KOMPARASI SINKRONISASI FILE ANTARA METODE *MOBILE AGENT* DAN METODE *PEER-TO-PEER*[1]

FILE SYNCHRONIZATION METHOD COMPARISON BETWEEN MOBILE AGENT AND PEER-TO-PEER

Faiq Wildana

Puslitbang Aplikasi Informatika dan Informasi dan Komunikasi Publik, Badan Litbang SDM,
Kementerian Komunikasi dan Informatika, Jl. Medan Merdeka Barat No. 9 Jakarta Pusat 10110, Indonesia,
Telp./ Faks: 021- 3800418
E-mail: faiq001@kominfo.go.id

Naskah diterima tanggal 11 September 2015, direvisi tanggal 18 Oktober 2015, disetujui tanggal 12 November 2015

Abstract

Good server has high availability, rapid response in addressing a number of the user according to his ability. Servers often run into problems so that the data might be inconsistent, and even data loss happen. The use of more than one server can be an alternative, either virtually or physically for the distribution of resources and reserves data. File synchronization is done to overcome it. Comparative research with quantitative approach aims to compare between mobile agent in the method of synchronizing files and to peer-to-peer application in performing file transfers. The peer-to-peer act as media to synchronize file. The literature study shows that this method of mobile agent as well as peer-to-peer have the same advantages. The results showed that the theoretical (literature) file synchronization methods can be implemented using a combination of mobile agent and peer-to-peer. In which the mobile agent is used as files change information carrier and used as a automation synchronization. Then the file transfer function is handled by the transfer files method adapt to Bit Torrent way. So that the synchronization speed can be improved. But there is a significant difference in the speed of the data comparison of both methods. In theory supposed to transfer files using peer-to-peer faster. From the results of this research can be done further engineering synchronize files using a combination of mobile agent and peer-to-peer, so that the results of the research can be proved definitely more precise.

***Keywords**: Mobile Agent, Peer-to-peer, File Synchronization*

Abstrak

Server yang baik memiliki ketersediaan yang tinggi *high* (*availability*), respon cepat dalam menangani sejumlah pengguna sesuai dengan kemampuannya. Server sering mengalami masalah sehingga data dalam server tidak konsisten, bahkan bisa kehilangan data. Penggunaan server lebih dari satu bisa menjadi alternatif, baik secara virtual maupun fisik untuk pembagian sumber daya dan cadangan data. Sinkronisasi file dilakukan untuk mengatasi hal tersebut. Penelitian dengan pendekatan komparatif kuantitatif ini bertujuan mengkomparasi antara metode *mobile agent* dalam melakukan sinkronisasi file dibandingkan dengan metode *peer-to-peer* dalam melakukan transfer file. Metode *peer-to-peer* dianalogikan sebagai media untuk melakukan sinkronisasi file. Studi literatur menunjukkan bahwa metode mobile agent maupun *peer-to-peer* sama-sama memiliki memiliki keunggulan. Hasil penelitian menunjukkan bahwa secara teori (studi literatur) metode sinkronisasi file dapat diterapkan menggunakan kombinasi antara *mobile agent* dan *peer-to-peer*. Di mana *mobile agent* digunakan sebagai pembawa informasi perubahan file serta digunakan sebagai otomatisasi sinkronisasi. Kemudian pengiriman file ditangani oleh metode pengiriman file mengadaptasi cara yang digunakan Bit Torrent. Sehingga kecepatan sinkronisasi dapat ditingkatkan. Namun terjadi perbedaan kecepatan signifikan dari komparasi data kedua metode. Seharusnya secara teori transfer file menggunakan *peer-to-peer* lebih cepat. Dari hasil penelitian ini diharapkan dapat dilakukan perekayasaan lanjut sinkronisasi file menggunakan gabungan dari mobile agent dan *peer-to-peer,* sehingga hasil pasti dari penelitian dapat dibuktikan lebih tepat.

Kata kunci : *Mobile Agent*, *Peer-to-peer*, Sinkronisasi *file*

[1]Naskah ini merupakan salah satu dari naskah yang telah dimuat dalam Prosiding Temu Ilmiah Nasional Peneliti Badan Litbang SDM, Kementerian Komunikasi dan Informatika, 2015

PENDAHULUAN

Server merupakan basis utama tersedianya layanan di internet. Hal ini karena jika kita mengakses sesuatu konten di internet, baik berupa situs web, aplikasi, maupun bentuk layanan lain bisa dipastikan kita terhubung dengan server. Kehandalan server diperlukan agar user dapat mengaksesnya kapanpun dengan tanpa masalah. Server yang handal adalah yang memiliki ketersediaan tinggi (*high availability*), respon cepat dalam menangani sejumlah user sesuai dengan kemampuannya. *High availability* menurut Tom Seldon (2001) adalah kondisi dalam periode waktu tertentu dimana selalu tersedia 24/7 (24 jam per hari,7 hari seminggu). Server sering mengalami masalah sehingga data dalam server tidak konsisten, bahkan bisa kehilangan data. Penggunakan server lebih dari satu bisa menjadi alternatif, baik secara virtual maupun fisik untuk pembagian sumber daya dan cadangan data.

Pembagian sumber daya biasa digunakan untuk server dengan jumlah pengguna yang sangat banyak. Beban user dibagi pada beberapa server identik dengan tujuan agar respon yang didapat *user* bisa lebih cepat. Namun dengan banyaknya *user* dan server yang ada, akan rawan terjadinya inkonsistensi data. Selain hal tersebut masalah lain yang mungkin timbul adalah apabila server *down* maka harus ada salinan atau *backup data*, agar ketersediaan data server tetap terjaga. Untuk mewujudkan dua hal tersebut diperlukan sinkronisasi *file* agar data tetap konsisten dan salinan untuk cadangan data.

Ada banyak metode sinkronisasi yang biasa digunakan pada server. Beberapa penelitian terkait antara lain: Rsync dengan penambahan protokol zeromQ yang dilakukan oleh Heru Setyo Nugroho dan Wahyu Suadi (2010). Metode lain yaitu menggunakan FTP server ditambah dengan SMS notifikasi yang dilakukan oleh I Made Darma Susila dan AAG Agung Putra Ratu Asmara (2014) Selain itu, ada juga penelitian yang membandingkan kinerja antara SAN dan NAS terkait sinkronisasi data yang dilakukan oleh Lekso Budi Handoko dan Chaerul Umam (2015). Gede Wahyudi dan Trisna Hanggara (2013) juga melakukan penelitian perbandingan antara NFS dan PDC.

Pada paper ini penulis mencoba mengomparasikan metode sinkronisasi yang telah penulis lakukan sebelumnya, yaitu sinkronisasi menggunakan metode mobile *agent* menggunakan *aglets* (2013) dibandingkan dengan metode *peer-to-peer* dalam melakukan transfer *file* yang dilakukan oleh Ary Mazharuddin Shiddiqi, dkk (2010). Metode *peer-to-peer* dianalogikan sebagai media untuk melakukan sinkronisasi *file*. Data tersebut dianggap masih relevan karena menyajikan data yang lengkap. Dari kedua metode tersebut penulis melakukan perbandingan secara teori melalui studi literatur dan perbandingan data sehingga diharapkan dapat menghasilkan kesimpulan dan dapat memberikan rekomendasi dari kesimpulan yang didapatkan.

Pada penelitian yang penulis lakukan sebelumnya menjelaskan bahwa *Aglets* merupakan teknologi *mobile agent* yang memiliki kemampuan berpindah dari satu *host* ke *host* lain. Ketika berpindah, *state* program ini disimpan dan dibawa ke *host* selanjutnya dan berjalan sebagai proses yang normal dengan kemampuan tersebut *agent* dapat digunakan untuk melakukan pekerjaan secara otomatis yaitu untuk sinkronisasi *file* pada server.

*Aglet*s menggunakan *platform* Java yang dikembangkan Danny B. Lange dan Mitsuru Oshima (1998) dari IBM Tokyo Research Laboratory. *Mobile agent* merupakan agen yang aktif dan dapat bergerak menuju *host* lain, atau mejelajahi jaringan untuk menjalankan tugasnya. *Mobile Agent* sering digunakan untuk mengumpulkan data, informasi atau suatu perubahan.

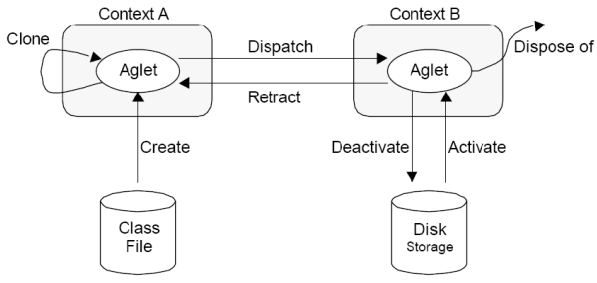
Keuntungan menggunakan *mobile agent.* Menurut Lange dan Oshima (1998) antara lain:

1. *Mobile Agent* dapat mengurangi beban jaringan. Sistem terdistribusi sering mengandalkan protokol komunikasi yang melibatkan beberapa koneksi untuk menyelesaikan tugas yang diberikan. Sehingga trafik jaringan yang terjadi membesar.

2. *Mobile Agent* memungkinkan pengguna untuk melakukan pengiriman sebuah set perintah dan mengirimkannya ke sebuah *host* tujuan di mana pemrosesan dilakukan secara lokal, sehingga dapat mengurangi trafik jaringan.
3. *Mobile Agent* mengatasi kelambatan jaringan. Sistem yang membutuhkan komunikasi secara *real-time*, memerlukan respon saat itu juga untuk melakukan perintah di mana dia berada. Pengaturan sistem tersebut melalui sebuah jaringan yang panjang sehingga berimbas pada respon yang lambat. *Mobile Agent* memberikan solusi atas hal tersebut. Agent dapat dikirimkan dari sebuah kontroler pusat untuk melakukan pemrosesan secara lokal dan mengeksekusi perintah dari kontroler secara langsung.
4. *Mobile Agent* mengenkapsulasi protokol. Ketika data dipertukarkan dalam sebuah sistem terdistribusi, tiap *host* memiliki sebuah protokol. Namun, apabila protokol yang ada berbeda dan maka perlu dilakukan *upgrade*. Tidak demikian dengan *Mobile Agent*. *Mobile agent* dapat berpindah ke *host* melalui protokolnya sendiri.
5. *Mobile Agent* dieksekusi secara asinkron dan mandiri.

   Perangkat bergerak *(mobile device*) sering mengandalkan koneksi jaringan secara terus-menerus. Selain tidak layak secara ekonomis maupun teknis juga rawan terputusnya koneksi. Untuk memecahkan masalah ini, tugas-tugas dapat ditanamkan pada *Mobile Agent*, yang dapat dikirimkan ke jaringan. Setelah pengiriman dilakukan, agen itu menjadi bebas dari proses di mana pembuatan *Mobile Agent* itu berada, dapat beroperasi secara asinkron dan mandiri. Perangkat bergerak dapat melakukan koneksi ulang apabila dibutuhkan untuk menarik agen-agen tersebut.
6. *Mobile Agent* beradaptasi secara dinamis. *Mobile Agent* dapat merasakan lingkungan eksekusinya dan bereaksi untuk berubah secara mandiri. *Mobile Agent* memiliki kemampuan yang unik dalam menjelajah *host-host* dalam jaringan dan dan memecahkan suatu masalah tertentu.
7. Sifat heterogen dari *Mobile Agent*. Sistem terdistribusi pada dasarnya adalah heterogen, baik dalam perspektif *perangkat keras* maupun *software*. Karena *Mobile Agent* secara umum tidak tergantung pada perangkat maupun software, hanya tergantung pada lingkungan di mana dia melakukan eksekusi. *Mobile Agent* dapat diintegrasikan pada sistem yang heterogen.
8. *Mobile Agent* dapat bertahan dan memiliki toleransi terhadap kegagalan. Kemampuan *Mobile Agent* yang dapat bereaksi secara dinamis pada situasi-situasi dan kejadian-kejadian yang beragam membuatnya toleran terhadap kegagalan dalam lingkungan terdistribusi. Jika sebuah *host* hendak dimatikan (*shutdown)*, eksekusi agen pada mesin itu dapat diperintah untuk berpindah (sebelum host tersebut benar-benar mati) dan melanjutkan operasinya sesuai tugas yang diperintahkan.

Dengan keuntungan *mobile agent* yang pada penjelasan sebelumnya. Dapat dilihat siklus hidup *aglet* sehingga dapat menjalankan tugasnya. Gambar 1 menunjukkan siklus hidup *aglet*.



Gambar 1. Siklus Hidup *Aglet*

1. *Create:* penciptaan sebuah *Aglet*.
2. *Clone:* proses penggandaan sebuah *Aglet.*
3. *Dispatch:* pemindahan sebuah *Aglet* dari satu *context* ke *context* yang lain.

4. *Retract:* proses untuk menarik kembali *Aglet* dari *context* yang sedang berlangsung dan masuk ke *context* yang melakukan permintaan kembali.
5. *Activate:* kemampuan untuk mengembalikan *Aglet* ke dalam *context.*
6. *Deactivate:* kemampuan untuk menghentikan sementara jalannya eksekusi *Aglet* dan menyimpan state *Aglet* dalam penyimpan sekunder.
7. *Dispose:* proses untuk menghentikan jalannya eksekusi *Aglet* yang sedang berlangsung dan mengeluarkan *Aglet* dari *context* yang sedang berlangsung.

Pada penelitian yang dilakukan Ary Mazharuddin Shiddiqi, dkk (2010) jaringan *peer to peer* dibuat dengan menggunakan konsep *semantic overlay networks*. Konsep tersebut merupakan salah satu bentuk arsitektur, yaitu *structured peer-to-peer. Structured peer-to-peer* merupakan salah satu tipe *peer-to-peer* yang diklasifikasikan oleh N. Minar (2002) di mana *peer* dikelompokkan ke dalam struktur *tree* yang bertujuan agar pencarian *peer* dapat dilakukan lebih cepat dan efisien.

Peer to peer merupakan istilah dalam jaringan komputer di mana tiap komputer dapat bertindak sebagai *client* maupun server. *peer to peer* memanfaatkan jumlah *host* yang saling terhubung berkolaborasi untuk meningkatkan kecepatan transfer. Pada aplikasi *Peer to Peer*, terdapat sebuah server yang akan membagi user yang terkoneksi pada server tersebut kedalam sebuah atau beberapa sub jaringan yang lebih kecil. Tujuannya adalah untuk mempercepat proses pencarian *resource*. Sistem *peer-to-peer* yang dikembangkan Ary Mazharuddin Shiddiqi, dkk (2010) menggunakan bahasa pemrograman Java dengan *framework* JXTA yang dikembangkan oleh Li Gong (2001) yang merupakan *framework open source* yang menyediakan fungsi-fungsi standar *peer-to-peer*.

Pertanyaan penelitian yang diajukan dalam penulisan paper ini, adalah bagaimana perbandingan sinkronisasi *file* yang menggunakan *aglets* dengan sinkronisasi *file* yang menggunakan metode *peer to peer*. Rekomendasi apa yang bisa diberikan dari hasil perbandingan antara sinkronisasi *file* yang menggunakan *aglets* dengan sinkronisasi *file* yang menggunakan metode *peer to peer.*

Pelaksanaan penelitian ini dimulai dari beberapa tahapan, di mana setiap tahapan yang satu dengan tahapan lainnya tidak bisa terpisahkan. Tahapan dimulai dari, (1) **Studi Literatur,** yakni mempelajari dan memahami konsep sinkronisasi *file*, penggunaan *mobile agent* dan *peer to peer* untuk melakukan sinkronisasi *file*. Sumber studi literatur didapatkan dari buku-buku literatur, jurnal, dokumen, artikel baik dari media massa dan media online, serta data sejenis lainnya yang terkait dengan permasalahan penelitian. (2) **Komparasi dua metode,** dari hasil literatur secara teori (dengan catatan metode *peer-to-peer* dianalogikan sebagai media untuk melakukan sinkronisasi) dan komparasi data menggunakan metode komparatif, yaitu :

1. Data primer dari penelitian "Sinkronisasi File Menggunakan Mobile Agent Aglets" yang penulis lakukan pada tahun 2013. Data tersebut antara lain:
   - variasi ukuran *file*
   - variasi jumlah *file*
   - variasi ukuran dan jumlah *file*

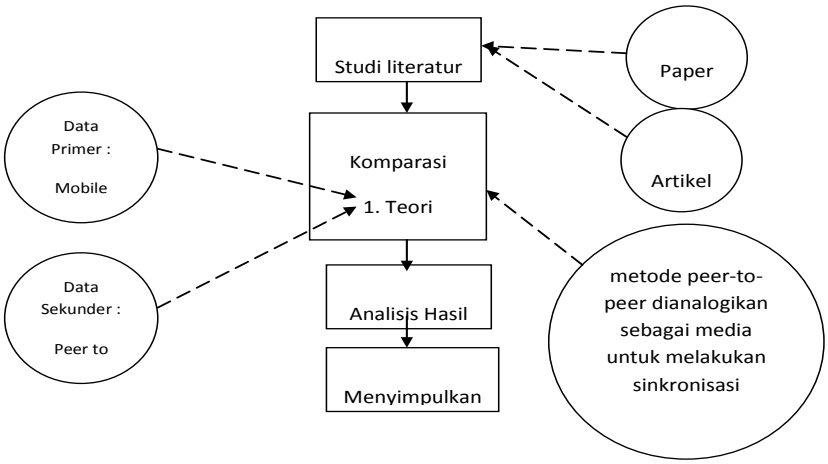
   Pengambilan data memakai dua server di mana dihubungkan oleh sebuah *switch.* Data diambil dengan cara mencatat waktu yang diperlukan oleh agent untuk menyelesaikan sinkronisasi *file*.
2. Data sekunder dari penelitian "Rancang Bangun Jaringan *Peer To Peer* dengan Konsep *Semantic Overlay Networks*" yang dilakukan oleh Ary Mazharuddin Shiddiqi, dkk pada tahun 2010. Data tersebut masih dianggap relevan karena menyajikan data yang lengkap. Data tersebut antara lain:
   - Kecepatan koneksi *peer* ke dalam sistem
   - Kecepatan pencarian *resource*
   - Kecepatan join grup
   - Kecepatan *transfer rate download*

Pengambilan data metode *peer-to-peer* menggunakan 2 (dua) komputer yang dihubungkan secara *peer-to-peer* dengan menjalankan 1 aplikasi node alocator, 18 *edge peer* dan 3 (tiga) *rendezvous peer*.

(3) **Analisis Hasil** yakni, melakukan analisis hasil komparasi antara metode mobile agent dalam melakukan sinkronisasi *file* dibandingkan dengan metode *peer-to-peer* dalam melakukan transfer *file* secara teori maupun dari data penelitian (4). **Menyimpulkan dan memberikan rekomendasi**:

- Memberikan simpulan atas perbandingan dua metode
- Memberikan rekomendasi atas hasil perbandingan penggunaan dua metode mobile aglets dan metode *peer to peer*.

Untuk menjabarkan kerangka konsep dalam pelaksanaan penelitian ini dapat di ilustrasikan dalam *framework* pada Gambar 2.

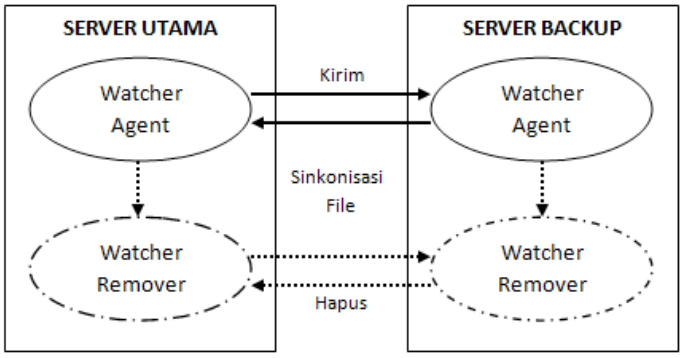


Gambar 2 *Framework* Konsep Penelitian

HASIL DAN PEMBAHASAN

Mobile Agent Aglets

Sinkronisasi file menggunakan *mobile agent Aglets* yang telah dilakukan penulis sebelumnya deskripsinya adalah sebagai berikut. Sinkronisasi dilakukan dengan cara mengirimkan *agent* yang diberi nama *WatcherAgent* ke dua server utama dan *backup*. *WatcherAgent* dapat bertindak sebagai pengirim dan penerima *file*. Mekanisme kirim dan terima *file* dilakukan dengan cara membandingkan daftar *file* pada dua server. Selain itu ada satu *agent* lagi yang diberi nama *WatcherRemover* yang bertindak menghapus *file*. *WatcherRemover* dibuat oleh *WatcherAgent* ketika mendeteksi ada *file* yang dihapus pada salah satu server. Skema dapat dilihat pada Gambar 3.



Gambar 3. Skema Sinkronisasi Menggunakan Mobile Agent Aglets

Pembuatan *agent* dilakukan dengan cara mengembangkan *framework* Aglets. Penambahan fungsi kirim, terima dan hapus dikembangkan sesuai bahasa pemrograman Java.

Pengujian dengan Variasi Ukuran File

Pengujian ini dilakukan dengan cara memberikan *file* baru secara berkala dengan kelipatan tertentu. Kemudian mencatat waktu yang dibutuhkan untuk menyelesaikan sinkronisasi *file*. Data menunjukkan kecepatan sinkronisasi *file* yang tidak tetap dengan minimum sekitar 2,54 MB/s, kecepatan maksimum sekitar 4,53 MB/s dan kecepatan rata-rata sekitar 4,13 MB/s. Hal tersebut dikarenakan aplikasi dibuat menggunakan koneksi *stream* menggunakan protokol TCP. Proses pengiriman diawali dengan menyelesaikan pertukaran *handshake message* agar koneksi dapat terjalin. Selain itu pengiriman dikirimkan dalam paket-paket secara berurutan, apabila terjadi kesalahan atau paket hilang perlu dikirimkan ulang. Ini merupakan karakteristik dari protokol TCP.

Pengujian dengan Variasi Jumlah File

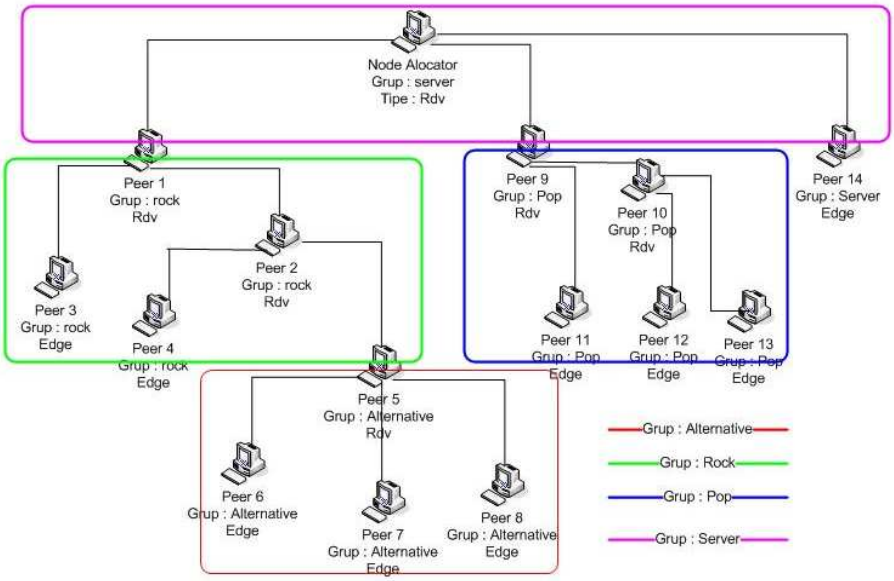
Sama dengan pengujian sebelumnya namun pemberian *file* baru (ukuran sama) dengan variasi jumlah. Data menunjukkan kecepatan sinkronisasi didapatkan semakin banyak jumlah *file* maka kecepatan sinkronisasi cenderung naik. Ini dikarenakan proses sinkronisasi dijalankan secara *multithreading*. *Multithreading* merupakan beberapa *thread* yang dapat melakukan tugas tertentu secara paralel atau konkuren. Setiap *file* disinkronkan oleh satu *thread*. Semakin banyak *file* semakin banyak *thread* dibuat dan *thread-thread* tersebut dieksekusi secara bersamaan. Namun pembuatan *thread* harus memperhatikan juga *resource* (RAM) yang ada, jangan sampai membebani kerja server hanya untuk sinkronisasi file saja.

Pengujian dengan Variasi Ukuran dan Jumlah File

Pengujian ini dilakukan dengan cara memberikan *file* baru dengan jumlah tertentu dengan ukuran yang berbeda-beda dan di-*generate* secara acak. Kemudian mencatat waktu yang dibutuhkan program untuk menyelesaikan sinkronisasi *file*. Data menunjukkan kecepatan yang bervariasi dengan kecepatan minimum sekitar 7,22 MB/s, kecepatan maksimum sekitar 8,98 MB/s dan kecepatan rata-rata sinkronisasi *file* adalah 8,34 MB/s. *Thread* maksimal yang dibuat pada percobaan tersebut jumlahnya adalah 25. Angka tersebut bisa disesuaikan dengan kebutuhan ataupun kemampuan server.

Peer-to-peer

Data pada metode *peer to peer* ini berupa data kecepatan proses koneksi, proses pencarian, proses join group dan kecepatan *transfer rate* proses download. 1 (satu) aplikasi *node alocator,* 18 *edge peer* dan 3 (tiga) *rendezvous peer*. Skema jaringan yang dibuat dapat dilihat pada Gambar 4.



Gambar 4. Skema Jaringan *Peer-to-Peer*

Pengujian Kecepatan Koneksi

Pengujian kecepatan ini menghitung berapa lama waktu yang dibutuhkan untuk tiap peer agar dapat saling terkoneksi. Waktu dihitung mulai dari membangun koneksi dengan *node alocator* hingga *peer* bergabung dengan salah satu grup yang ada. Data menunjukkan proses koneksi antar peer tergantung dari secepat apa peer memperoleh *rendezvous advertisement*. Rata-rata waktu yang dibutuhkan sebuah *peer* terkoneksi adalah 6 sampai 7 detik.

Pengujian Kecepatan Pencarian

Pengujian kecepatan pencarian dalam dua cara berbeda. *Pertama*, pencarian *peer* dalam grup lokal. *Kedua*, pencarian *peer* pada grup lain. Rata-rata waktu yang diperlukan dalam pencarian ini 2 (dua) sampai 3 (tiga) detik untuk kedua cara pencarian.

Pengujian Join Grup

Pengujian join grup menghitung waktu yang diperlukan tiap *peer* untuk join pada grup lain. Data menunjukkan waktu yang diperlukan oleh *peer* untuk bergabung dengan sebuah grup eksternal adalah kira-kira 6 sampai 7 detik.

Pengujian Download File

Pengujian *download file* dilakukan dengan cara download 1 buah *file* secara simultan (didownload bersama) bersamaan dengan *download file* lain. Data menunjukkan waktu rata-rata transfer file adalah 5 sampai 6 detik dan transfer rate rata-rata sebesar 1,1 MB/s.

Analisis Perbandingan

Dari studi literatur dapat disarikan bahwa, penelitian yang penulis lakukan sebelumnya yaitu sinkronisasi *file* menggunakan *mobile agent aglets* masih sebatas pengembangan. Secara paradigma agent dikhususkan untuk menyampaikan pesan dalam hal ini berupa *string* (kumpulan karakter) sebagai informasi atau perintah. Informasi atau perintah tersebut dapat dilanjutkan untuk dieksekusi apabila agent memiliki akses untuk itu. Sebagaimana kita ketahui tiap *host* kini sudah menawarkan pengamanan *firewall* secara default (bawaan *operating system*). *Firewall* ini menutup atau membuka port akses bagi program yang akan berjalan. Jadi apabila otoritas *firewall* tidak mengizinkan maka eksekusi perintah gagal. Selain itu, karena *aglets* dikembangkan menggunakan bahasa pemrograman *java* artinya hanya fungsi-fungsi yang ditawarkan pada *framework* Java saja yang dapat digunakan. Atau dengan cara lain, yaitu mengkodekan sendiri fungsi atau tugas yang diperlukan. Sinkronisasi *file* berjalan dengan cara mengatur tugas agent untuk dapat mengirimkan daftar perubahan file secara berkala. Kemudian informasi tersebut dieksekusi menjalankan fungsi pengiriman file melalui fungsi *java socket* menggunakan koneksi TCP.

Sedangkan penelitian yang dilakukan oleh Ary Mazharuddin Shiddiqi dkk (2010) mencoba mewujudkan jaringan *peer-to-peer* yang pada umumnya dalam bentuk *unstructured peer-to-peer* menjadi bentuk yang terstruktur sesuai dengan pengelompokan tertentu. Konsep tersebut dinamakan *Semantic Overlay Networks*. Konsep yang diterapkan pada percobaan tersebut mengelompokkan *peer-peer* yang memiliki kesamaan konten ke dalam jaringan yang sama. Sehingga *flooding* koneksi dapat diminimalisir dan proses pencarian peer semakin cepat (efisien).

Di antara beberapa keunggulan *mobile agent*, kemampuan *agent* beradaptasi secara dinamis patut dijadikan pertimbangan untuk dapat digunakan sebagai media sinkronisasi file. Kemampuan agent yang dinamis memungkinkan *agent* dapat mengatasi sinkronisasi file yang gagal misalkan tanpa tahu sebabnya. *Agent* dapat ditugaskan untuk mencari tahu kemungkinan-kemungkinan penyebab gagalnya sinkronisasi *file*, mulai dari menyimpan status sinkronisasi terakhir, sampai dengan kondisi terakhir pada host di mana agent berada. Dengan kemampuannya tersebut akan didapatkan informasi yang luas, tidak hanya bagi kelancaran sinkronisasi file namun juga dapat menemukan informasi-informasi lain yang sebelumnya belum diprediksikan.

Di samping keunggulan yang telah dijabarkan, *agent* memiliki beberapa kekurangan antara lain:

Pertama, *Agent* terikat pada lingkungan hidupnya, yaitu *context*. Secara harfiah dapat dijelaskan bahwa agent adalah objek java. Agent dapat berjalan apabila server atau PC perlu terinstal Java *Runtime Environment*. Sebenarnya Java sudah biasa digunakan dan menjadi kebutuhan yang wajar diperlukan bagi PC maupun server. Namun dilihat dari aspek kemudahan dan kebutuhan, bisa jadi pengguna yang tidak memerlukan Java harus menginstal Java hanya untuk menggunakan kemampuan agent tersebut.

Kedua, *Aglets* berjalan pada platform Java, sehingga fungsi pengiriman *file* masih menggunakan Java Socket, yaitu menggunakan koneksi TCP. Tidak menutup kemungkinan menggunakan metode pengiriman lain yang lebih baik, namun tetap menggunakan *mobile agent* sebagai media penyampaian perubahan file dan pemicu otomatisasi kerja.

Ketiga, *Keamanan* pengiriman *file* menggunakan *mobile agent* masih kurang. Aglet sudah menggunakan Protokol koneksinya sendiri yang sudah terenkapsulasi yaitu menggunakan ATP (Aglet Transport Protokol). Namun apabila port dari protokol diketahui ada kemungkinan keamanan data terancam. Solusinya akan lebih aman apabila menambahkan fitur keamanan enkripsi data. Sehingga apabila data berhasil diambil penyadap, data masih dikatakan aman karena perlu metode dekripsi untuk mendapatkan informasi yang dikirim.

Peer-to-peer pada hakikatnya adalah bentuk jaringan lain selain *client-server*. Jaringan *peer-to-peer* menghilangkan konsep *client-server*. *Peer-peer* yang saling terhubung dapat bertindak sebagai client maupun server. Penerapan *peer-to-peer* paling terkenal digunakan sebagai media *file sharing*. Dalam paper ini penulis mencoba menerapkan konsep *peer-to-peer file sharing* sebagai metode sinkronisasi *file*. Contoh *peer to peer* sebagai *file sharing* yang cukup dikenal adalah *Bit torrent*. Penulis mencoba menganalogikan penerapan menggunakan acuan konsep yang dipakai *Bit torrent* ditinjau dari keunggulan dan kelemahannya.

Peer to peer apabila diterapkan sebagai media sinkronisasi file memiliki keunggulan yaitu *Pertama*, metode pengiriman file yang dipakai metode *peer-to-peer* adalah file dipecah-pecah menjadi banyak blok dan pengirimannya dilakukan per-blok. Sehingga apabila pengiriman file berhenti, dapat dilanjutkan dari blok selanjutnya yang perlu dikirim. Ini juga dapat dilakukan jika file sumber berasal dari lokasi lain. Sehingga kontinuitas dan validitas pengiriman file dapat terjaga.

Kedua, apabila jumlah *peer* meningkat, maka kapasitas sistem keseluruhan juga meningkat. Ini menguntungkan apabila digunakan untuk sinkronisasi demi reliabilitas data pada *cluster* (lebih dari satu server) maupun untuk tujuan backup data. Bahkan dengan PC biasa yang berada di luar jaringan lokal server. Asalkan terdapat koneksi internet, sinkronisasi dapat berlangsung.

Penulis tidak menemukan kekurangan berarti *peer-to-peer* pada penggunaan sinkronisasi file. Karena untuk tujuan tertentu, tentu kondisi dan lingkungan sudah diatur sedemikian rupa. Namun apabila berada di luar lingkungan (lokal server) maka keamanan data harus diperhatikan.

PENUTUP

Simpulan

Melihat dua perbandingan metode secara teori (studi literatur) disimpulkan bahwa secara teori dapat diterapkan kombinasi antara *mobile agent* dengan *peer-to-peer*. Di mana *mobile agent* digunakan sebagai pembawa informasi perubahan file serta digunakan sebagai otomatisasi sinkronisasi. Informasi yang dikirim antar *host* berupa alamat *peer* dan *port peer* sehingga tidak perlu melalui proses pencarian *peer*. Ini akan mempersingkat waktu. Kemudian pengiriman *file* ditangani oleh metode pengiriman *file* mengadaptasi cara yang digunakan *Bit Torrent*. Akhirnya kecepatan sinkronisasi dapat ditingkatkan dengan banyaknya server (*cluster*) maupun untuk tujuan *backup* data. Dengan memanfaatkan mobile

agent yang dapat bergerak dengan bebas di mana lingkungannya tersedia memungkinkan diciptakannya sinkronisasi file yang cepat dan tahan terhadap kegagalan seperti yang telah dijelaskan sebelumnya di atas. Namun perlu memperhatikan keamanan yang menjadi kelemahan-kelemahannya.

Dari dua variabel data di atas dapat dianalisis bahwa kecepatan rata-rata sinkronisasi file menggunakan *mobile agent* adalah 8,34 MB/s, sedangkan kecepatan download menggunakan metode *peer to peer* adalah sebesar 1,1 MB/s. Pada kecepatan sinkronisasi menggunakan *mobile agent* sudah memperhitungkan waktu yang dibutuhkan dari dimulainya sinkronisasi sampai selesai. Kecepatan sinkronisasi ini sama dengan pengiriman sejumlah file dalam sekali siklus. Data transfer file menggunakan metode *peer-to-peer* juga mengirimkan file dalam satu kali siklus. Namun perbedaan kecepatan terlihat dari perbandingan kedua metode. Padahal secara teori harusnya transfer file menggunakan *peer-to-peer* bisa lebih cepat, namun hasil penelitian ternyata berbeda. Dari hasil penelitian ini diharapkan dapat dibuat perekayasaan sinkronisasi file menggunakan gabungan dari *mobile agent* dan *peer-to-peer,* sehingga hasil pasti dari penelitian dapat dibuktikan lebih tepat.

DAFTAR PUSTAKA

Calvert, Kenneth L., and Michael J. Donahoo. (2011). *TCP/IP sockets in Java: practical guide for programmer.,* Morgan Kaufmann.

Gong, L. (2001). *Project JXTA: A technology overview.* Technical report, SUN Microsystems. http://www.jxta.org/project/www/docs/TechOverview.pdf

Lange, D.B. and M. Oshima. (1998). *Programming and deploying Java™ Mobile agents with aglets™*. Addison-Wesley.

Lange, Danny B., and Mitsuru Oshima. (1998). *Mobile agents with Java: the Aglet API*, *World Wide Web*, Vol. 1.3, pp. 111-121.

Minar, N. (2002). *Distributed systems topologies: Part 2*. http://www.mba.intercol.edu/MBA731/DistSysTopologies.pdf Diakses 20 Agustus 2015.

Multithreaded Programming Guide. (2008). Sun Microsystems, Inc. https://docs.oracle.com/cd/E19253-01/816-5137/816-5137.pdf. Diakses 2 Desember 2014.

Parameswaran, M., Susarla, A., & Whinston, A. B. (2001). *P2P networking: An information-sharing alternative*. Computer, (7), 31-38.