IF3230 Sistem Paralel dan Terdistribusi
Tugas Besar 1 Consensus Paxos - Werewolf

A. Background Story

Setelah bekerja seharian penuh, para member dari μ's, 765 Production, dan 346 Production melalui chatroom mereka memutuskan ingin memainkan sebuah permainan yang disebut Werewolf. Namun, saat mereka berdiskusi untuk berkumpul untuk memainkan game ini, mereka terkendala oleh jadwal mereka yang padat dan tak menentu sehingga sangat sulit bagi mereka yang sibuk itu untuk bermain bersama. Oleh karena itu salah seorang diantara produser mengusulkan untuk membuat sebuah aplikasi yang dapat dimainkan secara masif dan bebas oleh para member group tersebut. Pada kesempatan ini, peserta kuliah IF3230 mendapatkan kehormatan untuk membangun aplikasi tersebut. Aplikasi yang akan dibangun ini membutuhkan keahlian dari para peserta kuliah IF3230 yang memahami sistem terdistribusi terutama protokol paxos karena pada permainan itu akan banyak voting dan member yang bermain bisa saja mendapat panggilan mendadak dan permainan harus tetap berlanjut.

B. Spesifikasi Permainan

Aplikasi yang akan dibuat adalah aplikasi simulasi permainan *werewolf* yang telah disimplifikasi menjadi hanya dua peran, dengan minimal jumlah pemain adalah 6, yaitu 2 *werewolf* dan 4 penduduk desa. Peserta IF3230 diminta untuk membuat aplikasi dengan arsitektur *client-server*. Pemain dapat saling mengontak secara langsung melalui daftar IP pemain yang disediakan oleh *server*.

Alur dari permainan ini adalah sebagai berikut.

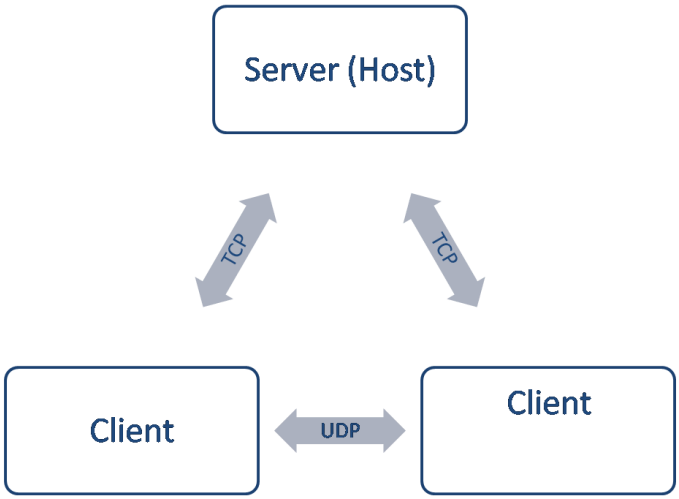
1. Malam Hari
    a. Permainan dimulai pada malam hari, yaitu situasi untuk para *werewolf* untuk saling berkenalan.
    b. Pemain non-*werewolf* tidak dapat melihat siapa *werewolf* selayaknya pada game *werewolf* seharusnya..
    c. Mulai dari malam kedua, setiap malamnya para *werewolf* akan mencapai kesepakatan target pembunuhan malam, dengan *werewolf* sendiri tidak termasuk sebagai target pembunuhan yang dapat dipilih. Pemilihan korban dilakukan berdasarkan hasil pemilihan mayoritas dari orang-orang dengan peran *werewolf*.
        i. *Proposer* untuk setiap tahap pada malam hari mengambil *proposer*/*leader* dari hari yang sama.
        ii. Bedanya dengan kasus siang hari, pemilihan korban dapat dilakukan berulang kali hingga tercapainya keputusan.
        iii. Teknis penyampaian proposal lainnya sama dengan pada siang hari.
    d. Malam hari akan selesai ketika *werewolf* telah mencapai kesepakatan korban.

2. Siang Hari
    a. Korban dinyatakan apabila ada.
    b. Dilakukan pemilihan *client* pengurus pemilihan suara (KPU) oleh para *client* yang masih aktif (bukan secara *gameplay*, namun untuk pemrosesan), yaitu dengan fase *leader election* dari algoritma pencapaian konsensus *Paxos*. Proses pemilihan KPU ini bertujuan untuk melakukan voting untuk membunuh pemain yang diduga sebagai *werewolf*.
    c. Setelah *leader* didapatkan, dilakukan pengumpulan suara dari *client-client* yang ada. *Leader* akan meneruskan suara terbanyak (memberikan proposal) kepada *acceptor*, selayaknya *proposer* pada konsep *Paxos*.
        i. Untuk kasus tidak ada mayoritas (cth: 2 pemain mendapat 2 vote), maka dapat dilakukan pemilihan satu kali lagi.
        ii. Apabila pada pemilihan kedua juga masih tidak ada mayoritas, maka tidak ada yang dibunuh pada hari itu.
    d. Untuk tugas ini, *leader* sendiri tidak boleh memegang peran *acceptor*.
    e. Hasil keputusan akan diberitahukan ke semua (sebagai *listener*) melalui server terpusat.

Ada pula spesifikasi lainnya, yaitu sebagai berikut

- alamat *server* yang dimasukkan secara dinamik saat aplikasi berjalan (tidak di-*hardcode*)
- ronde permainan dan peran masing-masing tetap ditampilkan pada layar masing-masing
- antar *client-server* satu kelompok dengan kelompok lainnya dapat berkomunikasi tanpa masalah dengan protokol yang ditetapkan pada subbab C.
- Permainan berakhir saat jumlah *werewolf* sama dengan jumlah penduduk, atau saat semua *werewolf* berhasil ditangkap.
- Pemain yang telah dibunuh tetap mendapat informasi terbaru keadaan permainan
- Peran pemain yang telah terbunuh ditampilkan pada seluruh klien.
- Untuk menguji berjalannya *Consensus Paxos*, terdapat modul kontrol yang berfungsi untuk mensimulasikan paket yang hilang saat pengiriman. Modul ini **wajib diimplementasikan** agar dapat mensimulasikan *unreliable connection* dan penanganan yang tepat. Modul ini hanya digunakan dalam di bagian *Paxos* pada seluruh fase yang ada pada *Paxos*.
  Contoh kode dapat dilihat pada link berikut:
  https://www.dropbox.com/s/y1xq5t55458im9d/sister.zip?dl=0
- Bahasa pemrograman yang dapat digunakan untuk membuat program ini dibebaskan kepada peserta
- Pada sisi client, User Interface dibebaskan kepada kreatifitas peserta, tetapi ditegaskan bahwa UI **tidak menjadi komponen penilaian utama** dari tugas ini sehingga utamakan fungsionalitas program dibandingkan tampilan dari client.

C. Protokol Komunikasi



Peserta praktikum diwajibkan untuk membuat dua program yaitu program server dan program client, adapun protokol komunikasi dari server ke client menggunakan protokol TCP, sedangkan protokol antar client menggunakan protokol UDP. Berikut ini adalah protokol komunikasi yang harus diterapkan oleh seluruh peserta praktikum, diharapkan seluruh peserta untuk mengikuti protokol yang telah dibuat sehingga nantinya program antar kelompok dapat saling berkomunikasi.

1. Join Game

Flow	**Client -> Server**

Request example	`{`
	`  "method": "join",`
	`  "username": "sister"`
	`}`

OK Response example	`{`
	`  "status": "ok",`
	`  "player_id": 3 // Dari 0 - N player!!`
	`}`

Fail Response example	`{`
	`  "status": "fail",`
	`  "description": "user exists"`
	`}`

	OR
	
	{
	"status": "fail",
	"description": "please wait, game is currently running"
	}

Error Response example	{
	"status": "error",
	"description": "wrong request"
	}

2. **Leave Game**

Flow	**Client -> Server**

Request example	{
	"method": "leave"
	}

OK Response example	{
	"status": "ok",
	}

Fail Response example	{
	"status": "fail",
	"description": <penjelasan>
	}

Error Response example	{
	"status": "error",
	"description": "wrong request"
	}

3. **Ready Up**

Flow	**Client->Server**

Request example	{
	"method": "ready"
	}

OK Response	{

example	"status": "ok", "description": "waiting for other player to start" }
Fail Response example	

Error Response example	{
	"status": "error",
	"description": "PUT DESCRIPTION HERE"
	}

4. **List Client**

Flow	**Client->Server**

Request example	{
	"method": "client_address"
	}

OK Response example	{ "status": "ok", "clients" : [["player_id" : 0, "is_alive" : 1, "address" : 192.168.1.1, "port" : 9999, "username" : "sister"], ["player_id" : 1, "is_alive" : 1, "address" : 192.168.1.2, "port" : 9999, "username" : "gaib"], ["player_id" : 2, "is_alive" : 1, "address" : 192.168.1.3, "port" : 9999, "username" : "irk"], ["player_id" : 3, "is_alive" : 0, "address" : 192.168.1.4, "port" : 9999, "username" : "basdat"],]

	"description": "list of clients retrieved" }

Fail Response example	{
	"status": "fail",
	"description": <penjelasan>
	}

Error Response example	{
	"status": "error",
	"description": <penjelasan>
	}

5. **Paxos Prepare Proposal**
   Yang bisa jadi proposer cuman pid dua terbesar (player ke n dan n-1)

Flow	**Client (Proposer) -> Client(Acceptor)**

Request example	{
	"method": "prepare_proposal",
	"proposal_id": (1, <player-id>)
	}

OK Response example	{
	"status": "ok",
	"description": "accepted",
	"previous_accepted":
	<previous_accepted_kpu_id>
	}

Fail Response example	{
	"status": "fail",
	"description": "rejected"
	}

Error Response example	{
	"status": "error",
	"description": <penjelasan>
	}

6. **Paxos Accept Proposal**
   Yang bisa jadi proposer cuman pid dua terbesar (player ke n dan n-1)

Flow	Client (Proposer) -> Client(Acceptor)

Request example	{
	“method”: “accept_proposal”,
	“proposal_id”: (1, <player-id>),
	“kpu_id”: <nilai sesuai protokol paxos>,
	}

OK Response example	{
	“status”: “ok”,
	“description”: “accepted”,
	}

Fail Response example	{
	“status”: “fail”,
	“description”: “rejected”
	}

Error Response example	{
	“status”: “error”,
	“description”: <penjelasan>
	}

7. **Client Accept Proposal**
   Acceptor adalah semua client

Flow	Client (Acceptor) -> Server(Learner)

Request example	{
	“method”: “prepare_proposal”,
	“kpu_id”: <player-id>,
	“Description”: “Kpu is selected”
	}

OK Response example	{
	“status”: “ok”,
	“description”: “”
	}

Fail Response example	{
	“status”: “fail”,
	“description”: <penjelasan>
	}

Error Response example	{ "status": "error", "description": <penjelasan>

8. **Kill Werewolf Vote**

Flow	**Client -> KPU**

Request example	{
	"method": "vote_werewolf",
	"player_id": 2
	}

OK Response example	{
	"status": "ok",
	"description": ""
	}

Fail Response example	{
	"status": "fail",
	"description": ""
	}

Error Response example	{
	"status": "error",
	"description": ""
	}

9. **Info Werewolf Killed**

Flow	**KPU->Server**

Request example	{
	"method": "vote_result_werewolf",
	"vote_status": 1,
	"player_killed": 4,
	"vote_result": [(0, 1), (1, 2), …]
	}
	
	{
	"method": "vote_result",
	"vote_status": -1,
	"vote_result": [(0, 1), (1, 2), …]

	} **// (0,1): player_id 0 divote sebanyak 1 pemain** **// (1,2): player_id 1 divote sebanyak 2 pemain**

OK Response example	{
	"status": "ok",
	"description": ""
	}

Fail Response example	{
	"status": "fail",
	"description": ""
	}

Error Response example	{
	"status": "error",
	"description": ""
	}

10. Kill Civilian Vote

Flow	**Client -> KPU**

Request example	{
	"method": "vote_civilian",
	"player_id": 2
	}

OK Response example	{
	"status": "ok",
	"description": ""
	}

Fail Response example	{
	"status": "fail",
	"description": ""
	}

Error Response example	{
	"status": "error",
	"description": ""
	}

11. Info Civilian Killed

Flow	**KPU->Server**

Request example	{
	"method": "vote_result_civilian",
	"vote_status": 1,
	"player_killed": 4,
	"vote_result": [(0, 1), (1, 2), …]
	}
	
	{
	"method": "vote_result",
	"vote_status": -1,
	"vote_result": [(0, 1), (1, 2), …]
	}

OK Response example	{
	"status": "ok",
	"description": ""
	}

Fail Response example	{
	"status": "fail",
	"description": ""
	}

Error Response example	{
	"status": "error",
	"description": ""
	}

12. Start Game

Flow	**Server -> Client**

Request example	{
	"method": "start",
	"time": "day",
	"role": "werewolf",
	"friend": ["ahmad", "dariel"],
	"description": "game is started",
	}
	// Role available: "werewolf" and "civilian"

	// friend is given if the role is werewolf, otherwise it is empty
OK Response example	{ "status": "ok" }
Fail Response example	{ "status": "fail", "description": "" }
Error Response example	{ "status": "error", "description": "" }

13. **Change Phase**

Flow	**Server -> Client**
Request example	{ "method": "change_phase", "time": "day", "days": 3, "description" : "PUT NARRATION HERE" }
OK Response example	{ "status": "ok", }
Fail Response example	{ "status": "fail", "description": "" }
Error Response example	{ "status": "error", "description": "" }

14. **Game Over**

Flow	**Server -> Client**

Request example	{
	"method": "game_over",
	"winner": "werewolf",
	"description": "PUT NARRATION HERE"
	}

OK Response example	{
	"status": "ok",
	}

Fail Response example	{
	"status": "fail",
	"description": ""
	}

Error Response example	{
	"status": "error",
	"description": ""
	}

D. Pengerjaan & Pengumpulan

Tugas dikerjakan secara berkelompok, dimana terdapat maksimal 3 orang untuk tiap kelompok (berada pada kelas yang sama). Deadline untuk pengumpulan Tugas Besar IF3230 adalah 29 April 2016, 23.59. Mekanisme pengumpulan serta jadwal untuk demo tugas besar akan diumumkan kemudian melalui grup milis. Adapun segala kecurangan akan mendapatkan konsekuensi. Bila ada pertanyaan yang kurang jelas mengenai spesifikasi soal, peserta dapat menggunakan milis sebagai sarana diskusi dan tanya jawab.

Selamat mengerjakan!

Link menuju referensi mengenai paxos:

1. https://www.quora.com/Distributed-Systems-What-is-a-simple-explanation-of-the-Paxos-algorithm/answer/Vineet-Gupta?srid=Cehd
2. http://the-paper-trail.org/blog/consensus-protocols-paxos/
3. https://raft.github.io/ (Have some similarities with paxos, good visualization to understand the protocol)