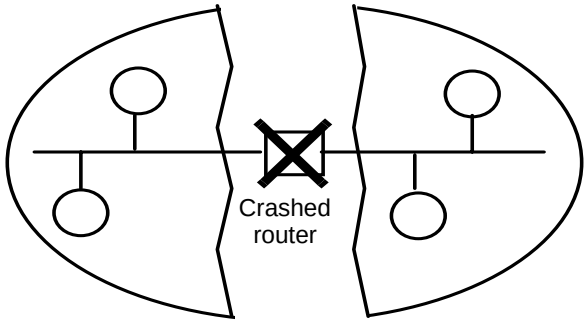
Coordination and Agreement

Ferdian Pramudya P (32582)
Agung kaharesa W (32649)
Jurusan Teknik Elektro FT UGM,
Yogyakarta

I. PENDAHULUAN

Coordination and Agreement, dalam makalah ini kami akan menjelaskan sekumpulan algoritma yang tujuannya bermacam-macam namun men-*share* tujuannya, sebagai dasar dalam sistem terdistribusi : berupa sekumpulan proses untuk mengkoordinasikan tindakan atau menyetujui satu atau beberapa nilai. Contohnya pada kasus mesin seperti pesawat ruang angkasa. Hal itu perlu dilakukan, komputer mengendalikannya agar setuju pada kondisi tertentu seperti apakah misi dari pesawat luar angkasa dilanjutkan atau telah selesai. Komputer tersebut harus mengkoordinasikan tindakannya secara tepat untuk berbagi sumber.[2]

Hal yang penting dalam *Coordination and* Agreement adalah apakah sistem terdistribusi asinkron atau sinkron. Algoritma –algoritma yang digunakan juga harus mempertimbangkan kegagalan yang terjadi, dan bagaimana caranya untuk berhubungan satu sama lain ketika sedang mendesaian algoritma. Selanjutnya di makalah ini juga akan dijelaskan mengenai masalah dalam mendistribusikan *mutual exclusion, election, multicast communication,* dan mengenai masalah dalam persetujuan(*agreement)*.



Gambar 1. Crash Router,Network Partition[2]

II. PEMBAHASAN

A. Asumsi Kegagalan(Failure Assumption) dan Pendeteksi Kegagalan (Failure Detectors)

Sebelum membicarakan tentang masalah dalam *Coordination and Agreement* , kami akan terlebih dahulu menjelaskan tentang asumsi kegagalan dan keadaan yang berguna dalam mendeteksi kegagalan pada sistem terdistribusi.

Salah satu masalah dalam pembuatan algoritma yang bisa menyebabkan terjadinya *crash* adalah bila terjadi proses *crash* itu sendiri. *Failure Detector* [Chandra and Toueg

1996, Stelling et al. 1998] adalah suatu layanan yang mempertanyakan apakah proses yang terjadi gagal atau tidak. *Failure Detector* kadang tidak tepat. Kebanyakan kegagalannya adalah pada *failure detector* yang tidak handal, yang bisa menghasilkan hal yang diduga maupun tidak terduga. Kedua hasil tersebut merupakan petunjuk yang menggambarkan keakuratan atau ketidakakuratan apakah suatu proses benar-benar gagal. Hasil yang tidak terduga berarti pendeteksi telah menerima fakta-fakta bahwa proses tidak gagal. Kemudian hal yang diduga berarti *failure detector* memiliki beberapa indikasi yang menyatakan bahwa proses itu gagal. *Failure Detector* yang handal merupakan salah satu pendeteksi kegagalan yang paling akurat. Ia menjawab proses pertanyaan salah satu dari yang tidak terduga, seperti sebelumnya, mendapat petunjuk atau gagal. *Failure detector* kadang-kadang dapat memberikan respon yang berbeda jika kondisi komunikasi bermacam-macam dari proses ke proses.[2]

Failure detector dapat membantu kita untuk mengetahui sebab-sebab terjadinya kegagalan pada sistem terdistribusi dan agar kita mengetahui solusi untuk memecahkan masalah dalam pengkordinasian proses bila terjadi kegagalan.

B. Distributed Mutual Exclusion

Jika sekumpulan proses bersama-sama menggunakan sumber, *mutual exclusion* dibutuhkan untuk mencegah gangguan dan menjaga konsistensi ketika sedang mengakses sumber. Ini adalah masalah yang kritis. Pada sistem terdistribusi, tidak ada satupun variabel yang dipakai secara bersama-sama maupun fasilitas yang diberikan melalui lokal kernel tunggal bisa digunakan untuk memecahkan hal tersebut.[2]

Syarat yang dibutuhkan untuk algoritma *mutual Exclusion*

1. Keamanan(*safety)*, hanya satu proses yang dieksekusi pada satu waktu,
2. *Liveness,* meminta untuk masuk dan keluar dari *critical section* yang secepatnya menggantikan.
   Kondisi *liveness* menyatakan kebebasan dari keduanya , *deadlock* dan *starvation.*
   Deadlock akan melibatkan dua atau lebih program yang dalam keadaan saling tunggu. Starvation adalah penundaan suatu proses yang telah memintanya.
3. *Ordering,* akses diterima dan terjadi sebelum relasi, dengan begitu proses dapat berkoordinasi. Proses dapat menunngu akses dan berkomunikasi dengan proses lainnya.

Performa kriteria :

1. *Bandwith* digunakan sebanding dengan jumlah pesan yang dikirim pada setiap masuk atau keluar dari C.S (*Critical Section*).
2. *Client Delay* terjadi jika proses memasuki atau keluar dari C.S.
3. *Throughput,* efek dari algoritma atas *throughput* pada sistem.

C. Elections

Algoritma yang digunakan untuk memilih proses unik untuk memainkan fakta-fakta disebut *election alghoritm. Election alghoritm* dibutuhkan untuk memilih proses yang akan memainkan peran pada server.Hal ini diperlukan bahwa setiap proses setuju pada pilihannya. Setelah itu, jika proses dapat memainkan peran server untuk berhenti maka pemilihan lain dibutuhkan untuk memilih pengganti.

D. Multicast

Multicast atau ***multicasting*** adalah sebuah teknik di mana sebuah data dikirimkan melalui jaringan ke sekumpulan komputer yang tergabung ke dalam sebuah grup tertentu, yang disebut sebagai ***multicast group***. Multicasting merupakan sebuah cara pentransmisian data secara *connectionless* (komunikasi dapat terjadi tanpa adanya negosiasi pembuatan koneksi), dan klien dapat menerima transmisi *multicast* dengan mencari di mana lokasinya, seperti halnya ketika kita membuka sebuah stasiun radio untuk mendengarkan siaran radio. Multicast sebenarnya merupakan mekanisme komunikasi *one-to-many*, atau *point-to-multipoint*, dan berbeda dengan cara transmisi *unicast*.

Sebuah *multicast group* memiliki sebuah alamat *multicast*, yaitu kelas D dalam alamat IP versi 4 atau memang alamat *multicast* dalam alamat IP versi 6. Pada kelas D alamat IP versi 4, alamat yang direservasikan untuk sebuah *multicast* group adalah `224.0.0.0` hingga `239.255.255.255`.[10]

Pengiriman *multicast dirancang untuk* menghemat bandwith dari jaringan IPv4 . Hal itu dapat mengurangi lalu lintas dengan memungkinkan *host* untuk mengirim satu paket ke beberapa *host* lain yang dipilih. Untuk mencapai tujuan beberapa host menggunakan komunikasi unicast, sumber host akan perlu mengirim paket yang ditujukan kepada individu masing-masing host. Dengan multicast, sumber host dapat mengirim satu paket yang dapat mencapai sangai banyak host.

Beberapa contoh transmisi multicast adalah:

1.Video and audio broadsatc
2.Routing information exchange by routing protocols
3.Distribution of software
4.News feeds ideo and audio broadcasts
5.Routing information exchange by routing protocols
6.Distribution of software
7.News feeds. [10]

MultticastKlien

Host yang ingin menerima data multicast tertentu disebut multicast klien. Multicast klien yang menggunakan layanan ini didahuluii oleh sebuah program klien yang berlangganan ke grup multicast. Tiap *multicast grup* diwakili oleh satu tujuan alamat IPv4 multicast. Ketika sebuah host IPv4 berlangganan multicast grup, *host* proses paket ini dialamatkan ke alamat multicast serta paket-paket yang dialamatkan ke alamat unicast juga dialokasikan secara unik.

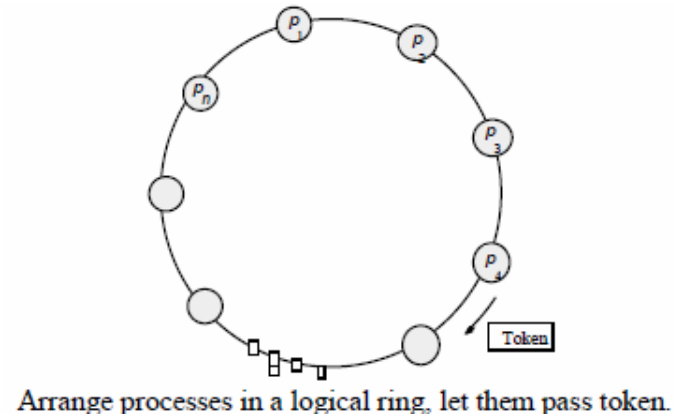
Multicast Forwarding Algorithm

Berfungsi untuk menentukan jalur yang akan dilewati..Jalur yang ditentukan harus dapat menjangkau seluruh members.Beberapa Algoritma yang digunakan oleh multicast berfungsi :

1. mengarahkan data hanya untuk members
2. mengoptimalkan jalur yang dilewati
3. menyediakan tempat dan mainten bagi members
4. secara fleksibel menentukan jalur

E. Beberapa algoritma yang terkait pada Coordination & Agreement

1) Ring Based Alghoritm



Gambar 2 Ring-based Alghoritm[1]

Ring-based algorithm
Proses:

1.) Meneruskan token sekeliling ring, dalam satu arah
2.) Jika tidak memerlukan akses *critical section*, lewat menuju tetangga sebelahnya
3.) Sebaliknya, menunggu token dan menahannya ketika dalam *Critical section*

Cara kerja:

1.) Penerusan menggunakan network bandwith
2.) Delay untuk masuk tergantung ukuran ring
3.) Sepintas order request tidak memberikan *respect*

Ricart-Agrawala algorithm
Bedasarkan pada komunikasi multicast

1.) N *inter-connected* proses asinkron, masing-masing dengan id yang unik
2.) Proses request multicast untuk memasuki *critical section*
3.) Masukan diterima
    a.) Ketika semua proses lain menjawab
    b.) Request simultan dipisahkan dengan *timestamp*

Cara kerja:

1.) Meyakinkan properti yang kuat (MC3)
2.) Ketika hardware mensuport multicast, hanya satu pesan untuk masuk

Ricart-Agrawala algorithm

Dalam inisialisasi
state := RELEASED;

Untuk masuk ke Critical Section
state := WANTED;

Multicast request ke seluruh proses; pemrosesan request request ditunda disini
T := request's timestamp;

Menunggu hingga (jumlah balasan yang diterima= (*N* – 1));
state := HELD;

dalam penerimaan request *<Ti, pi> at pj (i ≠ j)*
if (*state* = HELD) or ((*state* = WANTED) *and* ((*T*, *pj*) < (*Ti*, *pi*))

then
antrian request dari pi tanpa balasan:

else
membalas dengan cepat ke *pi*;

end if
untuk keluar dari ***critical section***

state := RELEASED;
menjawab seluruh request antrian.

2.) Koordinasi

Algoritma koordinasi
Yang merupakan dasar dalam sistem terdistribusi adalah:

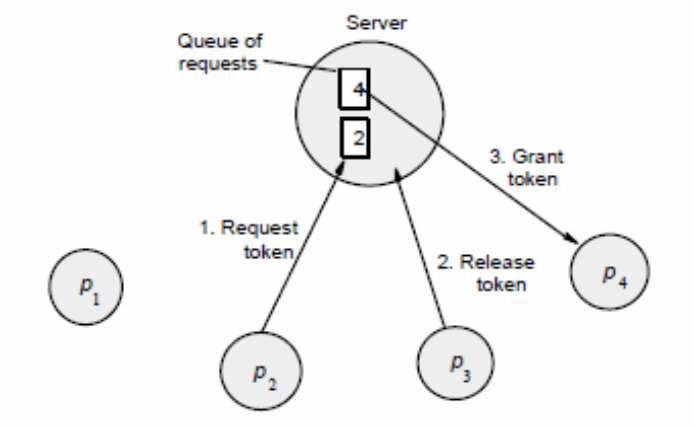
1.) Untuk *resource sharing*: *Concurent* update dari:
    a.) Simpanan dalam database (record locking)
    b.) File (File lock)
2.) Untuk meyetujui suatu tindakan
    a.) Untuk menjalankan/mengakhiri transaksi database
    b.) Menyetujui pembacaan sekelompok sensor
3.) Untuk melakukan *re-assign* secara dinamis aturan :
    a.) Memilih time server utama setelah *crash*
    b.) Memilih koordinator setelah konfigurasi ulang dari network

Masalah dalam Koordinasi

1.) Mutual exclusion

2.) Pemimpin pemiliha n
    a.) Setelah kegagalan crash terjadi
    b.) Setelah rekonfigurasi jaringan
3.) Persetujuan umum
    a.) Sama halnya dengan serangan koordinasi
    b.) Beberapa berdasarkan pada komunikasi multicast
    c.) Varian bergantung pada tipe kegagalan, network, dan lainnya.

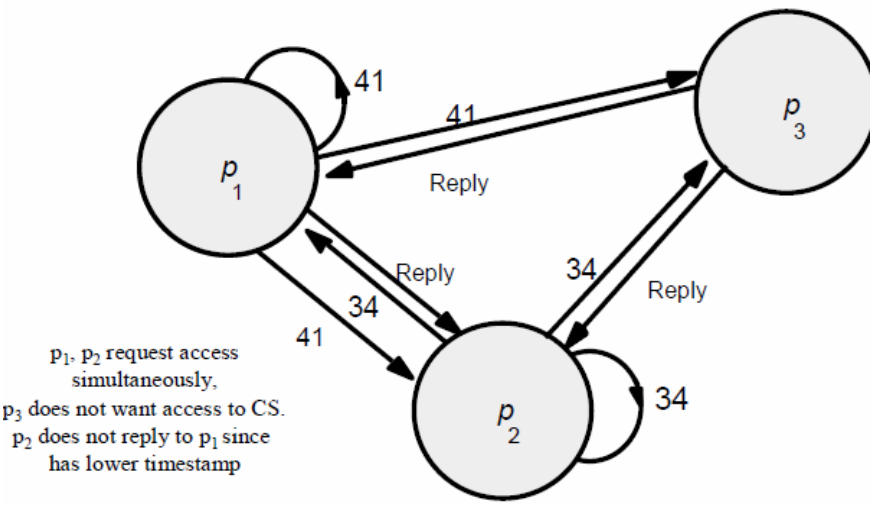
3.) Centralized Mutual Exclusion



Gambar 3. Centralised Mutual Exclusion[1]

Centralized Service
Server tunggal mengimplementasikan bukti imajiner:
proses pemegangan bukti dapat di dalam *critical section*
Server menerima request sebagai bukti
Membalas penerimaan akses ketika *Critical section* bebas, sebaliknya, request diantrikan
Ketika proses menerima bukti (token), request yang lebih lama dalam antrian diterima Hal itu berjalan selama tidak mempengaruhi order dari request.

4.) Multicast Mutual Exclusion



Gambar 4Multicast Mutual Exclusion [1]

\

Mutual exclusion summary

Performa

a.) Sebuah "request-reply" cukup untuk masuk
b.) Secara relatif, pemakaian network bandwithnya tinggi
c.) Delay klien tergantung pada frekuensi dari akses dan ukuran network

Toleransi kesalahan

a.) Biasanya diasumsikan sebagai jaringan yang reliable
b.) Beberapa dapat diadaptasikan untuk deal dengan crash

Leader election algorithms

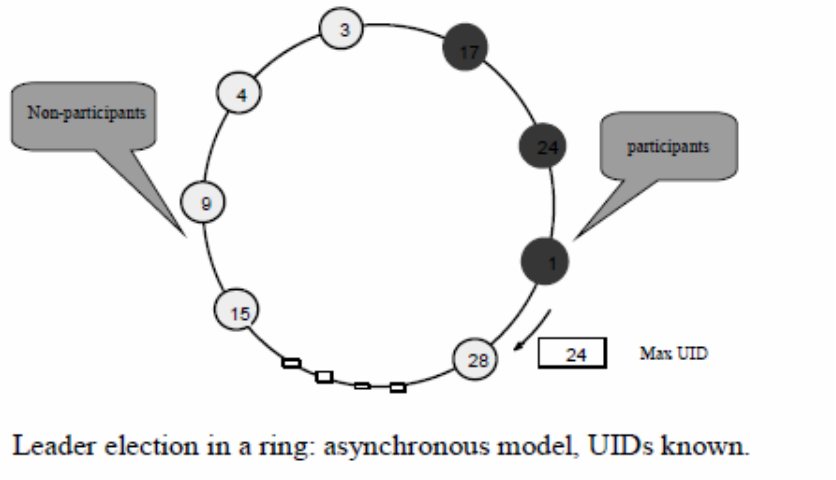
Masalah

a.) N Proses
b.) Untuk penyederhanaan asumsi tidak ada crash
c.) Harus memilih master koordinator yang unik antara proses
d.) Pemilihan dipanggil setelah kegagalan terjadi
e.) Satu atau lebih proses dapat memanggil pemilihan bersamaan

Kebutuhan

a.) Setiap proses mengenal P, identifikasi dari *leader*, dimana P adalah proses id yang unik (biasanya maksimum) atau belum didefinisikan
b.) Seluruh proses berpartisipasi dan secepatnya menemukan identitas dari *leader* (tidak bisa tak terdefinisi).

5.) Chang&Robert alghiritm



Gambar 5. Chang&Robert alghiritm [1]

Chang&Roberts algorithm
Asumsi
Ring tanpa arah tertentu, asinkron, setiap proses memiliki UID
Pemilihan
Diawali setiap proses tidak berpartisipasi
Menentukan Leader – inisiator menjadi partisipan dan melewati UID nya ke tetangga, ketika bukan partisipan menerima *election message*, meneruskan maksimum nya dan penerimaan UID, dan menjadi partisipan.
Memberitahukan pemenang – Ketika partisipan menerima *election message*, dengan UIDnya, menjadi *leader* dan non-partisipan, dan meneruskan UID dalam pesan pemilihan (*elected message*)

Bagaimana kerjanya?

a.) Jika UID, maka identitas *leader* unik
b.) Dua pergantian disekitar ring: election, pengunguman *winner*
c.) Jika satu proses memulai pemilihan (*election*)

Namun, tidak mentoleransi kegagalan (memerlukan detektor kegagalan yang handal), berjalan jika lebih dari satu proses secara bersamaan memulai pemilihan (*election*).

6.) Richart – Agrawala Alghoritm

Ricart&Agrawala Algoritma
1.) Berbasis pada komunikasi *multicast*
– proses inter-connected N yang asinkron dengan masing-masingnya
2.) Identitas yang unik
3.) Lamport'S logical clock
– proses multicast untuk meminta masuk:
4.).timestamped dengan waktu Lamport'S dan proses identitas

– masuk agar diterima

5.) ketika semua proses yang lain menjawab permintaan bersama memecahkan dengan timestamp

Bagaimana kerjanya?

a.) membuat properti yang lebih kuat (MC3)

b.) jika perangkat keras mendukung untuk multicast, hanya satu pesan yang masuk Ricart&Agrawala Algoritma Pada status initialisasi:= YANG DILEPASKAN; Untuk memasuki status bagian yang kritis:= YANG DIINGINKAN; Multicast meminta kepada semua proses; proses permintaan yang ditunda T:= request's timestamp; Menunggu sampai (jumlah jawaban yang diterima= (N– 1)); status:= YANG DIPEGANG; Sesudah menerima suatu permintaan < Ti, Pi (22:7)> pada pj (i ?? j)

jika(tate = HELD) atau ((state = WANTED) dan ((T, pj) < (Ti, pi))
then queue request from pi without replying; else reply immediately to pi; end if To exit the critical section state := RELEASED; reply to any queued requests;

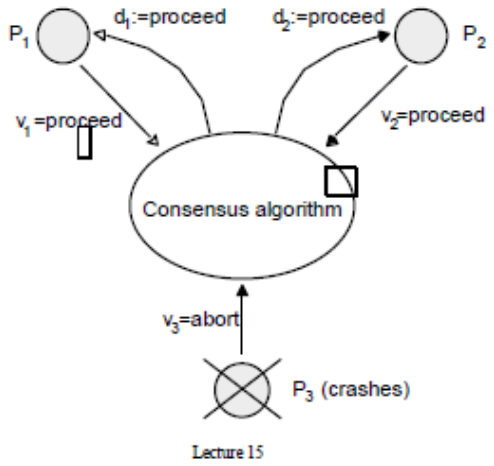
kemudian menunngu permintaan dari pi (22:7) tanpa menjawab; seketika jawaban itu akan muncul ke pi (22:7); berakhir; jika status bagian yang kritis keluar:= YANG DILEPASKAN; menjawab setiap permintaan queued;

Multicast pengeluaran timbal balik p1, p2 meminta akses [yang] secara bersamaan, p3 tidak dapat melakukan pengaksesan.

p2 tidak menjawab ke p1 karena mempunyai timestamp yang rendah.

F. Konsensus dan Permasalahan yang berhubungan

1.) Algoritma Konsensus



Gambar 6. Algoritma Konsensius [2]

Digunakan ketika persetujuan dibutuhkan pada tindakan :

1.) Pada proses transaksi
   Menjalankan atau mengakhiri transaksi
2. *) Mutual Exclusion*
   Prsoes mana yang boleh masuk C.S(*Critical Section*)
3.) Pada Sistem Kontrol

Meneruskan atau mengakhiri berbasis pada pembacaan sensor

Model dan Asumsi
Model :

1.) N proses
2.) Menyampaikan pesan
3.) Sinkron dan asinkron
4.) Komunikasi yang handal

Kegagalan

1.) Proses *crash*
2.) (Byzantine) gagal, berubah-ubah

Proses dapat menjadi berbahaya

Consensus : Ide Utama
Mula-mula

1.) Proses dimulai pada *state* yang tidak ditentukan
2.) Mengusulkan nilai inisal dari D

Kemudian

1.) Proses berkomunikasi, pertukara nilai
2.) Mencoba untuk menentukan
3.) Tidak bisa merubah keputusan nilai value pada *state* yang ditentukan

Kesulitan
Harus tetap mengambil keputusan ketika terjadi *crash.*

Syarat Konsensus

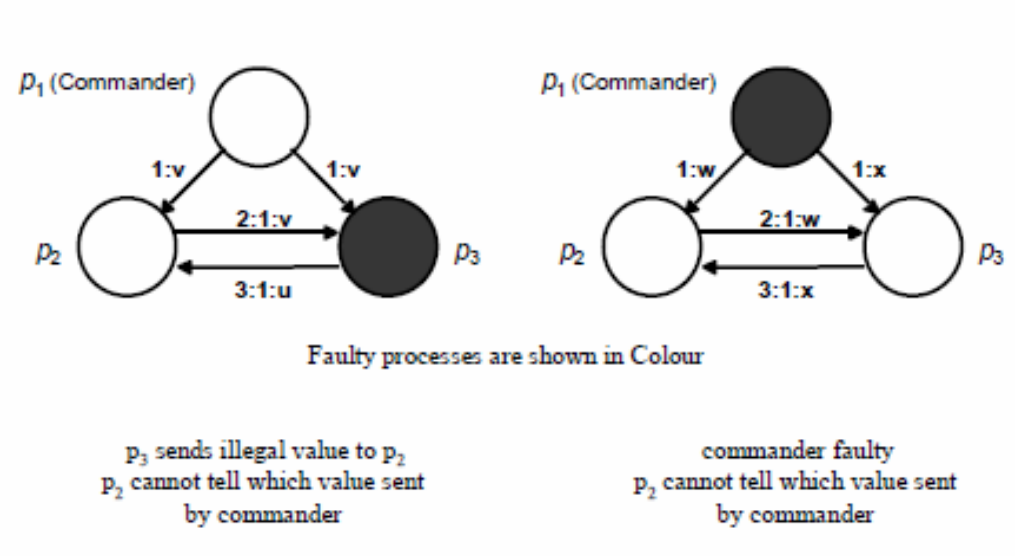
Termination, secara cepat mengatur semua proses secara benar pada keputusan nilai(*decision value*).
Agreement, satu atau beberapa proses harus sudah ditentukan pada keputusan nilai *(decision value*) yang sama

Konsensus pada sistem sinkron

1.) Menggunakan dasar *Multicast,* menjamin sampainya proses tanpa *crash.*
2.) Memuat/mengenal proses *crash.*

2.) Byzantine Generals

Proses penampilan dari kegagalan, bergantung kepada f dari N preoses kegagalan. Didalam sistem sinkron, dapat menggunakan *timeout* untuk men-detect ketiadaan dari sebuah pesan. Tidak dapat disimpulkan bahwa proses crash jika tidak ada jawaban. *Impossibility* dengan N < = 3f. Dalam sistem Asinkron, tidak dapat menggunakan *timeout* untuk mendeteksi ketiadaan dari sebuah pesan. *Impossibility* dengan kejadian sebuah kegagalan. Tidak ada Solusi untuk persetujuan/agreement untuk N=3 dan f=1. Dapat menggeneralisasi impossibility untuk N<=3f.



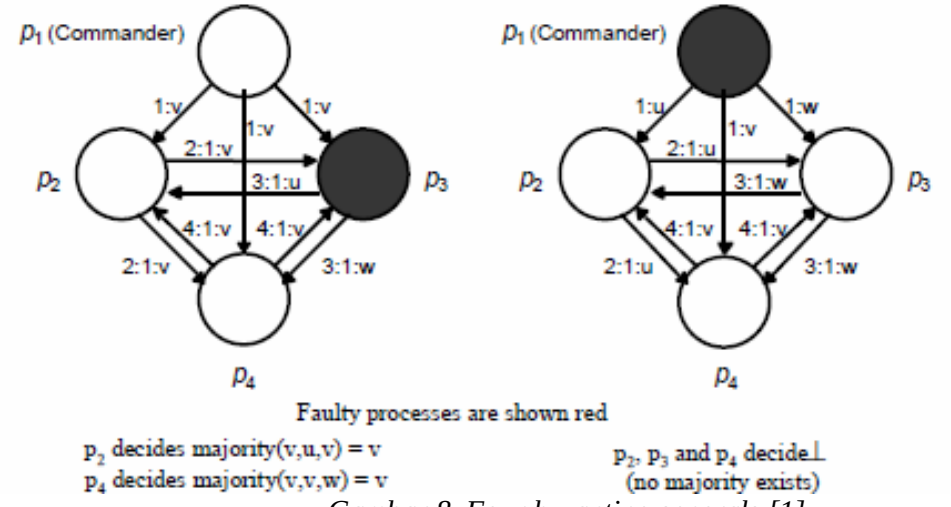
Gambar 7. Three Byzantine Generals [1]

Jadi, jika terdapat solusinya, P2 menentukan nilai yang dikirim oleh *commander* (v) ketika *commander* terhubun, dan juga ketika *commander* gagal(w), semenjak tidak dapat membedakan diantara kedua skenario. Dengan alasan yang sama untuk p3. Jadi karena p2 ditentukan di w, p3 di x jika tak ada kegagalan *commander*. Maka tidak ditemukan sebuah solusi.

Namun solusi ditemukan untuk 4 proses dengan satu kegagalan.

- Commander mengirim value ke tiap letnan
- Tiap letnan mengirim value yang diterima ke lawannya
- Jika commander gagal, maka letnan yang benar mencari semua value yang dikirim commander
- Jika letnan gagal, setiap letnan yang benar menerima 2 copy value dari commander.

Jadi letnan yang benar dapat menentukan mayoritas value yang diterima. Dapat men-generalisasi untuk N>=3f+1



Gambar 8. Four byzantine generals [1]

III. KESIMPULAN

1. Salah satu masalah dalam pembuatan algoritma yang bisa menyebabkan terjadinya *crash* adalah bila terjadi proses *crash* itu sendiri. *Failure Detector* [Chandra and Toueg 1996, Stelling et al. 1998] adalah suatu layanan yang mempertanyakan apakah proses yang terjadi gagal atau tidak.
2. *Failure detector* dapat membantu kita untuk mengetahui sebab-sebab terjadinya kegagalan pada sistem terdistribusi.
3. Jika sekumpulan proses bersama-sama menggunakan sumber, *mutual exclusion* dibutuhkan untuk mencegah gangguan dan menjaga konsistensi ketika sedang mengakses sumber.
4. Algoritma yang digunakan untuk memilih proses unik untuk memainkan fakta-fakta disebut *election alghoritm.*
5. ***Multicast*** atau ***multicasting*** adalah sebuah teknik di mana sebuah data dikirimkan melalui jaringan ke sekumpulan komputer yang tergabung ke dalam sebuah grup tertentu, yang disebut sebagai ***multicast group.***
6. Beberapa algoritma yang terkait pada Coordination & Agreement diantaranya adalah:
   - *Ring Based Alghoritm*
   - *Algoritma Koordinasi*
   - *Centralized Mutual Exclusion*
   - *Multicast Mutual Exclusion*
   - *Chang&Robert alghiritm*
   - *Richart – Agrawala Alghoritm*

7. Algoritma konsensus digunakan ketika persetujuan dibutuhkan pada tindakan :
   - Pada proses transaksi
     Menjalankan atau mengakhiri transaksi
   - *Mutual Exclusion*
     Prsoes mana yang boleh masuk C.S(*Critical Section*)
   - Pada Sistem Kontrol
     Meneruskan atau mengakhiri berbasis pada pembacaan sensor

8. Byzantine Generals merupakan proses penampilan dari kegagalan, bergantung kepada f dari N preoses kegagalan.

IV. REFERENSI

[1]http://www.cs.bham.ac.uk/Lecture14,15_H.pdf.
[2]Coulouris George, Jean Dollimore, Tim Kindberg. *Distributed System Concepts and Design.*:Addison Wesley.
[3] http://csli-publications.stanford.edu/LFG/9/lfg04petersonwrk.pdf.
[4]http://www.cs.bham.ac.uk/~blundeln/DS/handouts/DS_Lecture_12_Coordination_and_Agreement_2-2x3.pdf.

[5]http://www. users.jyu.fi/~annauvi/ties427/Coordination.ppt.
[6]http://www.cs.sfu.ca/~fedorova/Teaching/CMPT431/Spring2009/Lectures/Lecture9-CoordinationAgreement.ppt.
[7]http://www.cs.uiuc.edu/class/su06/cs425/lectures/lect-10.pdf.
[8]http://www.cs.bham.ac.uk/~blundeln/DS/handouts/DS_Lecture_11_Coordination_and_Agreement-2x3.pdf.
[9]www.nets.rwth-aachen.de/content/teaching/lectures/sub/vs/vsSS02/05_Cooperation.pdf.
[10]http://cna.te.ugm.ac.id.