

Laporan Tugas Besar II – Sistem terdistribusi

Implementasi Gunbond Peer to Peer dengan Memanfaatkan Senarai Sirkular



May 4, 2013

Abraham Krisnanda S. – 13510033

Raymond Lukanta – 13510063

Edward Samuel P. – 13510065

Anasthasia Amelia - 13510093

# Konfigurasi Jaringan

## Socket pada Peer

Setiap peer memiliki 3 buah socket yang aktif, yaitu:

1. Socket yang berhubungan ke Tracker (trackerSocket)

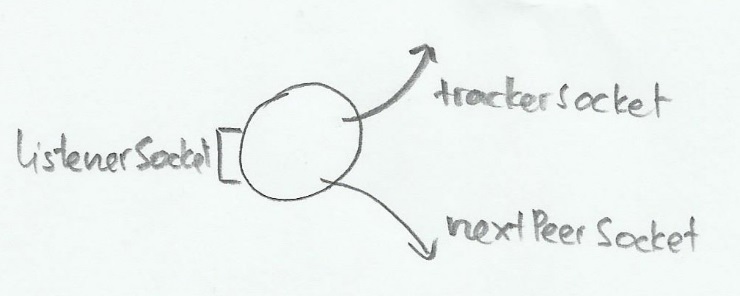
Socket ini selalu aktif untuk berhubungan ke Tracker. Baik untuk meminta bergabung dalam permainan, pembuatan room, bergabung ke room, keep alive, dll.

1. Socket yang berhubungan ke Next Peer (nextPeerSocket)

Socket ini yang berhubungan ke tetangga sebelahnya. Hanya terdapat sebuah peer tetangga yang aktif koneksinya. Socket ini bersifat sebagai “client” untuk berhubungan dengan peer tetangganya.

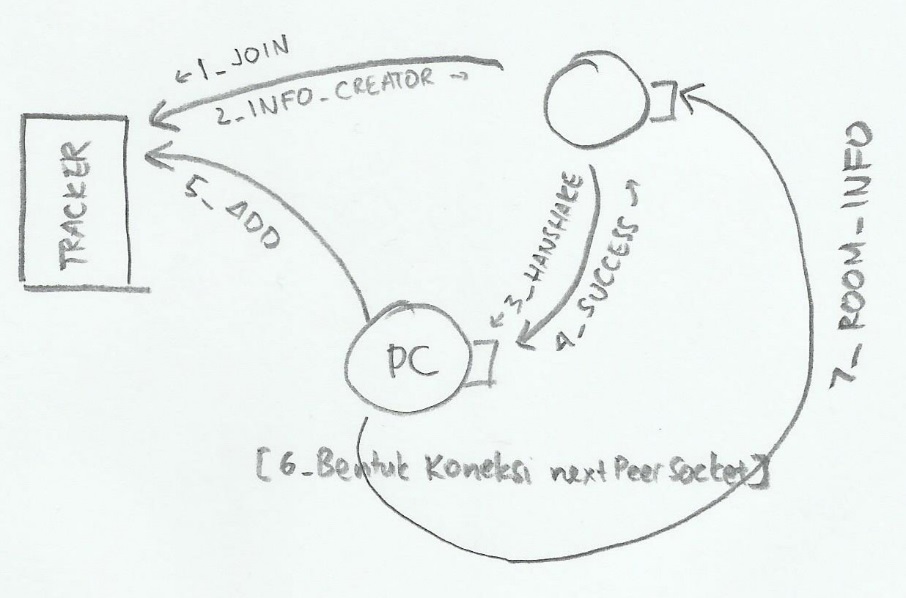
1. Socket yang mendengarkan request dari peer tetangga (listeningSocket)

Socket inilah yang mendengarkan permintaan dari peer tetangga yang terhubung dengan peer ini.



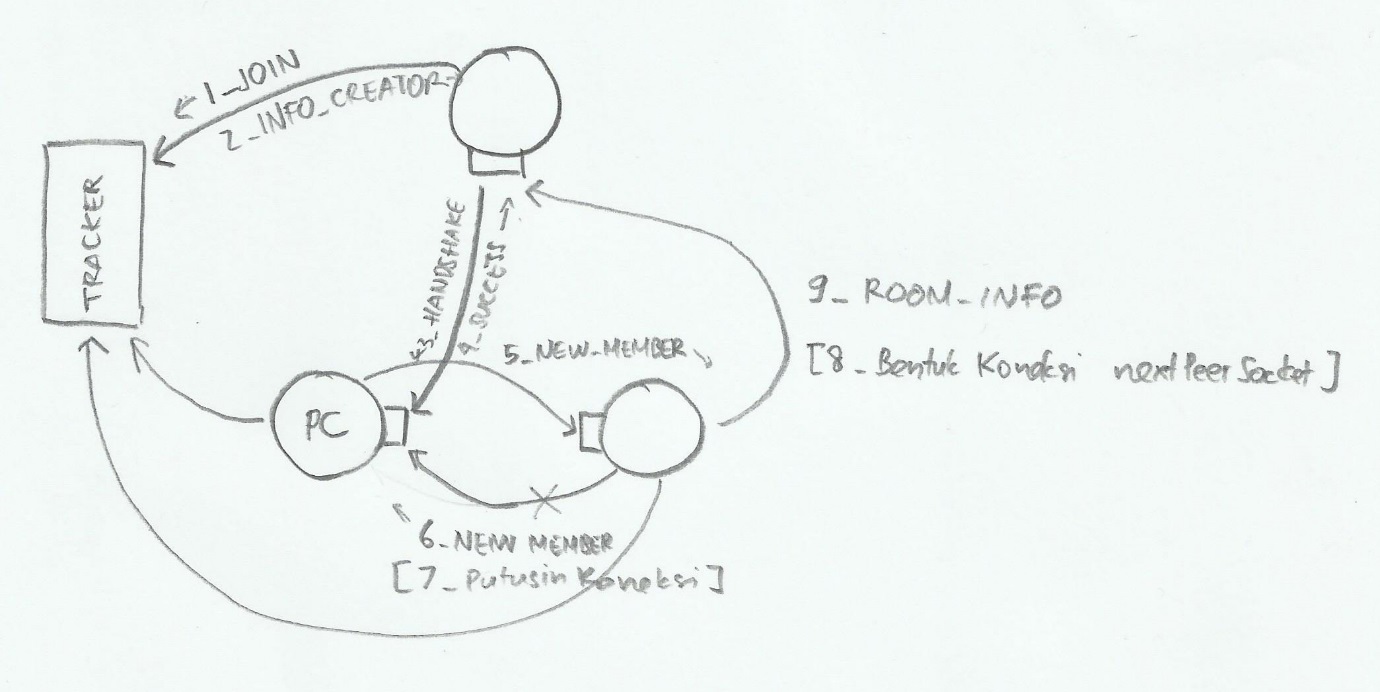
Berikut di bawah ini adalah penjelasan mekanisme jaringan yang dibentuk ketika baru terdapat tracker dan peerCreator:

1. Peer yang baru (newPeer) akan bergabung akan menghubungi tracker dengan mengirimkan pesan JOIN.
2. Ketika tracker menerima pesan JOIN, tracker akan membalas dengan pesan INFO\_CREATOR yang berisi informasi mengenai peerCreator.
3. Setelah newPeer menerima INFO\_CREATOR, newPeer akan mengirimkan pesan HANDSHAKE ke peerCreator.
4. PeerCreator lalu mengirimkan pesan SUCCESS ke newPeer.
5. PeerCreator juga mengirimkan pesan ADD ke tracker.
6. Selanjutnya, peerCreator akan menghubungkan nextPeerSocket-nya ke listenerSocket milik newPeer.
7. Terakhir, newPeer akan mengirimkan pesan ROOM\_INFO ke peerCreator.



Berikut di bawah ini adalah penjelasan mekanisme jaringan yang dibentuk ketika terdapat tracker, peerCreator, dan 1 peer:

1. Peer yang baru (newPeer) akan bergabung akan menghubungi tracker dengan mengirimkan pesan JOIN.
2. Ketika tracker menerima pesan JOIN, tracker akan membalas dengan pesan INFO\_CREATOR yang berisi informasi mengenai peerCreator.
3. Setelah newPeer menerima INFO\_CREATOR, newPeer akan mengirimkan pesan HANDSHAKE ke peerCreator.
4. PeerCreator lalu mengirimkan pesan SUCCESS ke newPeer.
5. PeerCreator lalu mengirimkan pesan NEW\_MEMBER ke peer.
6. Peer akan melanjutkan pesan NEW\_MEMBER yang dimilikinya ke peerCreator.
7. Koneksi trackerSocket dari peer ke peerCreator akan diputuskan.
8. Koneksi nextPeerSocket dari peer akan disambungkan ke listenerSocket milik newPeerSocket.
9. Peer mengirimkan pesan ROOM\_INFO ke newPeer.



## Model Jaringan Peer-to-Peer

Model jaringangan per to peer yang digunakan adalah sirkular list dalam setiap room. Antar room satu dengan yang lain, tidak terdapat hubungan. Struktur sirkular list dipakai karena jumlah peer dalam satu room tidak terlalu banyak. Sirkular list juga hemat dari sisi soket yang digunakan. Karena jumlah soket yang digunakan sedikit, maka konfigurasi jaringan menjadi tidak terlalu rumit.

# Spesifikasi Paket Message dalam Permainan

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Ukuran** | 11 | 8 | 1 | 4 | 4 | 4 | 4 | 4 | 1 | 4 |
| **Indeks** | 0-10 | 11-18 | 19 | 20-23 | 24-27 | 28-31 | 32-35 | 36-39 | 40 | 41-44 |
| **Message** | GUNBONDGAME | 00000000 | MessageType | Position (X) | Position (Y) | Angle | Power | Damage | isRocketFlying | PeerID |

1. **Position (float)**

Merupakan pesan yang merepresentasikan posisi pemain dalam game. Message position terdiri dari 2 komponen, yaitu X dan Y.

1. **Angle (float)**

Merupakan pesan yang merepresentasikan kemiringan cannon milik pemain yang akan menembak dalam game. Pemain dapat memutar cannon-nya dalam jangkauan 0o-180o.

1. **Power (float)**

Merupakan kekuatan pemain dalam menembakkan roket. Kekuatan minimum yang dapat diatur pemain adalah 0. Sedangkan kekuatan maksimum yang dapat diatur adalah 500.

1. **Damage (float)**

Merupakan tingkat kerusakan yang diterima lawan dalam game. Damage yang diterima bernilai sepertiga dari nilai Power.

1. **isRocketFlying (bool)**

merupakan pesan yang memberi tahu bahwa roket sedang diluncurkan.

1. **peerID (int)**

merupakan peerID.

Setiap message akan diubah menjadi paket-paket data dalam bentuk byte per byte. Untuk menampung seluruh message di atas, diperlukan 45 byte data. Message tersebut akan dipetakan sebagai berikut:

|  |
| --- |
| public static Message CreateMessageGame(float x, float y, float angle, float power, float damage, bool isRocketFlying, int peerId)  {  byte[] data = new byte[45];  FillHeader(data);  //data[0..18] = 0 for FillHeader(data)  //data[19] for Message Type  data[19] = 123;  // data[20..23] for X position  byte[] temp0 = BitConverter.GetBytes(x);  data[20] = temp0[0];  data[21] = temp0[1];  data[22] = temp0[2];  data[23] = temp0[3];  // data[24..27] for Y position  byte[] temp = BitConverter.GetBytes(y);  data[24] = temp[0];  data[25] = temp[1];  data[26] = temp[2];  data[27] = temp[3];  //data[28..31] for angle  byte[] temp3 = BitConverter.GetBytes(angle);  data[28] = temp3[0];  data[29] = temp3[1];  data[30] = temp3[2];  data[31] = temp3[3];  //data[32..35] for power  byte[] temp4 = BitConverter.GetBytes(power);  data[32] = temp4[0];  data[33] = temp4[1];  data[34] = temp4[2];  data[35] = temp4[3];  //data[36..39] for damage  byte[] temp5 = BitConverter.GetBytes(damage);  data[36] = temp5[0];  data[37] = temp5[1];  data[38] = temp5[2];  data[39] = temp5[3];  //data[40] for isRocketFlying  data[40] = Convert.ToByte(isRocketFlying);  // data[41..44] for peerId  byte[] id = ConvertIntToByte(peerId);  data[41] = id[0];  data[42] = id[1];  data[43] = id[2];  data[44] = id[3];  Message m = new Message();  m.data = data;  return m;  } |