Internet of Things

Smart Refrigerator

Yoan Pratama - 11110110122

Ritos Penyawang - 11110110132

Andreas Budiman - 11110110133

Riyadi Sembel - 10110210004

FAKULTAS TEKNOLOGI INFORMASI DAN INFORMATIKA

UNIVERSITAS MULTIMEDIA NUSANTARA

TANGERANG

2014

PENDAHULUAN

1. Latar Belakang

Internet of Things adalah sebuah interkoneksi antar embedded devices melalui jaringan internet. Pada dasarnya, IoT diharapkan untuk menyediakan koneksi devices, system, dan layanan yang melebihi machine-to-machine communications dan mencakup berbagai protocol, domain, dan aplikasi.

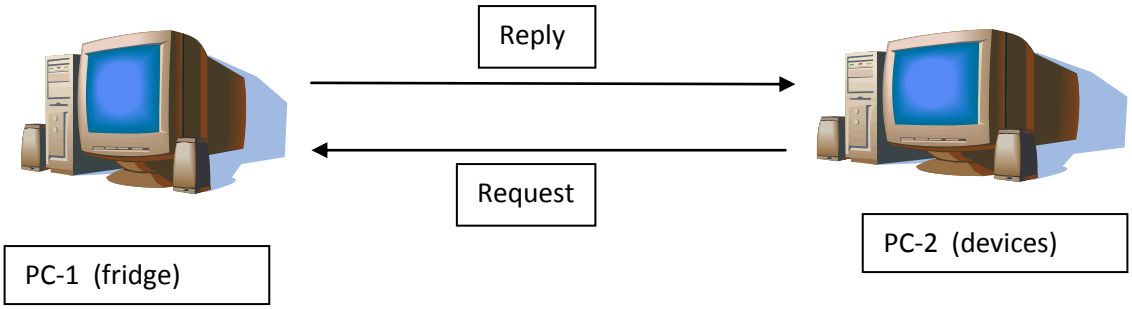
Embedded devices di sini bisa berupa bermacam-macam device seperti *heart monitoring implants*, *biochip transponder* yang ditanamkan pada hewan ternak, kendaraan yang memiliki sensor, dan masih banyak lagi. Pada saat ini, aplikasi yang telah diterapkan adalah *smart thermostat system* (pengatur suhu ruangan) dan mesin pencuci baju yang menggunakan koneksi wifi untuk memonitor secara remote.

Integrasi dengan jaringan internet membutuhkan setiap devices untuk memiliki IP Address sebagai *unique identifier*. Namun, karena keterbatasan ketersediaan IPv4 (yang hanya memperbolehkan 4.3 juta alamat unik), benda dalam IoT harus menggunakan IPv6 untuk mengakomodasi kebutuhan alamat unik yang banyak. Objek atau benda dalam IoT tidak harus yang memiliki sensor saja, melainkan objek seperti lampu atau kunci juga bisa diimplementasikan.

Dikarenakan objek-objek IoT akan membutuhkan koneksi internet, maka dibutuhkan sebuah *low-cost platform*. Bahkan untuk meminimalisasi dampak objek IoT terhadap lingkungan dan penggunaan energy, *low-power radios* akan digunakan untuk koneksi ke internet seperti *low-power radios* yang tidak membutuhkan wifi atau jaringan selular. Perusahaan Freeware Technologies telah merancang dan membuat alat komunikasi nirkabel hemat daya selama 20 tahun ini untuk mewujudkan implementasi machine-to-machine.

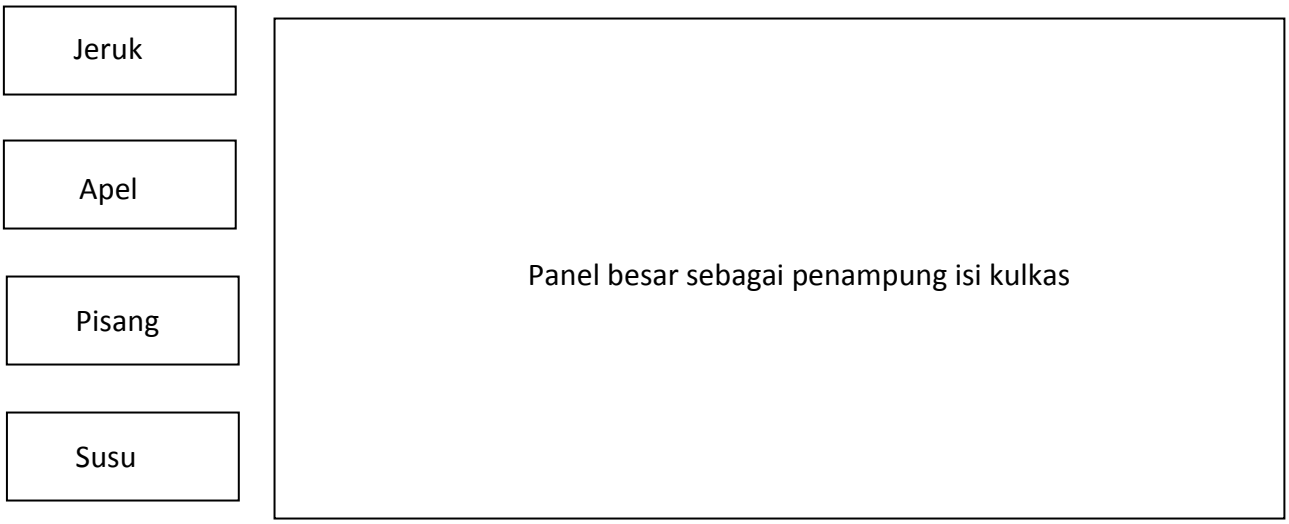
2. Desain Project

Dalam project ini, kami akan mencoba membuat program untuk menstimulasikan sebuah smart refrigerator, yaitu sebuah kulkas yang dapat diakses menggunakan device lain melalui sebuah jaringan. Nantinya, kita dapat mengetahui apa saja yang tersimpan didalam kulkas dan mengatur suhu kulkas melalui devices lain. Untuk itu akan dibutuhkan 2 buah PC untuk menstimulasikan project ini. PC-1 bekerja sebagai kulkas yang diberi jaringan, sedangkan PC-2 bekerja sebagai device yang dapat mengakses kulkas tersebut. Pada saat kita mau mengetahui apa yang ada didalam kulkas, kita dapat mengirimkan request kepada PC-1 yang selanjutnya request ini akan di balas oleh PC-1 dengan mengirimkan apa saja yang ada didalam kulkas. Begitu pun saat kita akan mengatur suhu kulkas, PC-2 akan mengirim pesan berupa suhu kulkas yang diharapkan user, sedangkan PC-1 akan mengatur suhu sesuai pesan yang diterima dari PC-2.



Gbr.2.1 Alur kerja project

Untuk programnya nanti PC-1 akan diberikan panel besar yang akan menampung item di dalamnya, panel ini bertindak penampung isi kulkas. Sedangkan untuk itemnya sendiri akan dibuat bentuk kotak dengan label nama yang melambangkan item tersebut. Seperti jeruk, apel, pisang, dsb.



Gbr 2.2 Prototype tampilan PC-1

Untuk memasukkan item kedalam kulkas kita cukup men-drag and drop kotak disebelah kiri kedalam panel besar disebelah kanan, yang nantinya item yang ter-drop didalam panel akan disimpan id-nya, sehingga nantinya kita dapat mengetahui jenis dan jumlah dari setiap item yang ada di dalam kulkas.

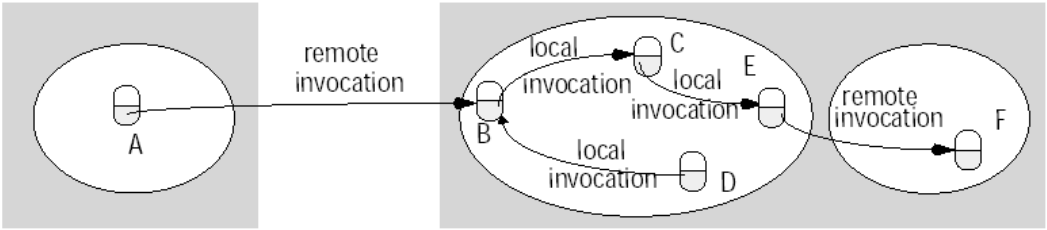
3. Remote Object Invocation (RMI)

Remote Object Invocation atau yang biasanya dikenal juga sebagai RMI, merupakan sebuah API yang ditulis dalam bahasa Java yang berorientasi objek. API ini memungkinkan sebuah Java Virtual Machine (JVM) untuk memanggil sebuah method lain yang berada dalam JVM yang berbeda.

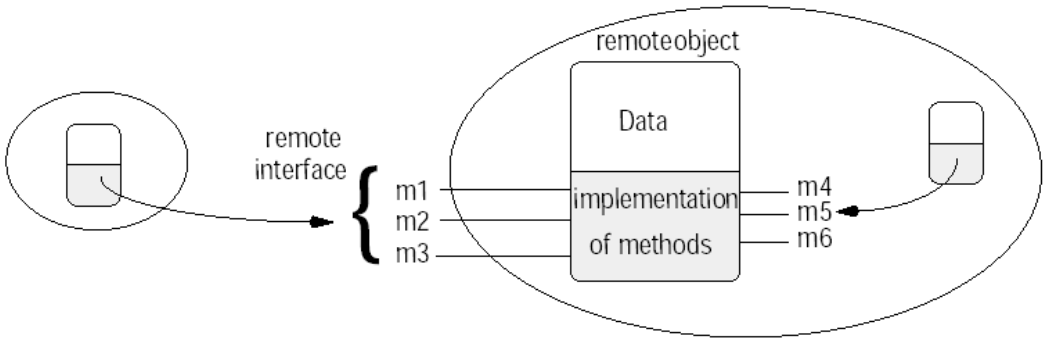
Pada awalnya, implementasi ini bergantung pada mekanisme representasi JVM classnya oleh karena itu hanya memungkinkan untuk melakukan pemanggilan method dari satu JVM ke JVM yang lain. Protokol ini dikenal sebagai Java Remote Method Protocol (JRMP). Untuk mendukung penggunaan API ini di platform non-Java, maka bisa digunakan CORBA.

Programmer pembuat API ini pada awalnya menggunakan kode umum untuk mendukung implementasi yang berbeda seperti HTTP transport. Oleh karena itu, kemampuan untuk mengirimkan value menggunakan "by value" ditambahkan ke CORBA.

Remote and local method invocations



A remote object and its remote interface



Terdapat dua konsep fundamental dalam model objek terdistribusi:

- *Remote Object Reference:* sebuah objek dapat memanggil sebuah method di objek yang lain hanya jika memiliki akses ke *remote object reference.*
- *Remote Interface:* setiap *remote object* memiliki sebuah *remote interface* yang menentukan method mana yang bisa dipanggil secara *remote*.

Terdapat 2 cara untuk mendapatkan sebuah *remote reference*:

- Melalui *parameter passing*, bisa sebagai input atau *return value* (seperti dalam java)

- Menggunakan sebuah *simple lookup service* secara explicit yang disebut *rmiregistry*

Dalam kedua kasus diatas, *remote reference* adalah sebuah object yang memiliki *client stub (proxy)*. Oleh karena itu kedua cara tersebut memiliki karakter yang mirip yaitu, mengimplementasi interface yang sama dengan *remote object*, merupakan sebuah instance dari *RemoteStub* dan proxy dapat di kirim.

Lookup service, rmiregistry, menyediakan binding servis yang dinamis. Hal ini memperbolehkan *rmiregistry* untuk mengatur asosiasi antara *symbolic name* dengan objek yang terdapat dalam server. Selain itu service ini juga tidak didistribusi, diimplementasikan melalui proses yang berbeda dan demi keamanan, maka service ini tidak memungkinkan untuk mengikat sebuah object yang dijalankan di host yang berbeda ke dalam registry.

Kelebihan

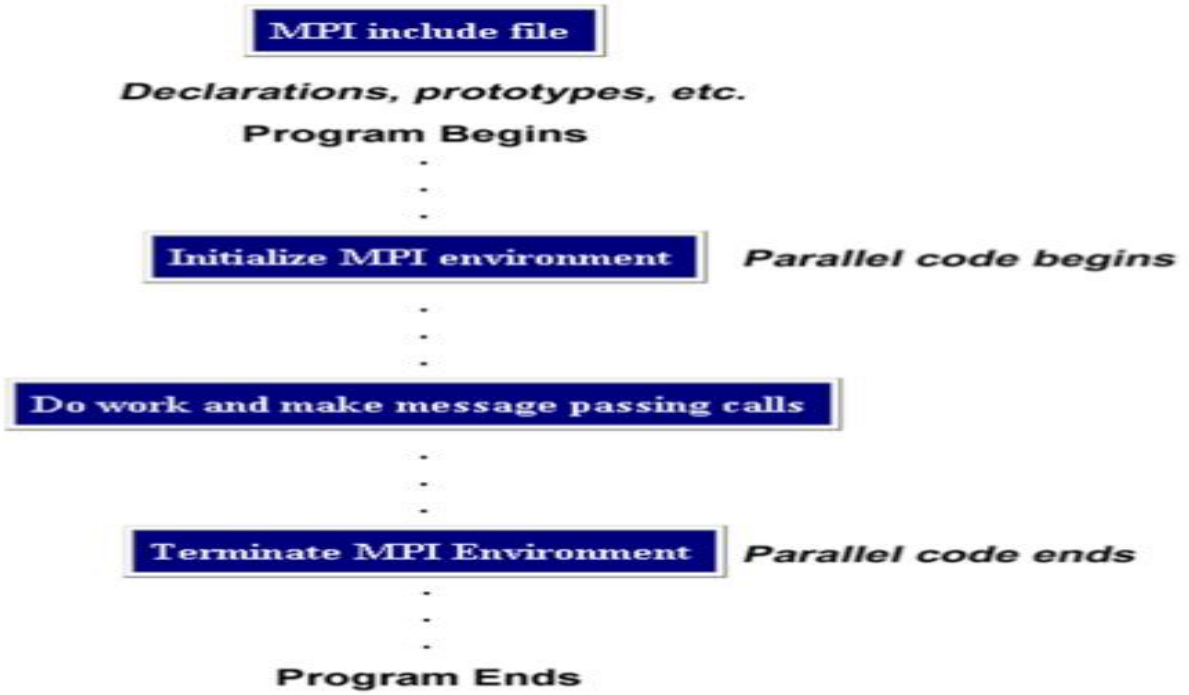
- Mudah untuk diimplementasikan
- Memanfaatkan model security java
- Dapat membuat interface yang dinamis
- Bekerja pada java yang artinya dapat multiplatform

Kekurangan

- Hanya dapat diimplementasi menggunakan bahasa pemrograman Java
- Kinerja lambat dari beberapa jenis middleware lainnya

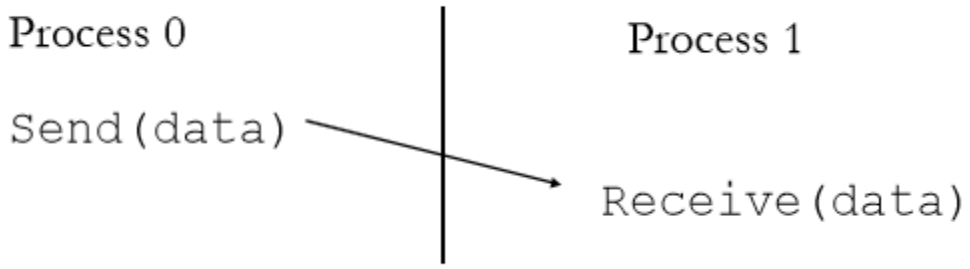
4. Message Passing Interface

Message Passing Interface adalah sebuah middleware yang digunakan untuk skenario komputasi performa tinggi. Tujuan dari dibuatnya MPI adalah untuk membuat suatu standard API untuk komputasi terdistribusi performa tinggi dengan memenuhi kriteria *easy to use, portable, efficient, flexible.*



MPI menggunakan sebuah objek yang dinamakan *communicators* dan *groups* untuk mengatur koleksi proses. Dalam sebuah *communicators,* setiap proses memiliki peringkatnya tersendiri:

- Sebuah Integer unik yang diberikan oleh system
- Peringkat dimulai dari 0 dan akan terus bertambah
- Digunakan oleh programmer untuk menentukan sumber dan tujuan message
- Sering digunakan juga untuk mengatur eksekusi program (if rank = 0 do this / if rank = 1 do that)



MPI menentukan:

- Bagaimana data disediakan
- Bagaimana pesan diidentifikasi oleh penerima
- Bagaimana implementasi komunikasinya

Kebanyakan MPI P2P dapat digunakan dengan *blocking* atau *non-blocking mode*

a) Blocking:
   - Hanya akan mengembalikan nilai setelah nilai tersebut aman untuk dimodifikasi oleh buffer aplikasi untuk digunakan kembali. Aman disini bukan berarti data yang dikirim telah diterima, namun bisa saja data tersebut terdapat dalam *system buffer* di host penerima.
   - Pengiriman dalam *blocking mode* dapat dilakukan secara *synchronous* (dengan konfirmasi dari penerima) atau juga bisa dilakukan secara *asynchronous* jika sebuah *system buffer* digunakan untuk menampung data yang akan dikirim secara berkala.
   - Penerimaan hanya mengembalikan nilai setelah data telah diterima dan siap untuk digunakan oleh program yang lain.

b) Non-blocking:
   - Ketika proses pengiriman dan penerimaan, *non-blocking* tidak harus menunggu sebuah komunikasi
   - Tidak boleh melakukan modifikasi terhadap buffer aplikasi sampai *non-blocking* terjadi. Hal ini bisa dilakukan dengan rutinitas “wait”
   - Komunikasi dalam *non-blocking* digunakan untuk mendahulukan komputasi dalam komunikasi dan memanfaatkan kemungkinan penambahan performa.

Kelebihan

- Memiliki tipe data sendiri (dapat di *define* dengan fungsi MPI)
- Kinerja jauh lebih cepat dari RMI
- Mudah digunakan, portable, efisien, dan fleksibel

Kekurangan

- Tidak multiplatform sehingga membutuhkan runtime environment khusus (untuk komunikasi antar komputer)
- Hanya dapat diimplementasikan pada C, C++, Fortran

REFERENSI

[1] Internet of Things http://en.wikipedia.org/wiki/Internet_of_Things

[2] Remote Method Invocation
http://en.wikibooks.org/wiki/Java_Programming/Remote_Method_Invocation

[3] Java Remote Method Invocation
http://en.wikipedia.org/wiki/Java_remote_method_invocation

[4] Remote Object References
http://www.csc.villanova.edu/~schragge/CSC8530/chapter5.htm

[5] Socket, RPC, and RMI http://corsi.dei.polimi.it/distsys/pub/13-socket_rpc_rmi.pdf

[6] Advance Message Passing with MPI http://corsi.dei.polimi.it/distsys/pub/19-mpi.pdf