OPTIMALISASI GOSIP DAN KONSENSUS DENGAN ROUND ROBIN DALAM JARINGAN KOMPUTER HETEROGEN

Risa Astari Dewi – NIM : 13506064
Program Studi Teknik Informatika, Institut Teknologi Bandung
Jl. Ganesha 10, Bandung
email: if16064@students.if.itb.ac.id, risa_klik6@yahoo.com

Abstrak - Protokol gosip dilengkapi dengan metode yang dapat mendeteksi kesalahan besar, juga sistem pendistribusian yang dirancang agar tidak saling bergantung mengingat pemakaiannya dalam jaringan heterogen.. Agar efektif dengan aplikasi konfigurasi dan pemanggilan kembali, protokol ini membutuhkan mekanisme yang dapat mendeteksi kesalahan dalam sistem yang luas. Makalah ini menyajikan 3 ragam protokol gosip yang didukung dengan algoritma untuk melengkapi konsensus jaringan komputer heterogen. Protokol *round robin* berkembang pada dasar gosip acak dengan mendistribusi pesan gosip melalui keputusan guna mengoptimalisasi penggunaan bandwidth. Gosip yang tidak perlu akan tereliminasi dalam *round robin* biner, dan *round robin* dengan pengecekan berurut sangat berguna dalam menghasilkan waktu pendeteksi yang efisien tanpa membutuhkan optimasi sistem tertentu. Distribusi algoritma konsensus bekerja dengan protokol gosip ini untuk mencapai kesepakatan diantara simpul yang beroperasi dalam jaringan komputer. Protokol yang berbeda akan mensimulasikan dan mengevaluasi pada saat waktu konsensus.

***Kata kunci**: jaringan komputer, konsensus, pendeteksi kesalahan, toleransi kesalahan, protokol gosip*

1. PENDAHULAN

Dalam makalah ini penggunaan kata gosip atau bergosip untuk menandakan kegiatan dari individu yang saling berkaitan dalam jaringan komunikasi. Setiap orang memiliki satu informasi unik yang akan disampaikan pada orang lain. Dalam dunia penyiaran ditemukan masalah yang berkaitan dengan bergosip, bagaimana menemukan gosip yang efektif. Dengan kata lain penggunaan jaringan komunikasi dalam prilaku ini lebih efisien.

Pendeteksi kesalahan sangat penting bagi managemen sistem, replikasi, dan distribusi sistem lainya. Dewasa ini, layanan pendeteksi kesalahan memiliki perbandingan yang sangat buruk dengan hal yang ini dipantau. Keakuratan pendeteksi kesalahan dalam distibusi sistem *non-realtime* sangat sukar. Terdapat kemungkinan proses yang dilakukan gagal atau koneksi jaringan sangat lambat, sehingga terdapat beberapa hal yang tidak mungkin ditemukan.

Jaringan komputer di lokasi kerja (yang terhubung dengan *high-speed networking* dengan aplikasi paralel) populer dengan keefektifan dan performanya dan digunakan secara luas. Guna untuk membangun sistem yang tidak mahal dari berbagai komponen komersial, hardware berbeda dan software dari beragam vendor, penempatannya penggunaan perlu diperhatikan. Karena kerja yang dilakukan meliputi replikasi, *load balancing* dan layanan distribusi lainnya.

Toleransi kesalahan merupakan layanan yang membutuhkan teknik deteksi yang efisien yang mampu meminimalkan efek kesalahan pada aplikasi yang dijalankan dalam sistem yang besar. Namun, minimalisasi deteksi kesalahan merupakan masalah non-trivial karena proses pendeteksian tergolong kompleks. Dan metoda komunikasi grup tradisonal hanya mampu menangani sistem yang kecil pada *broadcast* sedang.

Gosip acak, yang didasarkan informasi nyata antara simpul pernah dipelajari sebagai kemungkinan deteksi kesalahan bagi sistem heterogen[2]. Makalah ini mendeskripsikan protokol konsensus distribusi melibatkan simpul dengan perjanjian tentang kesalahan simpul dinamik dalam sistem. Makalah ini menyajikan optimasi algoritma gosip dalam 3 protokol : round-robin, round-robin dengan pengecekan berurut dan *round robin* biner bersama dengan algoritma konsensus.

2. PERMASALAHAN KONSENSUS

Tujuan utama membuat jaringan dari komponen komputer yang mandiri adalah menciptakan sumber daya tergabung yang lebih baik dalam hal performa dan *availability* dari bagian lepasnya. Secara formal, permasalahan konsensus didefinisikan sebagai proses dengan perjanjian mencapai perbaikan performa dan integritas sistem diantara prosesor tanpa salah[3]. Komponen dalam sistem mungkin membutuhkan kesepakatan dalam beberapa hal seperti jam dunia yang bersesuain, resolusi dari isi pesan, protokol sistem konfigurasi dan protokol *data base*.

Beberapa situasi yang dapat membawa *deadlock* bagi sistem dalam mengimplementasikan algoritma konsensus telah teridentifikasi. Misalnya, konsensus tidak pernah tercapai jika jaringan tidak terhubung dan

tidak ada jaminan pengantaran pesan. Umumnya, protokol konsensus yang benar [3]:

1. Konsisten : semua prosesor sepakat dalam nilai yang sama
2. Valid : kesepakatan nilai merupakan masukan bagi beberapa agen
3. Wait-free : semua prosesor harus kesepakatan dalam hitungan langkah yang telah ditentukan

Metoda tradisional dalam implementasi konsensus adalah **N-Modular Redundancy** (NMR). Dengan NMR, n prosesor menampilkan kerja yang sama dan dengan mengambil keputusan terbanyak menutupi sebagian t kesalahan, dimana $n \geq 2t + 1$. Proses ini sederhana tapi mahal sejak keluaran dari sistem secara keseluruhan terbatas.

Diagnosa distribusi memerlukan diagnosa mandiri terhadap sistem keseluruhan dari prosesor tanpa salah. Tampilan diagnosa mandiri kemudian dikirim kepada pengguna memalui beberapa mekanisme: information dissemination, penyampaian pesan dan roving diagnosis.

Kuhl dan Reddy[5] menjelaskan seri dari algoritma diagnosa distribusi (SELF) dengan tiap prosesor *Pi* memantau vektor kesalahan *Fi*. Sebuah prosesor *Pi* mencoba tetangganya dan mempebaharui vektor kesalahan. Jika tetangga *Pj* ditemukan kesalahan, vektor kesalahan akan disebar pada semua tetangga lainnya. prosesor lainnya akan mencoba prosesor *Pi* dan *Pj* dan menyampaikan hasilnya pada tetangga lainnya. namun, performa dari algoritma SELF sangat bergantung pada efektivitas dari tiap percobaan. Algoritmanya memiliki batasan skala dalam mengansumsi keefisienan mekanisme *broadcast*. Kini bergosip menjadi metode yang paling menjanjikan untuk merancang layanan deteksi kesalahan pada sistem berskala besar.. Bagian selanjutnya menjelaskan algoritma konsensus gosip seperti protokol gosip pada tulisan lain.

3. DESAIN PROTOKOL

Protokol dasar yang dijelaskan sebelumnya mampu melewati beberapa batasan. Salah satunya adalah pola acak dari pesan gosip yang dapat mengakibatkan salah dalam deteksi kesalahan. Simpul mungkin akan salah mendeteksi pada simpul kedua karena tidak menerima pesan gosip yang berisi perhitungan sebelumnya. Batasa kedua untuk gosip acak adalah *bandwith* jaringan mungkin akan terbuang percuma, jika beberapa simpul mengirimkan informasi yang sama.

Dalam makalah ini akan menampilkan teknik untuk mengurangi kesalahan dari *round robin* dasar, *round robin* biner dan *round robin* dengan pengecekan berurut.

3.1 Algortima konsensus

```
1. for Node ID = p where 0 ≤ p < n do
2.    Initialize F[i] and S[j, k] to 0 where 0 ≤ i , j , k< n
3.     On the receipt of S' from Node q do // Update S when another node
4.          for (j=0, j<n, j++) // gossips its suspect matrix S'
5.                if (j ≠q) then
6.                      for (k=0, k<n, k++)
7.                            S[j, k]=S[ j, k] || S'[j, k] // Logically OR
the elements
8.                      end for // for all rows except the qth
9.                end if // row
10.               if (j=q) then
11.                     for (k=0, k<n, k++)
12.                           S[j, k]=S'[j, k] // The qth row should be
13.                     end for // replaced with the qth row of S'
14.               end if
15.         end for
16. end do
17. Every Tgossip seconds do // Update S and F before
18.   for (k=0, k<n, k++) // gossiping; Node ID = p
19.         if ( fail_check(k)= true) then // fail_check is a
20.               S[p,k]← 1 else S[p,k]← 0 // procedure to check
21.   end for // for liveliness
22.   for (k=0, k<n, k++)
23.         temp← 0
24.         for (j=0, j<n, j++)
25.               if S[j, k]=1 then temp← temp+1 // Set F[j] to 1 only if a
majority of
26.         end for // the nodes suspect it to have died
```

```
27.         if temp > n/2 then F[k] ← 1 else F[k] ← 0 // otherwise reset to
0
28.    end for
29.    for (k=0, k<n, k++)
30.         temp← 0
31.         for (j=0, j<n, j++) // Check if all the fault-free
32.              if S[j, k] || F[j]=1 then temp← temp+1 // members have
detected
33.         end for // the failed node
34.         If temp=n then consensus_reached(j)
35.    end for
36.    Append S to message when gossip is sent
37.    end do
38. end for
```

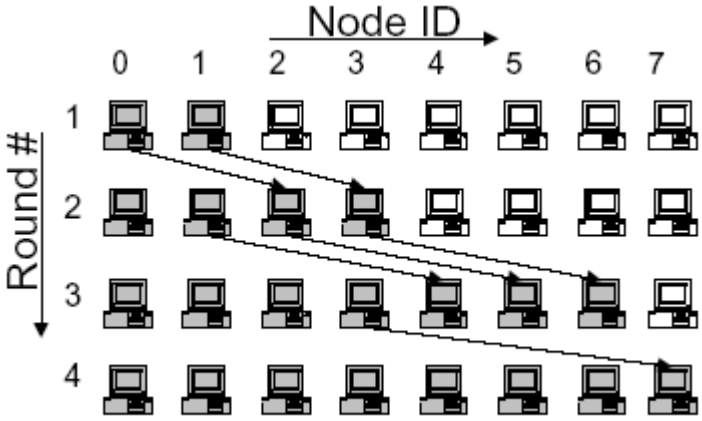
3.2 Protokol *round robin* (RR) dasar

Secara alami protokol ini acak. Artinya protokol memberikan waktu bagi simpul untuk tidak mengolah data dari pesan gosip berikutnya. Hal ini dapat memungkinkan simpul tidak menerima pesan gosip untuk waktu yang lama. Kepadatan gosip akan timbul jika menambahkan faktor penentu. Pada RR, gosip akan memiliki putaran tertentu setiap T_{gosip} detik. Untuk tiap putaran, tiap simpul menerima dam mengirim sebuah pesan gosip. Simpul tujuan pengiriman pesan mengikuti persamaan berikut, untuk r adalah jumlah putaran

Tujuan = sumber + r , $1 \leq r < n$

Untuk n jumlah simpul.

Misalkan banyaknya simpul adalah n = 8, maka gambaran putaran yang terjadi agar semua pesan tersampaikan sebagai berikut.



Gambar 1 Distribusi gosip untuk protokol RR

Dari gambar diatas, menunjukkan bahwa dibutuhkan paling tidak 4 putaran agar semua simpul sadar terhadap sinyal dari simpul 0. Hasilnya, andaikan $T_{cleanup}$ dengan nilai lebih kecil dari $4.T_{gosip}$ untuk kelompok 8 simpul akan menghasilkan kesalahan dalam pendeteksian dan konsensus menjadi tidak mungkin. Pengukuran serupa bisa juga dilakukan pada ukuran sistem lainnya. Tabel 1 mengilustrasikan hubungan antara jumlah putaran dan jumlah simpul dalam sistem *round robin*.

Tabel 1. Ukuran maksimal sistem dan jumlah putaran

Jumlah putaran	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Maksimal simpul	2	4	7	11	16	22	29	37	46	56

Secara umum, konsensus untuk sistem *round robin* dengan n simpul mungkin terjadi jika dan hanya jika

$$T_{cleanup} \geq \alpha . T_{gossip}, \text{dimana } \frac{\alpha(\alpha-1)}{2} + 1 \leq n \leq \frac{\alpha(\alpha+1)}{2} + 1$$

Nilai diatas merupakan hasil kalkulasi dari kasus terburuk parameter $T_{cleanup}$ karena semua simpul tidak bersesuain dengan baik. Saat simpul gosip telah mengrimkan pesan, semua simpul tidak bergosip di waktu yang sama dalam periode yang diberikan tiap putaran. Maka, hal ini memungkinkan untuk satu simpul untuk menerima pesan gosip dari simpul yang lain (untuk kedua kalinya) sebelum mengirimkan

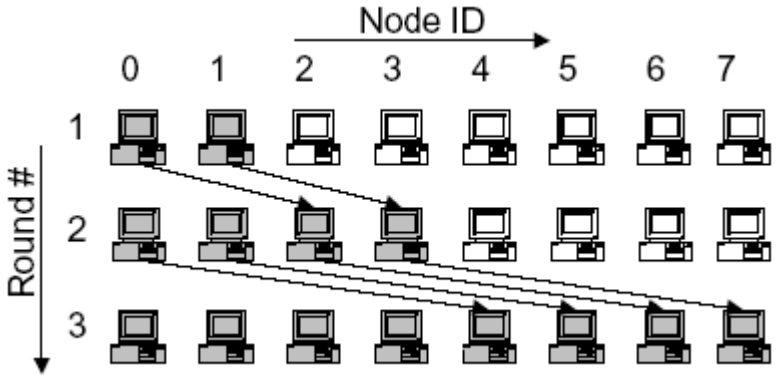
pesannya sendiri. Jika data dari simpul kedua tidak bersesuaian dengan simpul pertama, mungkin terjadi bahwa simpul pertama melemparkan gosip simpul kedua kepada simpul yang lain di putaran yang sama.

Pada gambar juga dapat dilihat komunikasi yang berlebih (redundant). Pada putaran ketiga, simpul 0 mengirimkan pesan lain ke simpul 3, dimana akan lebih optimal jika mengirimkan langsung ke simpul 4. Hal ini dapat dieliminasi dalam protokol *round robin* biner.

3.3 Protokol *round robin* biner (BRR)

Protokol BRR mencoba mengoptimalkan penggunaan *bandwith* gosip untuk menghilangkan gosip berlebih. Algortima diimplementasikan dengan memodifikasi protokol RR, dimana r adalah jumlah putaran :

Tujuan = sumber + 2r-1, $1 \leq r < \log 2$



Gambar 2 Distribusi gosip untuk protokol BRR

Dengan menggunakan sistem 8 simpul seperti contoh sebelumnya, waktu yang dibuthkan untuk semua simpul menerima pesan dapat dipersempit. Dan gosip yang berlebih tidak muncul pada protokol ini. Sehingga, konsensus untuk kasus ini sebanding $T_{cleanup} \geq 3.T_{gosip}$. sekali lagi, nilai tersebut masih hasil kalkulasi dari kasus terburuk dari parameter karena semua simpul tidak bersesuain dengan baik. Secara umum, konsensus untuk n simpul sistem BRR dapat terjadi jika dan hanya jika :

$$T_{cleanup} \geq (\log_2 n).T_{gossip}$$

Tabel 2 Perbandingan putaran dan ukuran sistem

Simpul	*2*	*4*	*8*	*16*	*32*	*64*	*128*	*256*	*512*	*1024*
Putaran RR	1	2	4	5	8	11	16	23	32	45
Putaran BRR	1	2	3	4	5	6	7	8	9	10

Dapat dilihat dari tabel perbandingan jumlah putaran antara RR dan BRR. Misalkan untuk sistem 1024 simpul membutuhkan 10 putaran untuk BRR sementara *round robin* memerlukan 45 putaran. Akan tetapi masih diperlukan pencarian modifikasi algoritma kembali untuk dapat mengimplementasikan algoritma dengan jumlah simpul bebas.

3.4 *Round robin* dengan pengecekan berurut (RRSC)

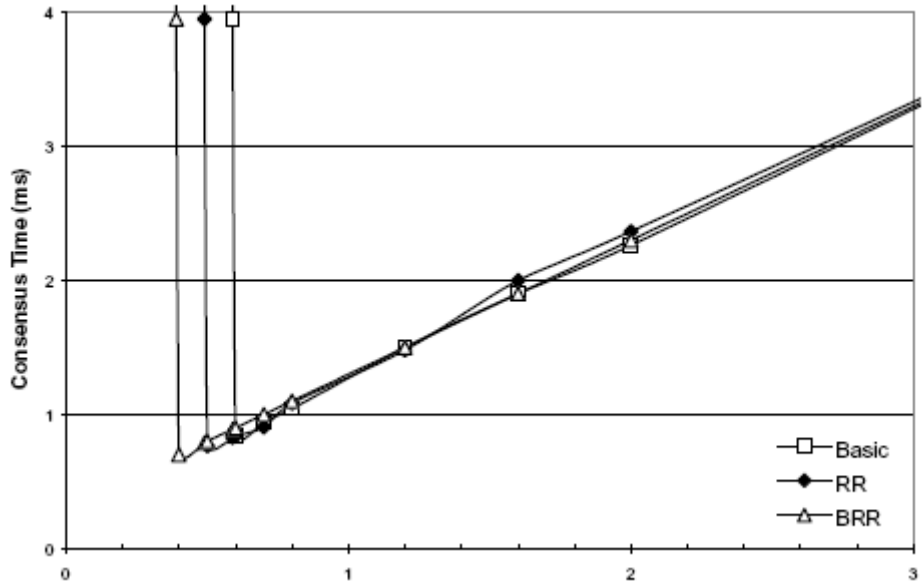
Dengan penambahan protokol RR dengan pengaruh komunikasi berurut menjadi protokol gosip baru, *round robin* dengan pengecekan berurut. Modifikasi yang diperlukan :

1. Simpul dianggap salah jika pesan gosip tidak diterima dalam selang putaran tertentu
2. Simpul selanjutnya akan ditandai salah jika dan hanya jika simpul merupakan sumber dari penerima daftar gosip.

Misalkan sebuah sistem dengan 16 simpul. Pada putaran pertama, kedua dan ketiga, node 1 akan menerima pesan dari simpul 0, 15 dan 14. Jika simpul 0 mati pada putaran pertama, ia akan gagal mengirim pesan ke simpul 1. Pada putaran kedua, simpul 1 menerima dari 15 pengecekan berurut akan menemukan kegagalan simpul 0 dalam mengirimkan pesan pada putaran sebelumnya. Maka simpul 0 akan dicoret dari daftar simpul 1. Diputaran ketiga, simpul 14 tidak mendeteksi kegagalan simpul 0 dan tetap memasukkannya dalam daftar dan mengirimkan pada simpul 1. Saat kedua daftar digabungkan, simpul 0 akan tetap hilang dari daftar simpul 1.

4. SIMULASI PROTOKOL DENGAN KONSENSUS

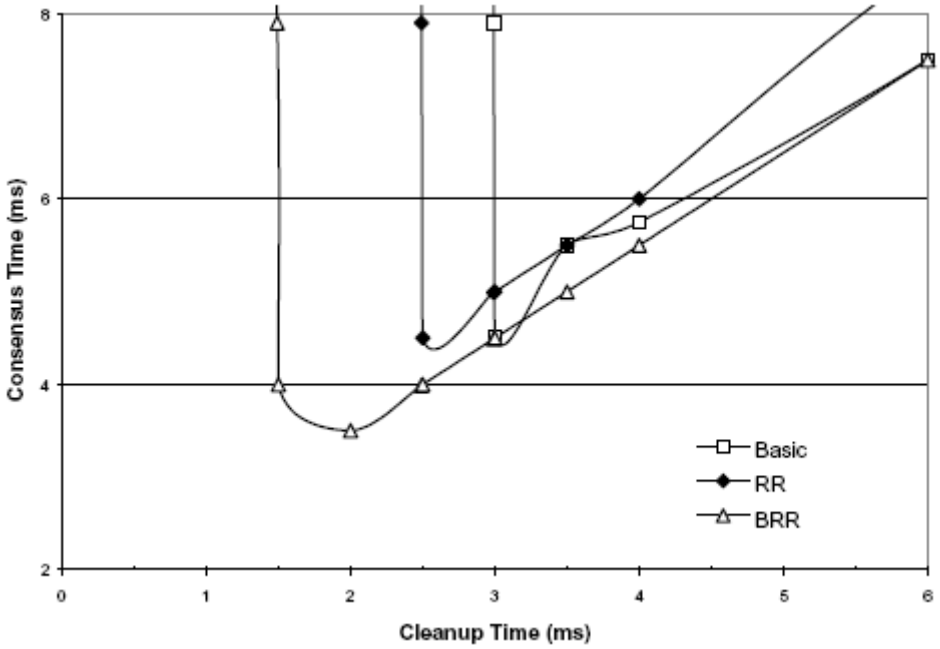
Untuk membandingkan ketiga protokol *round robin*, simulasi akan menggunakan sistem 16 simpul dengan Tgosip = 0.1ms dan masukan kesalahan tunggal.



Gambar 3 Waktu konsensus dengan waktu pembersihan untuk sistem 16simpul

Gambar diatas membandingkan waktu konsensus protokol dasar, RR dan BRR. Ditunjukkan bahwa konsensus tidak tercapai dengan protokol dasar untuk nilai $T_{cleanup} < 6.T_{gossip}$. Nilai minimum $T_{cleanup}$ yang lebih rendah mungkin dengan protokol RR karena mekanisme ini mampu mengurangi deteksi yang salah. Dari data tabel 1, nilai minimum $T_{cleanup}$ RR adalah $5.T_{gosip}$ dan BRR adalah $4.T_{gossip}$ untuk sistem 16simpul.

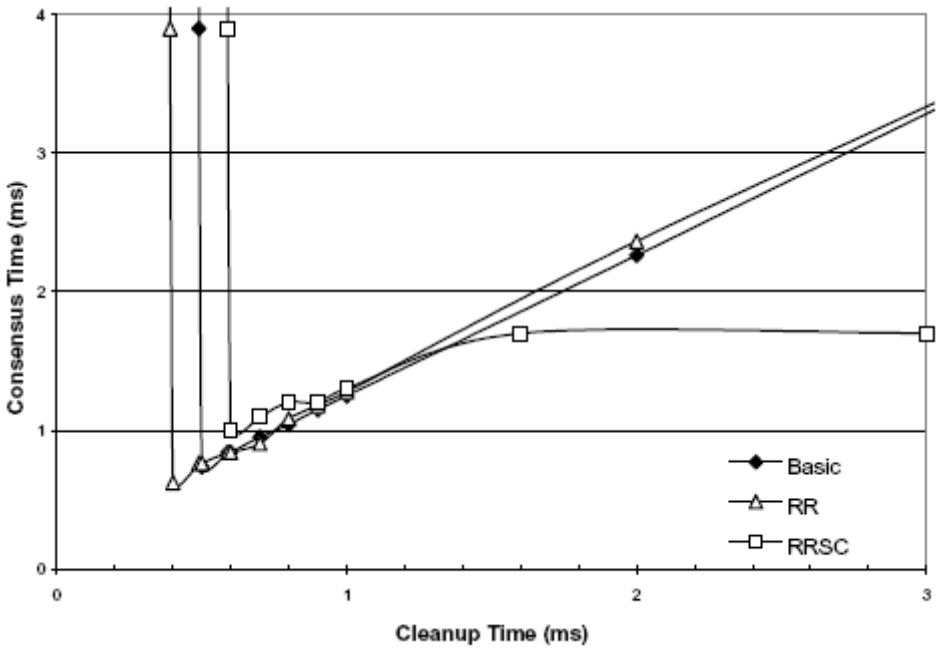
Seperti pada subbagian sebelumnya, putaran timpang dapat mengakibatkan perubahan hasil yang diluar dugaan. Dari hasil simulasi, terdapat nilai acak dari kegagalan bersesuain ε (dimana $0 \leq \varepsilon \leq T_{gosip}$) mempengaruhi nilai minimum $T_{cleanup}$ menjadi lebih kecil dari analisa. Gambar 5 menampilkan hasil percobaan sistem 16 simpul untuk T_{gosip}= 0.5m. Nilai minimum $T_{cleanup}$ protokol BRR menjadi $3.T_{gosip}$.



Gambar 4 Waktu konsensus dengan waktu pembersihan (Tgosip = 0.5ms)

Pada gambar 5 ditampilkan perbandingan antara protokol RRSC dengan RR dan protokol dasar. Gambar mengiindikasikan nilai minimum RRSC lebih besar dari RR. Batasan ini bergantung pada penyelesain deteksi hanya bisa dilakukan saat simpul menerima gosip langsung dari simpul yang salah. Maka dari itu salah deteksi tidak bisa diselesaikan untuk $T_{cleanup}$ yang rendah jika simpul tidak tergabung dalam komunikasi langsung dengan simpul yang salah. Bagaimanapun juga, hasil pada gambar

menunjukkan kegunaan dari performa pengecekan berurut. Protokol RRSC membutuhkan (n-1) putaran gosip untuk mencapai konsensus pada simpul salah dengan menggunakan pengecekan berurut dimana n adalah ukuran sistem. Maka dari itu, untuk Tcleanup > 15.Tgosip, kurva RRSC akan tidak berganung pada batas atas waktu konsensus bahkan jika sistem diatur pada nilai $T_{cleanup}$ yang optimal.



Gambar 5 Waktu konsensus dengan waktu pembersihan (RR dan RRSC)

5. KESIMPULAN

Makalah ini menampilkan beberapa protokol baru dalam mengatasi kekurangan protokol dasar untuk pendeteksi kesalahan pengiriman gosip pada jaringan komputer heterogen. Dari simulasi yang telah ditampilkan, ketiga protokol ini mampu mengurangi kesalahan deteksi dan lebih efisien. Terutama dalam mencapai nilai $T_{cleanup}$ yang lebih rendah dari T_{gosip} yang ditentukan.

Tiga protokol tersebut adalah :

1. Round robin yang mengirimkan pesan gossip berdasarkan keputusan
2. Round robin biner yang mengeliminasi gossip berlebih dan dapat mengoptimalisasi penggunaan *bandwith* gosip.
3. Round robin dengan pengecekan berurut mampu mengurangi salah deteksi dengan mengeliminsasi simpul yang dideteksi gagal.

Untuk bahan pembelajaran ke depan, diharapkan adanya penelitian lanjut dengan mengkombinasikan protokol dasar, *round robin*, biner, pengecekan berurut dan metode potensial lainnya untuk lebih mengoptimalkan pengguna *bandwith* gosip. Juga percobaan lebih lanjut untuk jaringan komputer skala besar, guna mengimplementasikan protokol ini dalam jaringan kerja dan sistem operasi

DAFTAR REFERENSI

[1] Brenda Baker and Robert Shostak. Gossips and telephones. Discrete Mathematics, 2(3):191-193, June 1972.

[2] Boden, N., Cohen, D., Felderman, R., Kulawik, A, Seitz, C., Seizovic, J. and Su, W. Myrinet: A Gigabit-per-Second Local Area Network. *IEEE Micro*. Vol 15, No 1 , 26-36, 1995.

[3] Barborak, M., Dahbura, A. and Malek, M. The Consensus problem in Fault-Tolerant Computing. *ACM Computing Surveys*. Vol 25, No 2, 171-220, 1993.

[4] Robbert van Renesse, Yaron Minsky, and Mark Hayden. A Gossip-Style Failure Detection Service. Dept. of Computer Science, Cornell University 2001

[5] Kuhl, J. and Reddy, S. Fault-Diagnosis in Fully-Distributed Systems. *Proceedings of the 11th International IEEE Symposium on Fault-Tolerant Computing*. 100-105, 1981

[6] Aarts, Ronald M. "Gossiping." From *MathWorld*--A Wolfram Web Resource, created by Eric W. Weisstein http://mathworld.wolfram.com/Gossiping.html