości w_i oraz w_j przez $(w_i + w_j)/2$ w wektorze w. Jeżeli wektor w zawiera nieskorelowane zmienne losowe o wartości oczekiwanej równej 0, wtedy wartość oczekiwana redukcji wariancji jest dana wzorem:

$$E(\sigma_w^2 - \sigma_{w'}^2) = \frac{1}{2(N-1)} E(w_i^2) + \frac{1}{2(N-1)} E(w_j^2)$$

Dowód jest prostym podstawieniem i wykorzystaniem faktu, że dla zmiennych nieskorelowanych:

$$E(w_i w_j) = E(w_i)E(w_j) = 0$$

Żeby matematyka była trochę prostsza

1. Możemy zamiast σ_i^2 rozważać średnią wektora

$$s_i = (s_{i,1} \ldots s_{i,N})$$

$$s_0 = (w_{0,1}^2 \ldots w_{0,N}^2)$$

2. s_i jest produkowane równolegle do w_i używając tej samej pary (i, j) zwróconej przez GETPAIR. Równolegle do kroku algorytmu wykonujemy $s_i = s_j = (s_i + s_j) \ / \ 4$
3. $E(\bar{s}_i)$ dobrze emuluje zachowanie $E(\sigma_i)$

Uwaga

To wszystko jest prawdą tylko pod warunkiem nieskorelowania zmiennych

Drugi lemat

Niech ϕ_k liczbą razy jaką indeks k będzie wybrany jako element pary zwróconej przez GETPAIR w algorytmie AVG podczas obliczania w_{i+1} z w_i.
Jeżeli GETPAIR ma następujące własności:

1. zmienne $\phi_1, ..., \phi_N$ mają identyczny rozkład (przez ϕ oznaczmy zmienną losową o tym wspólnym rozkładzie)
2. po tym gdy para (i, j) jest zwrócona przez GETPAIR, liczba razy jaką i oraz j zostanie wybrana przez pozostałe wywołania GETPAIR mają identyczne rozkłady

wtedy mamy:

$$E(s_{i+1}^{-}) = E(2^{-\phi})E(\bar{s_i})$$

Szkielet dowodu drugiego lematu

1. Pomyślmy o $s_{i,k}$ jako zmiennej reprezentującej ilość pewnego materiału.
2. Za każdym razem gdy k zostaje wybrane przez GETPAIR tracimy połowę materiału k, a pozostała połowa zostaje podzielona pomiędzy różne lokacje.
3. Korzystając z drugiego założenia drugiego lematu zauważamy że nie ma znaczenia dokąd trafi pozostała część materiału. Będzie miał taką samą szansę stracenia swojej połowy jak gdyby pozostał na miejscu.
4. To oznacza że każdy element będzie "oddawał poza sieć" połowę swojego materiału tyle razy co średnia ilość wybrania k przez GETPAIR czyli: $\frac{1}{N}E(2^{-\phi_k})E(s_{i,k}) = \frac{1}{N}E(2^{-\phi})E(s_{i,k})$
5. Sumując po wszystkich k otrzymujemy tezę

Kontynuujemy matematykę

1. Wskaźnik kowergencji pomiędzy cyklem i oraz $i+1$ definiujemy $E(\sigma_{i+1}^2)/E(\sigma_i^2)$
2. Wskaźnik kowergencji jest bardzo dobrą miarą dynamizmu protokołu - odzwierciedla szybkość zbieżność wartości węzłów do rzeczywstej wartości funkcji agregacyjnej
3. Oczekujemy że jeżeli zmienne losowe są śladowo skorelowane, to:
$$E(\sigma_{i+1}^2) \approx E(2^{-\phi})E(\sigma_i^2)$$
4. Obliczenie wskaźnika kowergencji polega zatem na znalezieniu $E(2^{-\phi})$

Wskaźnik kowergencji

- nie zależy od rozmiaru sieci
- nie zależy od czasu
- nie zależy od inicialnego rozkładu wartości
- w zasadzie zależy tylko od funkcji GETPAIR

GETPAIR - *Perfect Matching*

- optymalna strategia
- niestety nie może być zaimplementowany, ponieważ wymaga globalnej wiedzy
- pierwsze N/2 par indeksów jest wybierane w ten sposób aby każdy indeks występował w dokładnie jednej parze - nazwijmy to pierwszym doskonałym dopasowaniem
- kolejne N/2 par indeksów jest tworzone w ten sam sposób co pierwsze doskonałe dopasowanie, z tym że staramy się aby żadna para z drugiego doskonałego dopasowania nie należała do pierwszego doskonałego dopasowania

GETPAIR - *Perfect Matching*

- Spełnione założenia drugiego lematu:
  1. każdy z węzłów jest wybierany stałą ilość razy - dokładnie dwa razy
  2. po pierwszym wyborze węzła i jest zagwarantowane że węzeł ten zostanie wybrany dokładnie jeszcze raz
- wskaźnik kowergencji wynosi $E(2^{-\phi}) = E(2^{-2}) = 1/4$
- optymalność algorytmu wynika z następującego spostrzeżenia: *Dla każdej zmiennej losowej* X *, jeśli* $E(X) = 2$ *wtedy wartość oczekiwana* $E(2^{-X})$ *jest minimalna gdy* $P(X = 2) = 1$

Perfect Matching - optymalność algorytmu

- Dla każdego innego rozkładu niż $P(X = 2) = 1$ możemy zmniejszyć wartość $E(2^{-X})$ poprzez przekształcenie istniejącego rozkładu do nowego, który także spełnia warunek $E(X) = 2$
- Jeżeli $P(X = 2) < 1$ wtedy istnieją co najmniej dwie wartości $i < 2$ oraz $j > 2$, że $P(X = i) > 0$ oraz $P(X = j) > 0$
- Można technicznie pokazać, że gdy zamiast $P(X = i)$ oraz $P(Y = i)$ zwiększymy $P(X = 2)$ tak aby zachować warunek $E(X) = 2$, wtedy $E(2^{-X})$ ulegnie zmniejszeniu

GETPAIR - *Random Choice*

- Optymistyczna uwaga: może zostać zaimplementowany
- GETPAIR zwraca losową parę różnych węzłów
- Implementacja:
  1. podczas iterowania funkcji AVG czas czekania pomiędzy dwoma kolejnymi wyborami tego samego węzłama rozkład wykładniczy
  2. w systemie rozproszonym możemy zasymulować to diałanie gdy każdy węzeł wznawia komunikację po interwale czasowym wylosowanym z tym właśnie rozkładem
- Spełnione są założenia drugiego lematu:
  1. dla każdego węzła prawdopodobieństwo jego wybrania jest takie same w każdym kroku algorytmu
  2. każdy węzeł ma dokładnie takie samo prawdopodobieństwo bycia wybranym, niezależnie od tego jakie pary były wybierane poprzednio

Random Choice - wskaźnik kowergencji

- Rokład ϕ może być przybliżony przez rozkład Poisson'a z parametrem 2:

$$P(\phi = j) = \frac{2^j}{j!} e^{-2}$$

- Podstawiając do wzoru na wskaźnik kowergencji otrzymujemy:

$$E(2^{-\phi}) = \sum_{j=1}^{\infty} 2^{-j} \frac{2^j}{j!} e^{-2} = e^{-2} \sum_{j=1}^{\infty} \frac{1}{j!} = e^{-2} e = e^{-1}$$

- Porównując z rozwiązaniem optymalnym nasz rezultat nie jest najlepszy ($e^{-1} \approx 1/2.71$ vs. $1/4$).

GETPAIR - *Distributed Sollution*

- Symulujemy zachowanie funkcji AVG w systemie rozproszonym
- Każdy węzeł powinien być członkiem co najmniej jednej pary
- Możemy to osiągnąć biorąc losową permutację liczb od 1 do N
- Ten protokół jest nie tylko implementowalny ale również jego jakość jest lepsza niż w przypadku *Random Choice*
- Zmienna losowa ϕ może być przybliżona jako $\phi = 1 + \phi'$, gdzie ϕ' ma rozkład Poisson'a z parametrem 1:

$$P(\phi = j) = P(\phi' = j - 1) = \frac{1}{(j-1)!} e^{-1}$$

Distributed Sollution - wskaźnik kowergencji

- Podstawiając do wzoru na wskaźnik kowergencji otrzymujemy:

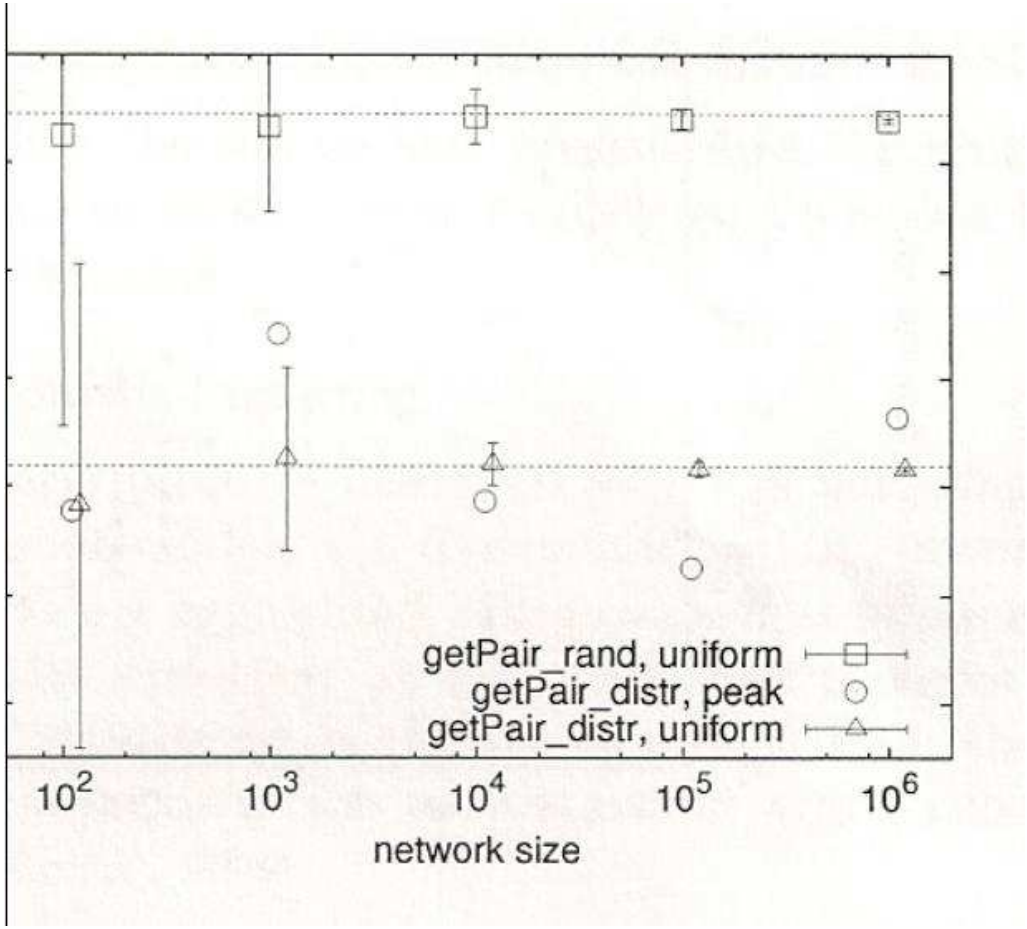
$$E(2^{-\phi}) = \sum_{j=1}^{\infty} 2^{-j} \frac{1}{(j-1)!} e^{-1} = \frac{1}{2e} \sum_{j=1}^{\infty} \frac{2^{-(j-1)}}{(j-1)!} = \frac{1}{2e} \sqrt{e} = \frac{1}{2\sqrt{e}}$$

- Porównując z poprzednimi rozwiązaniami: $\frac{1}{2\sqrt{e}} \approx 1/3.3$ vs. $e^{-1} \approx 1/2.71$ vs. $1/4$.

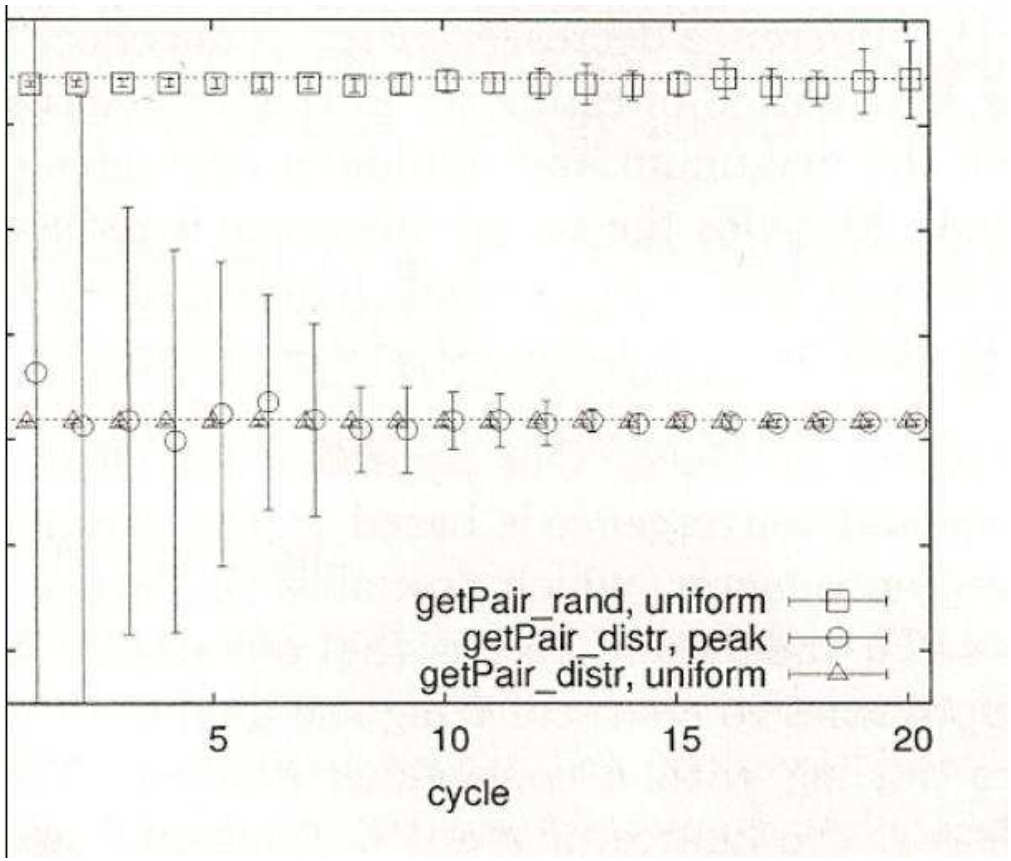
Środowisko doświadczeń

- Eksperymenty były przeprowadzane dla różnych rozmiarów sieci i dla różnych rozkładów inicjalnych
- Dla każdego ustawienia było przeprowadzanych 50 niezależnych eksperymentów
- Przypomnijmy sobie że wskaźnik kowergencji nie zależy od inicjalnych ustawień. Aby to przetestować sieć byla inicjowana na dwa różne sposoby:
  - *uniform* - każdy węzeł otrzymuje losową wartość z tego samego przedziału
  - *peak* - losowo wybrany węzeł otrzymuje pewną niezerową wartość, a pozostałe węzły otrzymują 0 - w tym przypadku odchylenie jest największe - reprezentuje to zatem potencjalnie najgorszy przypadek

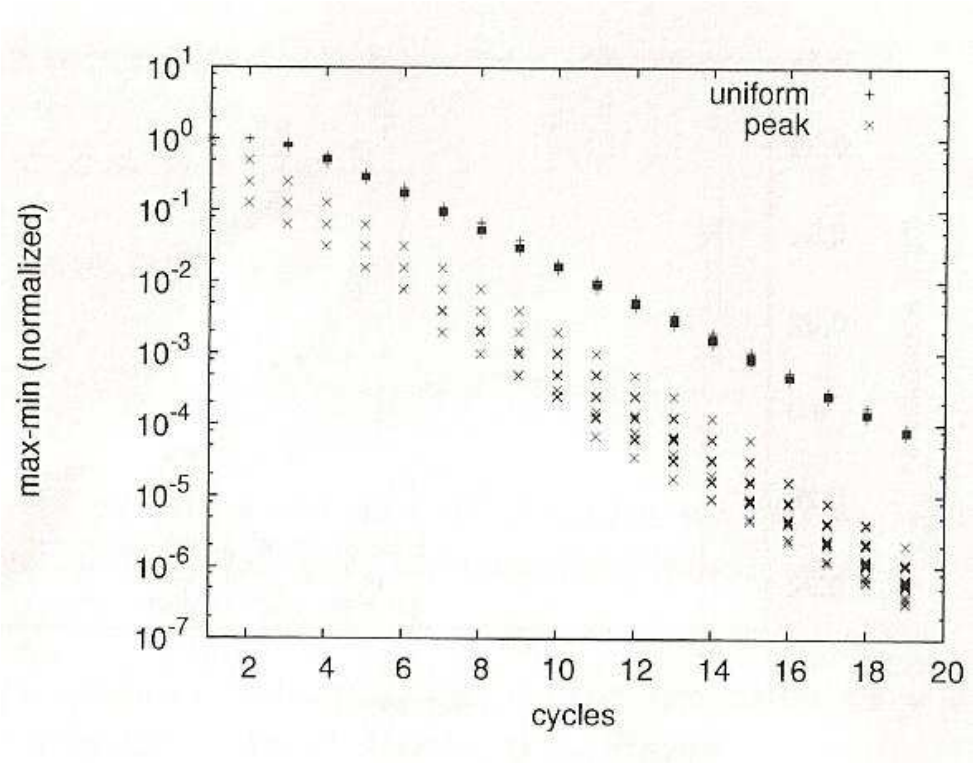
Wskaźnik kowergencji nie zależy od rozmiaru sieci



Wskaźnik kowergencji dla rozkładów inicjalnych: *uniform* i *peak*



Różnica pomiędzy górną a dolną estymacją dla rozkładów inicjalnych: *uniform* i *peak*



Co tak naprawdę mierzymy

- Nie skupiamy się na wariancji wskaźnika kowergencji
- Mierzymy proces nieskończony więc jakość rozwiązania zależy od czasu działania protokołu
- σ_i charakteryzuje dokładność przybliżeń w jednym cyklu
- W naszym rozwiązaniu nie badamy średniego odchylenia, ale badamy średnią tych średnich odchyleń w różnych przebiegach algorytmu - badamy $E(\sigma_i)$
- Pojedynczy węzeł w określonym przebiegu może mieć różną dokładność przybliżenia
- Nie badamy rozkładu dokładności - rozkład ten zależy od początkowego rozkładu wartości po węłach

Completeness - inna forma miary

- Zakładamy że agregata jest liczona w oparciu o wiedzę pewnego podzbioru węzłów (w idealnym lecz ze względu na błędy mało realistycznym przypadku rozważamy cały zbiór)
- Miara daje procent wartości które zostały wzięte pod uwagę podczas wyliczania agregaty
- W naszym rozwiązaniu ciężko to zinterpretować ponieważ lokalną agregatę w każdym kroku możemy rozważać jako średnią ważoną inicjalnych wartości - idealny przypadek - każda wartość powinna występować z tą samą wagą
- Aby przybliżyć tę miarę można rozważać rozkład tych wag, jako funkcję czasu

Automatyczne restarty

- Przedstawiony do tej pory protokół generyczny jest niepraktyczny
  1. nie bierze pod uwagę dynamizmu sieci
  2. nie bierze pod uwagę zmienności wartości które są agregowane
- Protokół musi być co pewien czas restartowany
  1. na każdym węźle co pewien czas protokół zamykany i jako wartość funkcji agregacyjnej zwracana jest aktualna wartość przybliżenia
  2. lokalne wartości są używane jako wartości początkowe przy ponownym starcie protokołu

Jak zaimplementować zamknięcie protokołu

- Każdy węzeł wywołuje protokół określoną liczbę razy - oznaczmy tę liczbę jako γ - liczba ta powinna zależeć od dokładności którą chcemy osiągnąć - jest zatem dobierana w zależności od wskaźnika kowergencji dla danej topologii
- Aby zaimplementować restartowanie protokołu:
  1. dzielimy wykonanie protokołu na epoki - długości $\Delta = \gamma\delta$ (gdzie δ jest długością cyklu)
  2. w każdej epoce rozpoczynamy nową instancję protokołu
  3. wszystkie komunikaty zawierają identyfikator epoki, który będzie używany przez system synchronizujący

Jak radzić sobie ze zjawiskiem podłączania się nowych węzłów

1. Zjawisko podłączania się nowych węzłów jest powszechne (ang. *churn*)
2. Jeżeli dany węzeł A chce przyłączyć się do sieci to kontaktuje się z dowolnym węzłem B działającym w sieci - zakładamy istanienie mechanizmu który umożliwi znalezienie takiego węzła
3. B przesyła do A
   1. identyfikator epoki
   2. czas który pozostał do rozpoczęcia nowej epoki
4. A nie ma prawa zacząć działać w aktualnej epoce
   1. chcemy mieć pewność że wartość wyliczana w danej epoce będzie zbiegać do rzeczywistej wartości z początku danej epoki
   2. ciągłe dodawanie nowych węzłów spowodowałoby niemożliwość uzgodnienia wartości

Jak radzić sobie z opuszczaniem się przez węzły

1. Węzeł ropoczynający wymianę komunikatów ustawia pewien timeout. W momencie gdy nie otrzyma odpowiedzi na swój komunikat przed upływem timeout'u dany krok w protokole zostanie pominięty
2. problem usuwania niedostępnych węzłów z sieci zostanie omówiony w dalszej części prezentacji - implementacja GETNEIGHBOUR za pomocą NEWSCAST

Synchronizacja

1. Do tej pory zakładaliśmy synchronizację cykli i epok. Jest to założenie nierealne ze względu na:
   - niespójności w lokalnych zegarach
   - opóźnienia komunikatów
2. Rozwiązanie: Jeżeli dany węzeł uczestniczy w epoce i oraz otrzyma komunikat otagowany identyfikatorem epoki j takim że $j > i$, przestaje on uczestniczyć w epoce i a zaczyna uczestniczyć w j
3. informacja o późniejszej epoce jest zatem propagowana zgodnie z protokołem plotkującym

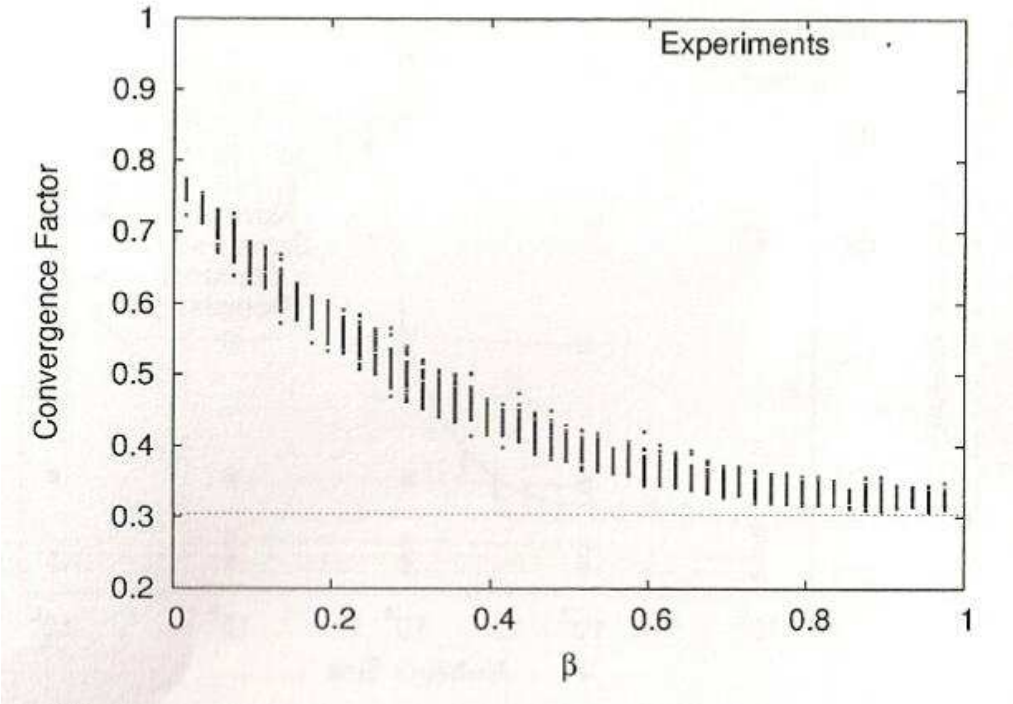
Znaczenie topologii sieci

1. Do tej pory zakładaliśmy że topologia sieci jest wystarcająco losowa - tzn. że GETNEIGHBOUR może zwrócić wystarcająco losowy podzbiór węzłów
2. Teraz będziemy rozważać konkretne topologie. Wszystkie za wyjątkiem jednej będą statyczne
3. Poza wyjątkiem pełnej sieci wszystkie rozważane topologie będą miały około 20 sąsiadów (stopień węzła)
4. Będziemy rozważać model topologii Watts-Strogatz

Model topologii Watts-Strogatz - połączenie pierścieniowe

1. Łączymy wszystkie węzły w pierścień
2. Od każdego węzła dodajemy połączenia do najbliższych sąsiadów do momentu aż węzeł uzyska określony stopień
3. Każda krawędź zostaje teraz z prawdopodobieństwem β przepięta. Przepięcie krawędzi przy węźle n polega na usunięciu danej krawędzi i podpięciu n do innego losowo wybranego węzła
   - dla $\beta = 0$ pierścień pozostaje niezmieniony
   - dla $\beta = 1$ wszystkie krawędzie zostają przepięte i dostajemy losowy graf

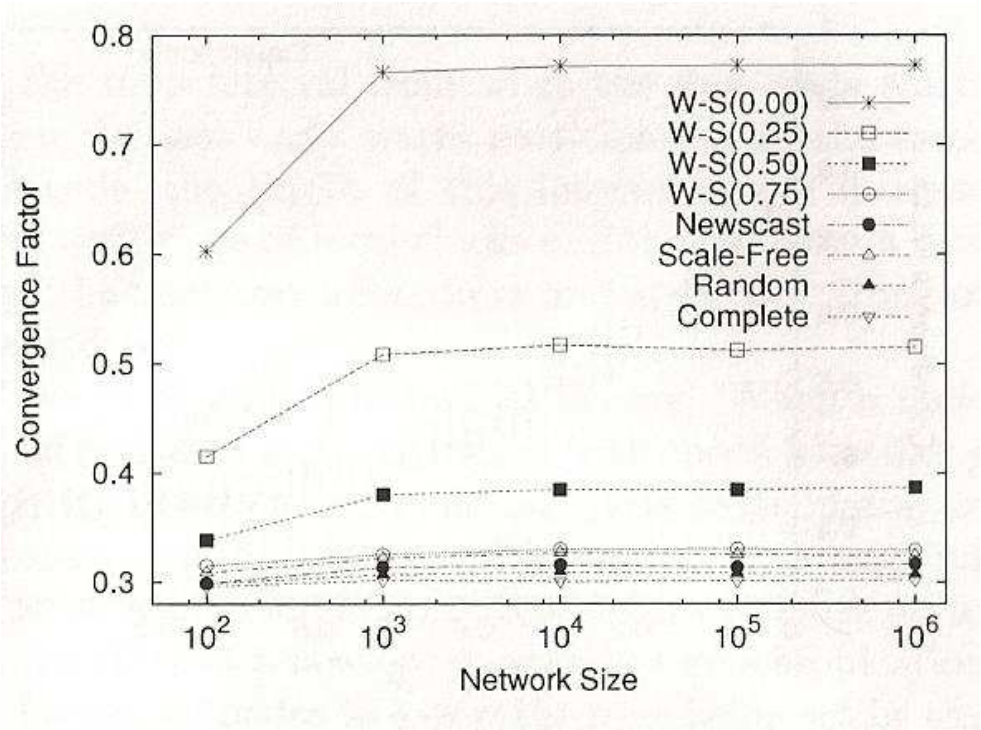
Wskaźnik kowergencji dla modelu Watts-Strogatz w zależności od zmieniającego się β



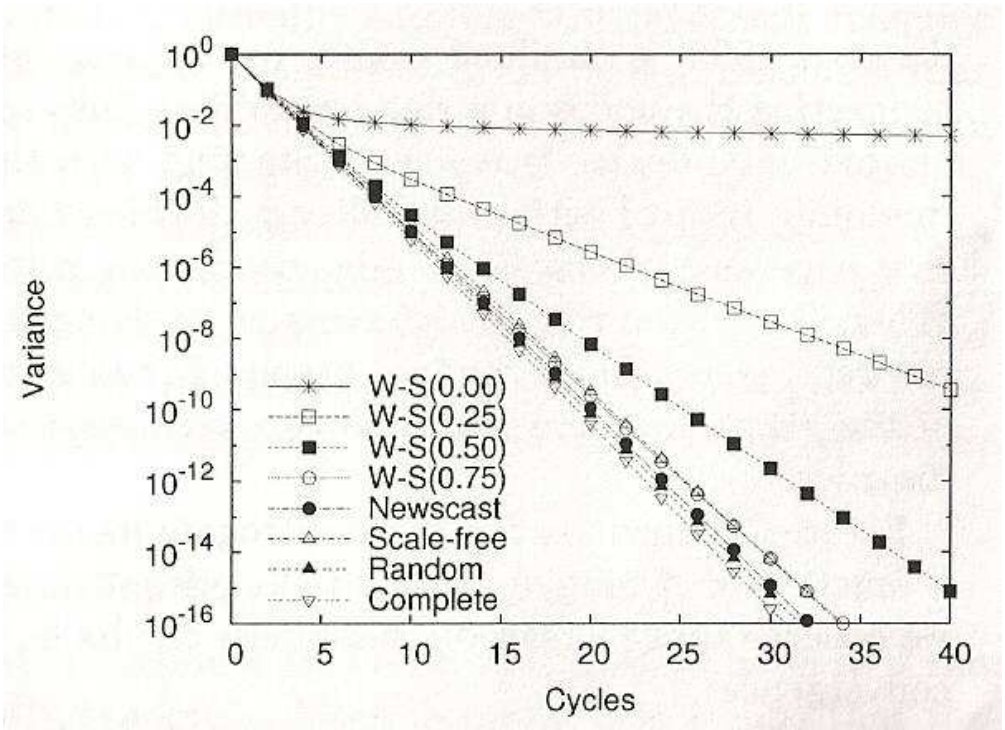
Topologia scale-free

1. Modeluje wiele rzeczywistych sieci:
   - sieci P2P (np. Gnutella)
   - Internet
2. Tworzenie topologii scale-free w oparciu o model Barabási and Albert
   - tworzymy sieć dodając węzeł po węźle
   - jeżeli chcemy dodać węzeł A do sieci, to podłaczamy go do węzła B, który już jest w sieci. Węzeł B jest wybierany losowo z prawdopodobieństwem proporcjonalnym do stopnia węzła.

Wskaźnik kowergencji w zależności od topologii i rozmiaru sieci



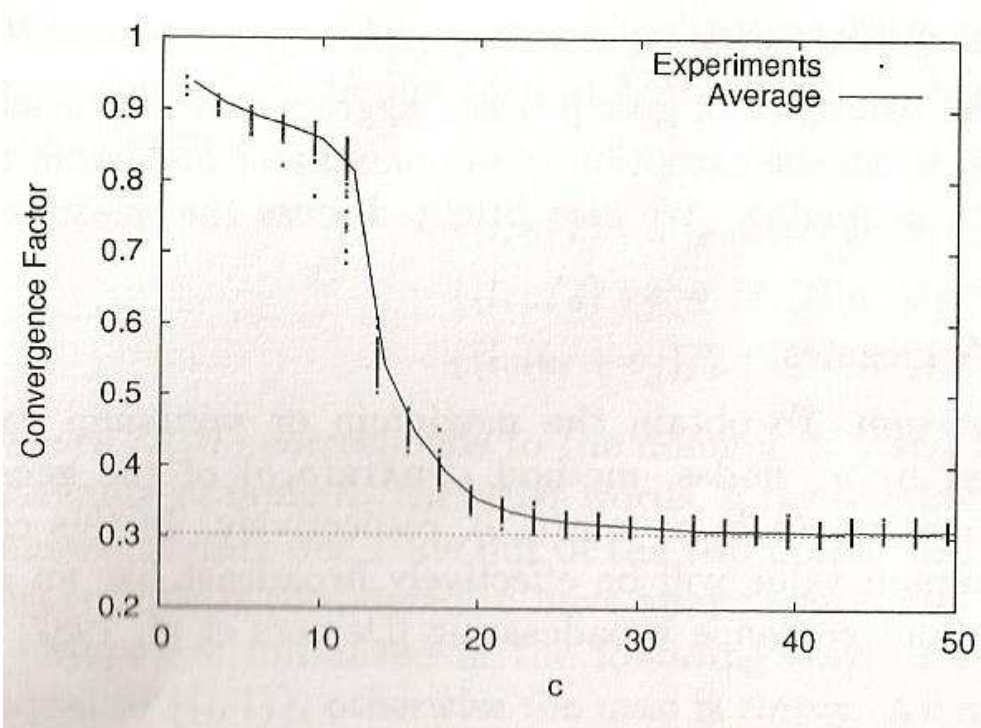
Redukcja odchylenia w zależności od topologii sieci



Dynamiczne topologie

1. Z poprzednich obserwacji wynika że wzrastająca losowość w systemie sprzyja wydajności protokołu
2. Dla systemów dynamicznych rozważmy protokół NEWSCAST
   - ciągła wymiana zbiorami sąsiadów, gdzie każdy element tego zbioru składa się z identyfikatora węzła i stempla czasowego.
   - każdy taki zbiór ma ustalony rozmiar - oznaczmy go przez c
   - po wymianie komunikatów każdy węzeł uaktualnia informacje o zbiorze sąsiadów wybierając c identyfikatorów z unii dwóch zbiorów - wybierane są te identyfikatory które mają najbardziej aktualny stempel czasowy
   - węzły które działają w sieci okresowo 'wpuszczają' swoje identyfikatory. W ten sposób stare identyfikatory są z czasem usuwane z systemu, a struktura topologiczna jest 'reperowana'
   - powyższa topologia ma małą średnicę

Wskaźnik kowergencji dla topologii NEWSCAST w zależności od parametru c



Analiza kosztu

1. Problem dobrania parametru δ
   - Liczba wymian komunikatów w δ jednostkach czasu ma rozkład $1 + \phi$ gdzie ϕ ma rozkład Poisson'a
   - średnio węzeł wymienia dwa komunikaty
2. Problem dobrania parametru γ - w oparciu o oczekiwaną dokładność
   - po γ cyklach mamy $E(\sigma_\gamma^2)/E(\sigma_0^2) = \rho^\gamma$ gdzie ρ jest wskaźnikiem kowergencji, a $E(\sigma_0^2)$ oczekiwaną wartością inicjalnej wariancji
   - jeżeli ϵ jest oczekiwaną dokładnością to γ powinna być tak dobrana aby: $\gamma \geq log_\rho \epsilon$
   - ponieważ ρ jest niezależne od N to złożoność czasowa otrzymania określonej precyzji wynosi $O(1)$

Inne funkcje agregujące

1. Wyznaczanie minimum i maximum za pomocą modyfikacji funkcji UPDATE
2. Uogólniona średnia wektora $w = (w_0 ... w_N)$ to:

$$f(w) = g^{-1}(\frac{\sum_{i=0}^{N} g(w_i)}{N})$$

- $g(x) = x$ - zwykła średnia arytmetyczna
- $g(x) = x^n$ - średnia n-tego stopnia
  - $n = -1$ - średnia harmoniczna
  - $n = 2$ - średnia kwadratowa
- $g(x) = ln(x)$ - średnia geometryczna

Wtedy UPDATE$(a, b) = g^{-1}[(g(a) + g(b))/2]$

Inne funkcje agregujące

Wariancje i inne momenty:

1. Aby obliczyć n-ty moment wektora należy zainicjować każdy węzeł n-tą potęgą inicjalnej wartości a następnie wywołać protokół obliczający średnią arytmetyczną
2. Aby obliczyć n-ty centralny moment $(avg((w - avg(w))^n))$ możemy:
   1. równolegle oblicyć wszystkie i-te momenty, gdzie $i < n$, a następnie obliczyć ich odpowiednią kombinację
   2. możemy też sekwencyjnie:
      1. obliczyć średnią
      2. w oparciu o średnią obliczyć odpowiedni moment

Zliczanie za pomocą średniej

1. Jeżeli inicialny rozkład jest taki że jeden z węzłów ma wartość 1 a pozostałe 0, to wartość średniej wynosi $1/N$
2. Jak wybrać lidera?
3. każdy węzeł losowo wybiera czy zostanie liderem czy nie
4. zamiast przekazywania jednej wartości prekazywana będzie mapa
5. kluczami w mapie będą identyfikatory węzłów a wartościami obliczone odpowiednio przybliżenia funkcji agregacyjnych

Zliczanie za pomocą średniej

Jeżeli węzły n_i oraz n_j się ze sobą komunikują to uaktualniają swoje mapy w następujący sposób:

$$M = \{(l, e/2)|e = M_i(l) \in M_i \wedge l \notin D(M_j)\} \cup$$

$$\{(l, e/2)|e = M_j(l) \in M_j \wedge l \notin D(M_i)\} \cup$$

$$\{(l, (e_i + e_j)/2)|e_i = M_i(l) \wedge e_j = D(M_j)\}$$

gdzie $D(M)$ oznacza zbiór kluczy dla mapy M, a e_i jest aktualną wartością węzła n_i Węzły są zainiciowane w następujący sposób:

- jeśli n_l jest liderem to jego mapa jest iniciowana na $\{(l, 1)\}$
- jeśli nie jest liderem to otrzymuje pustą mapę

Zliczanie za pomocą średniej

1. Nawet węzły z pustymi mapami iniciują wymianę komunikatów
2. Liczba liderów musi być ograniczona. Mechanizm który to realizuje:
   - Na początku każdy węzeł zostaje liderem z prawdopodobieństwem P_{lead}
   - W każdej epoce ustawiamy $P_{lead} = C/N'$, gdzie C to oczekiwana przez nas liczba jednoczesnych wykonań protokołu, a N' to oszacowanie otrzymane z poprzedniej epoki
   - jeżeli rozmiar sieci nie zmieni się dramatycznie w ciągu jednej epoki to rozwiązanie gwarantuje nam że liczba jednoczesnych wykonań protokołu będzie miała rozkład Poisson'a z parametrem C

Inne funkcje agregujące

1. Sumy i produkty - zagadka dla widowni :)
2. Niestety niektóre statystyki mogą być trudne do wyznaczenia (zwłaszcza statystyki oparte na obliczeniach związanych z indeksami - np. mediana)
3. Dynamiczne zapytania

Teoretyczny model dla błędów

1. Przed rozpoczęciem każdej epoki ustalona proprcja (oznaczmy ją jako P_f) węzłów ulega awarii
2. Mając N^* węzłów inicjalnie, $P_f N^*$ z nich zostaje usuniętych z sieci
3. Zakładamy że uszkodzone węzły nie odbudowują się
4. 'Najgorsze' są awarie na początku cyklu
5. Zauważamy że $\bar{w}_i$ oraz σ_i pozostają bez zmian ponieważ P_f jest symetryczne
6. wskaźnik kowergencji również pozostaje bez zmian - ponieważ nie zależy on od rozmiaru sieci
7. Interesująca jest wariancja $\bar{w}_i$ jako miara błędu

Trzeci lemat

Załóżmy że $E(\sigma_{i+1}^2) = \rho E(\sigma_i^2)$ oraz że wartości $w_{i,1}, ..., w_{i,N}$ są początkowo nieskorelowanedla $i = 0, 1, ...$ Wtedy:

$$Var(\bar{w}_i) = \frac{P_f}{N(1-P_f)} E(\sigma_0^2) \frac{1 - (\frac{\rho}{1-P_f})^i}{1 - \frac{\rho}{1-P_f}}$$

Szkic dowodu trzeciego lematu

1. Zapiszmy: $\bar{w_{i+1}} = \bar{w}_i + d_i$
2. d_i jest niezależne od $\bar{w}_i$ więc: $Var(\bar{w_{i+1}}) = Var(\bar{w}_i) + Var(d_i)$
3. Ponieważ $E(\bar{w_{i+1}}) = E(\bar{w}_i)$ to $E(d_i) = 0$

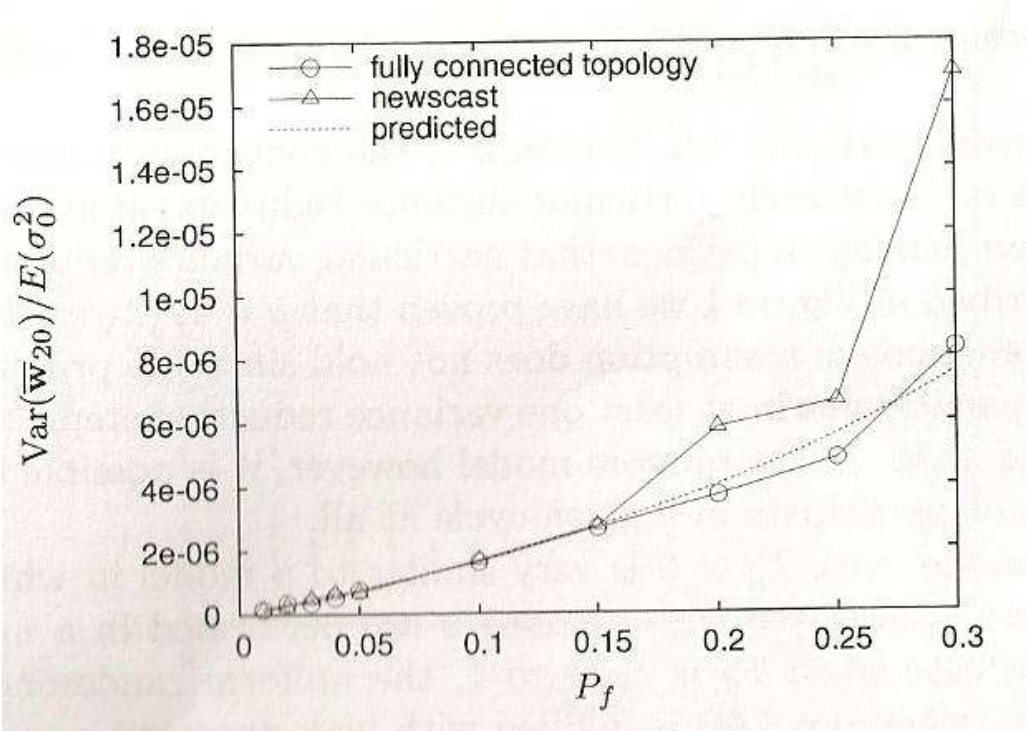
$$Var(d_i) = E((\bar{w}_i - \bar{w_{i+1}})^2) = \frac{P_f}{N_i(1-P_f)} E(\sigma_i^2) =$$

$$\frac{P_f}{1-P_f} E(\sigma_0^2) \frac{\rho^i}{N(1-P_f)^i}$$

$$Var(\bar{w}_i) = \sum_{j=0}^{i-1} Var(d_j)$$

co daje oczekiwaną formułę

Wpływ awarii węzłów na wariancję

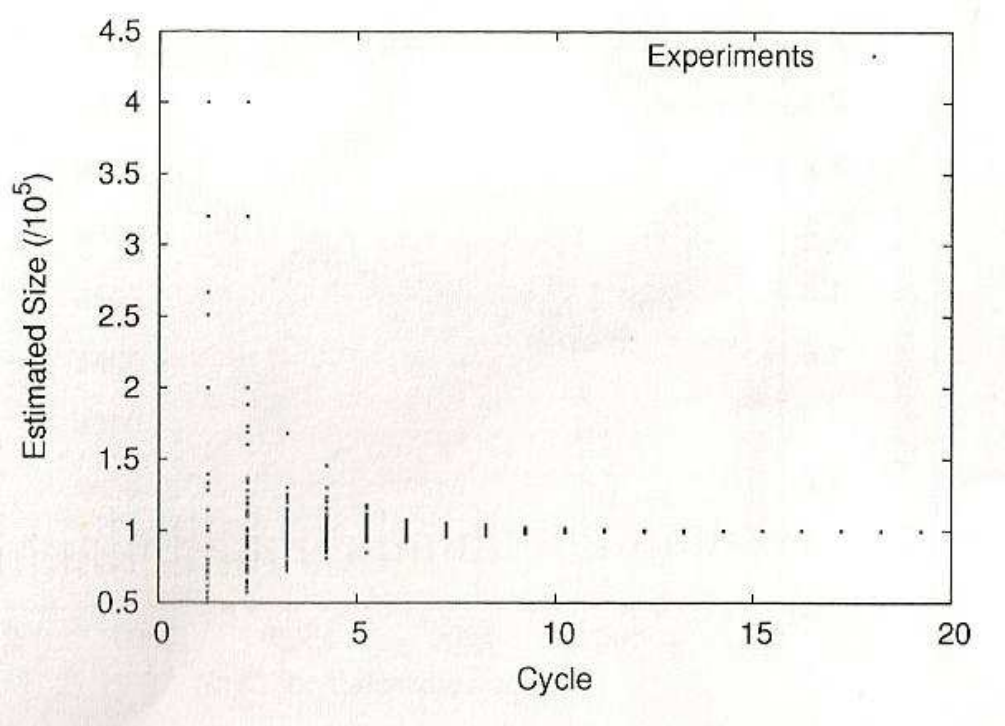


Model awarii sieci

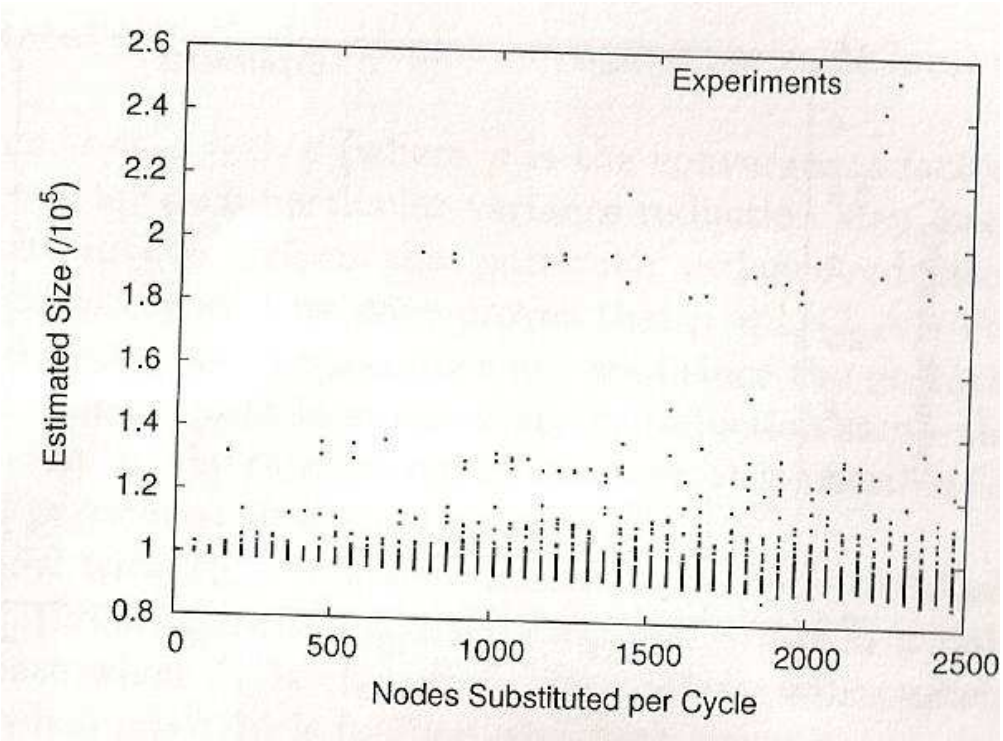
1. Zakładamy że każdy komunikat jest dostarczany prawdopodobieństwem $1 - P_d$
2. Dobrze modeluje krótkotrwałe błędy
3. Długotrwałych błędów nie da się zamodelować
4. Można policyć że dla sieci o wskaźniku kowergencji $1/2\sqrt{e}$ nowy wskaźnik kowergencji:

$$\rho_d = (\frac{1}{e})^{1-P_d}$$

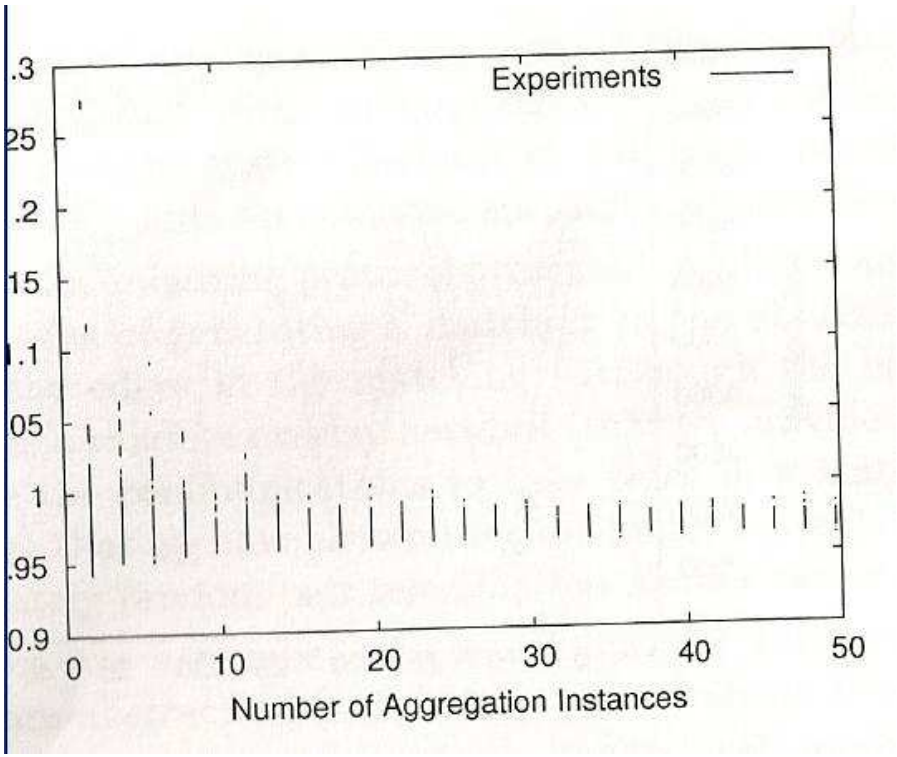
Dokładność oszacowania rozmiaru sieci w zależności od momentu 'nagłej śmierci'



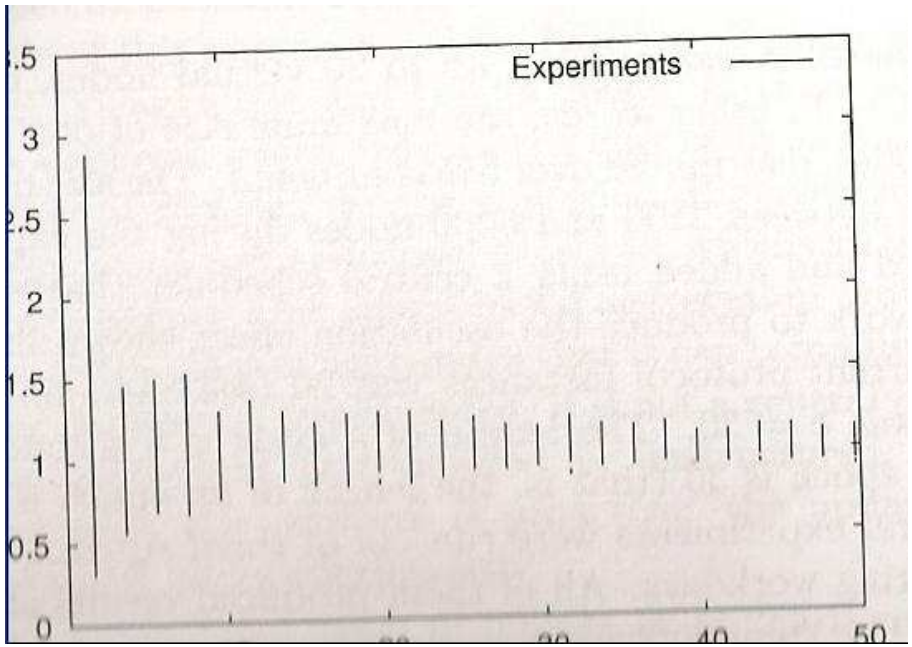
Podstawianie nowych węłów w miejsce istniejących



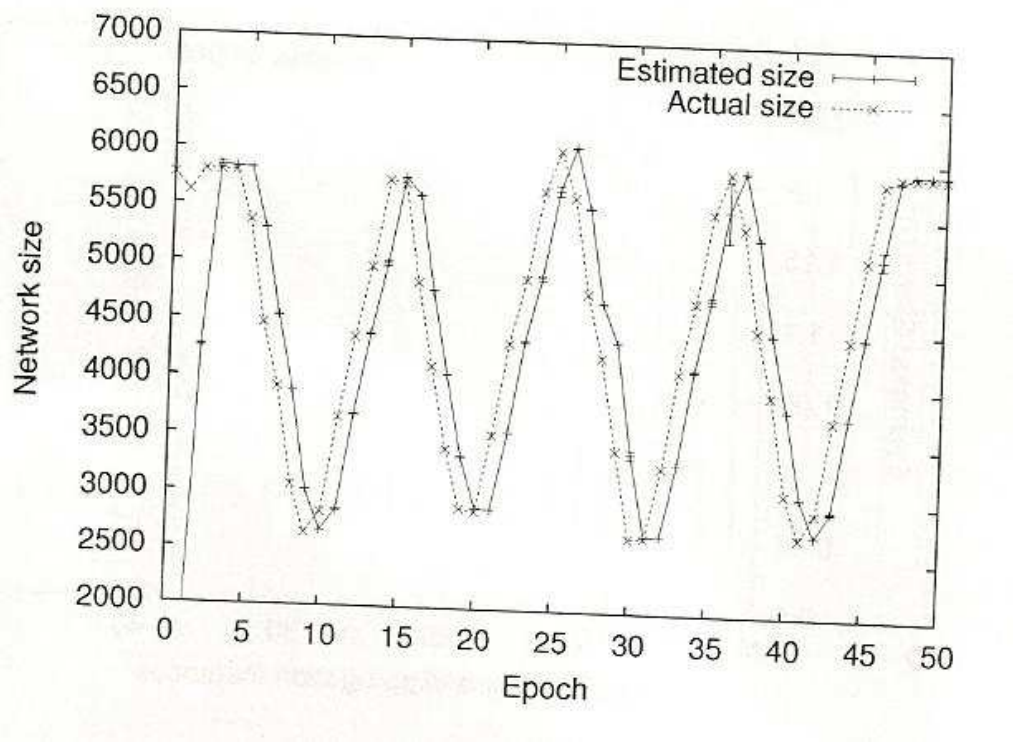
Dokładność oszacowania rozmiaru sieci w zależności od liczby instancji protokołu uruchamianych równolegle - 1000 węzłów upada na początku każdego cyklu



Dokładność oszacowania rozmiaru sieci w zależności od liczby instancji protokołu uruchamianych równolegle - 20 % komunikatów jest gubionych



Oszacowania rozmiaru sieci w poszczególnych epokach



Środowisko do rozproszonej agregacji - zastosowania

1. Wybór lidera
2. Głosowania
3. Routowanie
4. Znajdowanie zasobów
5. Balansowanie obciążenia
6. Obsługa błędów

Środowisko do rozproszonej agregacji - zastosowania

Kilka słów o środowisku Astrolabe

Kierunek rozwoju

1. Słaba spójność
2. ang. *Staleness*
3. Wysokie opóźnienie
4. Niewłaściwa hierarchia
5. Ograniczona ekspresywność
6. Ograniczona liczba atrybutów
7. Niskie bezpieczeństwo
8. Nieprzewidywalna wydajność

Jeżeli jest jeszcze ktoś na sali...

To dziekuję za uwagę :)