MULTI-DOMAIN VOIP PEERING DENGAN JARINGAN OVERLAY

Herry Imanta Sitepu

Jurusan Teknik Sistem Komputer

Institut Teknologi Harapan Bangsa

herry@ithb.ac.id

***Abstrak*— Jaringan VoIP yang terdiri dari multi domain membutuhkan mekanisme peering agar dapat melakukan routing panggilan dari satu domain ke domain yang lain. Kelemahan dari solusi VoIP peering yang digunakan saat ini adalah menggunakan arsitektur client/server. Dalam jaringan SIP misalnya, DNS server digunakan sebagai penyedia layanan yang melakukan pencarian (lookup) alamat IP proxy server yang menangani domain tertentu. Arsitektur client/server memiliki masalah skalabilitas, selain ituserver dapat menjadi single point of failure. Penelitian ini mengusulkan sebuah solusi VoIP peering dengan menggunakan arsitektur terdistribusi yang menyediakan layanan lookup melalui sebuah jaringan overlay. Aplikasi ini membutuhkan protokol tabel hash terdistribusi (Distributed Hash Table) sebagai lapisan overlay. Penelitian ini mengusulkan pula UnoHop, algoritma DHT yang efisien dengan kinerja pencarian (lookup) satu hop. Hasil eksperimen memperlihatkan kinerja multi-domain VoIP dengan menggunakan UnoHop dapat beradaptasi dengan event join/leave dalam waktu yang singkat.**

***Kata kunci*— distributed hash table, peer-to-peer, VoIP, multidomain**

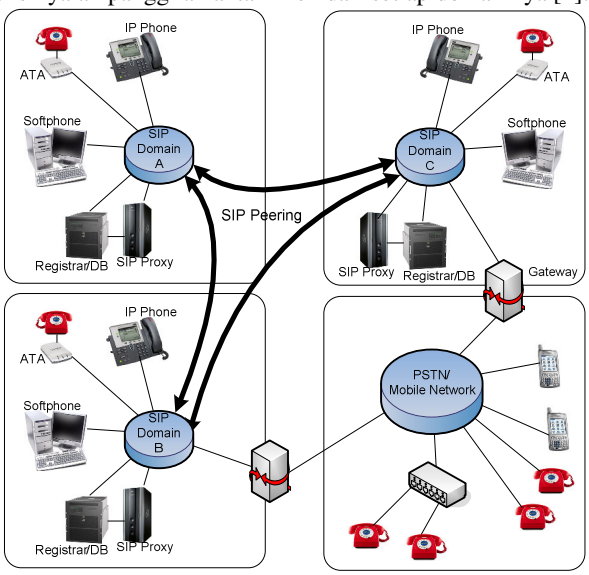
Abstract*— *Multi-domain VoIP network requires peering to allow call routing from one domain to other domains. The problem with current VoIP peering solution is because it uses client/server architecture. In the SIP network for example, the DNS server is usually used to provide services to lookup domain and locate the switching server in a domain. Client/server architecture is not scalable and the server can become the single point of failure. In this paper we propose a distributed architecture for VoIP peering solution by providing lookup services using overlay network. This particular application requires an efficient Distributed Hash Table (DHT) protocol as the overlay layer. We also present UnoHop, an efficient DHT algorithm with one hop lookup performance. Experiments shows that the performance of multi-domain VoIP using UnoHop is adaptable to the join/leave events with short time.

***Keywords*— distributed hash table, peer-to-peer, VoIP, multidomain**

I. PENDAHULUAN

Sebagian besar sistem telepon berbasis Internet (*Voice over Internet Protocol*) saat ini menggunakan arsitektur client/server. Sejumlah klien seperti gateway, *softphone*, atau IP *phone* dikelompokkan menjadi satu domain layanan dan dilayani oleh penyedia jasa layanan (*VoIP service provider*). Gambar 1 memperlihatkan sistem telepon Internet yang terdiri dari beberapa domain dan menggunakan SIP (*Session Initiation* Protocol) sebagai protokol pensinyalan . Konsep yang serupa terdapat pula pada jaringan VoIP yang menggunakan protokol pensinyalan lainnya seperti H.323, IAX2, atau MGCP.

Penyedia jasa layanan VoIP yang melayanai suatu domain menyediakan fungsi *switching* yang sering disebut sebagai *softswitch* untuk melakukan *routing* terhadap panggilan-panggilan ke atau dari domainnya. Fungsi *switching* dapat melakukan *routing* terhadap panggilan yang berasal dari kliennya ke tujuan yang berada di domain lain hanya jika *softswitch* tersebut telah dikonfigurasi sebelumnya untuk mencapat domain-domain tujuan tersebut. Rute panggilan antar domain tersebut dibangun dalam sebuah kesepakatan yang disebut VoIP *peering*, yaitu kesepakatan interkoneksi antar dua penyedia layanan dengan tujuan untuk melakukan pensinyalan panggilan antar klien dari setiap domainnya [2].



Gambar 1. Multi-domain VoIP peering dengan menggunakan SIP sebagai protokol pensinyalan.

Standar yang digunakan untuk melakukan VoIP peering ditetapkan oleh IETF. Standar tersebut disebut sebagai SPEERMINT, yaitu protokol di lapisan ke-lima OSI yang memanfaatkan ENUM [6] sebagai layanan lookup dan SIP sebagai protokol pensinyalan panggilan [4]. Menurut standar

tersebut [5], terdapat empat fungsi utama pada arsitektur yang diusulkannya. *Lookup Function* (LUF) menyediakan mekanisme untuk mengirimkan balasan dari permintaan pensinyalan ke domain tujuan, *Location Routing Function* (LRF) menentukan *Signalling Function* (LRF) pada domain tujuan, *Signalling Function* (SF) melakukan routing panggilan SIP, dan *Media Function* (MF) melakukan fungsi yang berhubungan dengan aliran media seperti melakukan transcoding. Standar tersebut menyarankan agar fungsi lookup dan fungsi routing lokasi sebaiknya menggunakan layanan lookup berbasis DNS seperti ENUM untuk mentranslasikan *request* ke target domain dan *Signalling Function* yang melayani target domain tersebut.

Masalah dengan protokol client/server seperti ENUM lookup yang berbasis protokol DNS adalah bahwa DNS server tersebut dapat menjadi *bottleneck* dan *single point of failure*. Penelitian ini mengusulkan arsitektur VoIP peering yang menggunakan pendekatan terdistribusi untuk menangani isu skalibilitas yang terjadi pada arsitektur client/server. Pada arsitektur yang diusulkan ini, Lookup Function (LUF) dan Location Routing Function (LRF) digantikan oleh lookup dan routing function terdistribusi dengan memanfaatkan jaringan overlay yang dibangun oleh berbagai *switching server* yang berperan sebagai node pada jaringan overlay.

Jaringan overlay adalah jaringan yang terbentuk oleh aplikasi-aplikasi peer-to-peer yang berkomunikasi satu dengan lainnya di atas jaringan fisik berbasis protokol Internet (IP). Setiap peer, atau yang disebut juga sebagai node, dapat berkomunikasi secara langsung dengan peer lainnya untuk memanfaatkan sumber daya secara bersama-sama (*sharing*). Sumber daya tersebut misalnya adalah pemrosesan CPU dan penyimpanan data (*storage*).

Algoritma *Distributed Hash Table* (DHT) adalah infrastruktur yang membangun jaringan overlay yang terstruktur [7]. DHT memungkinkan pemetaan kunci ke node dan menyediakan antarmuka pencarian yang serupa dengan struktur data hash table [7]. Kunci yang merepresentasikan suatu data akan dialokasikan ke node tertentu dengan menggunakan algoritma hashing konsisten. Data yang ingin disimpan di node tertentu terlebih dahulu dihitung untuk mendapatkan nilai kuncinya. Selanjutnya algoritma akan menentukan node mana yang diberikan kepercayaan untuk menyimpan kunci tersebut. Data yang direpresentasikan oleh kunci tersebut dapat disimpan di node tersebut.

Dari beberapa jenis algoritma DHT yang ada, penelitian ini berkonsentrasi dengan algoritma DHT dengan kinerja pencarian $O(1)$, yaitu yang memungkinkan pencarian kunci dapat diselesaikan dengan melakukan satu kali kontak ke node target yang menyimpan kunci tersebut. Hal tersebut dilandasi argumen bahwa proses lookup dijaringan VoIP membutuhkan latensi pencarian yang kecil. Latensi lookup yang besar akan memberikan kontribusi yang signifikan terhadap latensi pensinyalan sehingga akan berdampak pada pelanggan yang tidak puas karena panggilannya membutuhkan waktu yang lama.

Algoritma DHT dengan kinerja pencarian $O(1)$ menggunakan tabel lookup yang lengkap berisi daftar seluruh kunci yang dialokasikan pada seluruh node. Dengan cara tersebut suatu node yang membutuhkan untuk mencari kunci pada node lainnya dapat dengan mudah menemukannya di tabel lookup tersebut, sehingga langkah selanjutnya adalah melakukan kontak secara langsung dengan node yang bertanggung jawab terhadap kunci yang dimaksudkan.

Permasalahan yang muncul dalam perancangan algoritma DHT dengan kinerja pencarian satu hop, adalah bagaimana menangani event yang terjadi sebagai akibat terdapat node baru yang bergabung (*join*) atau node lama yang mati atau terputus dari jaringan (*leave*). Isu utamanya adalah bagaimana mendistribusikan event tersebut secara efisien dan cepat agar seluruh node dapat memperbaharui tabel lookup-nya masing-masing dan selalu merepresentasikan kondisi jaringan yang mutakhir.

Penelitian ini mengusulkan sebuah algoritma berbasis algoritma DHT dengan kinerja pencarian $O(1)$ yang sederhana dan efisien. Hasil simulasi di Bagian 4 memperlihatkan bahwa algoritma DHT yang dihasilkan memiliki karakteristik efisiensi dalam hal latensi distribusi event apabila dibandingkan dengan solusi yang serupa. Sebagai perbandingan penelitian ini menggunakan algoritma OneHop sebagai pembanding.

Arsitektur yang diusulkan pada penelitian ini diimplementasikan pada sebuah lingkungan eksperimen dengan menggunakan emulator jaringan modelnet [8].

Berikutnya, tulisan ini membahasa secara singkat tentang *Distributed Hash Table* di Bagian 2 dan menjelaskan rancangan algoritma DHT UnoHop yang diusulkan pemanfaatannya dalam arsitektur VoIP peering. Bagian 3 memperlihatkan hasil simulasi perbandingan protokol UnoHop dengan protokol OneHop. Bagian 4 memaparkan arsitektur multi-domain VoIP peering berbasis jaringan overlay dengan menggunakan protokol DHT yang dirancang. Bagian 5 menyajikan hasil eksperimen dari implementasi arsitektur VoIP peering yang diperoleh. Bagian 6 memaparkan kenodean dan saran penelitian lanjutan dari penelitian ini.

II. PROTOKOL UNOHOP

Banyaknya penelitian di bidang DHT dalam pengembangan algoritma dan analisis didasari oleh potensi aplikasi yang sangat besar [9][10][11][12]. Aplikasi file sharing yang sangat populer telah membuka jalan yang lebar bagi perkembangan aplikasi peer-to-peer (p2p) diantara pengguna. Dilain pihak, aplikasi potensial yang dapat memanfaatkan sistem p2p tidak terbatas hanya pada aplikasi file sharing saja. Sistem p2p dapat pula digunakan untuk layanan DNS, mobile p2p, distribusi konten, *instant messaging*, komputasi terdistribusi, dan masih banyak aplikasi lainnya [13].

Algoritma DHT pada dasarnya bekerja dengan prinsip yang sederhana, yaitu memetakan kunci-kunci tertentu ke node-node dengan menggunakan suatu prosedur. Kunci-kunci tersebut misalnya dapat merepresentasikan data yang dapat disimpan pada suatu node. Sehingga untuk mendapatkan data-data tertentu, maka suatu node harus dapat menemukan node-node mana saja yang menyimpan kunci-kunci yang

merepresentasikan data-data tersebut. Proses menemukan node yang bertanggung jawab atas kunci tertentu disebut sebagai *lookup*. Hasil akhir dari sebuah proses lookup adalah alamat IP dari node yang bertanggung jawab atas kunci yang dicari dalam sebuah *lookup request*. Alamat IP tersebut selanjutnya dapat digunakan untuk berkomunikasi secara langsung dengan node tersebut untuk mengakses data yang diinginkan. Antarmuka *lookup* dalam menemukan node penyimpan kunci pada algoritma DHT mirip dengan antarmuka strutktur data hash table [7].

Pada bagian ini akan dipaparkan secara singkat cara kerja protokol UnoHop yang dikembangkan pada penelitian ini. Untuk deskripsi yang lebih mendalam dan perbandingannya dengan UnoHop dapat dilihat di tulisan sebelumnya [14].

Protokol UnoHop adalah protokol DHT yang masuk dalam keluarga algoritma dengan kinerja pencarian (*lookup performance*) $O(1)$, yaitu pencarian dapat selalu diselesaikan dengan satu langkah (*one hop*) tanpa bergantung terhadap jumlah node yang tergabung ke jaringan p2p. Kinerja pencarian tersebut dapat dicapai karena setiap node mengelola sebuah tabel (*routing table*) yang berisi informasi lengkap tentang seluruh node yang terkoneksi ke jaringan p2p. Dengan informasi yang lengkap tersebut, setiap node selalu dapat mengetahui kunci-kunci mana saja yang dialokasikan ke node-node tertentu. Pencarian kunci dapat dilakukan dengan mencarinya di tabel routing tersebut.

Permasalahan dengan algoritma DHT yang mengelola tabel rute yang lengkap seperti UnoHop adalah bagaimana mengelola tabel rute yang mutakhir dan merepresentasikan kondisi jaringan terakhir sekalipun selalu terjadi node-node yang bergabung atau keluar dari jaringan. Dinamika node yang bergabung dan keluar jaringan akan selalu terjadi di setiap jaringan p2p. Kemampuan algoritma untuk mengkomunikasikan perubahan tersebut akan menentukan seberapa akurat algoritma tersebut menemukan kunci.

UnoHop memiliki mekanisme yang efisien dalam mendeteksi terjadinya event node bergabung atau keluar (join atau leave). Ketika sebuah node mendeteksi terjadinya suatu event, maka node tersebut bertanggung jawab menjadi inisiator dalam mendisitribusikan informasi tersebut ke seluruh bagian jaringan. Dengan cara tersebut setiap node lainnya akan dapat mengetahui telah terjadi perubahan dan dapat memperbaharui tabel routing-nya sehingga selalu akurat merepresentasikan kondis jaringan terakhir.

Protokol OneHop [12] menggunakan pembagian *ring* menjadi segmen yang tidak terputus yang disebut sebagai *slice*. Selanjutnya setiap *slice* dibagi kembali menjadi segmen yang tidak terputus yang disebut sebagai *unit*.

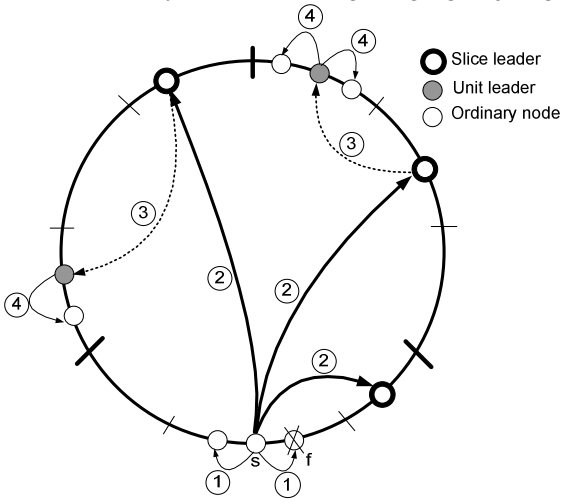
Setiap *slice* direpresentasikan oleh sebuah *slice leader*, yaitu node yang bertugas untuk mengumpulkan event dari *slice* lainnya untuk didistribusikan ke seluruh *unit leader* yang ada di segmennya. Akhirnya, setiap *unit leader* menumpangkan event yang diterimanya dari *slice leader* ke node *predecessor* dan *successor* pada pesan *ping* yang dikirimkan secara periodik. Dengan skema sederhana ini maka lebar pita yang digunakna oleh node biasa menjadi lebih kecil tetapi dengan konsekuensi bahwa lebar pita yang digunakan oleh node *slice leader* dan *unit leader* menjadi lebih besar.

Protokol UnoHop menggunakan segmentasi *ring* yang serupa dengan protokol OneHop [12]. Perbedaannya adalah, UnoHop tidak menetapkan *slice leader* dan *unit leader* dari satu segment. OneHop menggunakan aturan sederhana dalam menentukan *slice leader* dan *unit leader*, yaitu node *successor* dari titik tengah slice dan node *successor* dari titik tengan unit. UnoHop memilih *slice leader* dan *unit leader* secara dinamis sesuai dengan faktor kedekatan tertentu yang akan dijelaskan selanjutnya.

Karena setiap node mengelola tabel rute berisi informasi lengkap seluruh node di jaringan, maka UnoHop menerapkan *Proximity Neighbor Selection* (PNS) seperti yang diusulkan oleh Saroiu [15]. Dengan menentukan hop selanjutnya sebagai node yang paling dekat dengan node pengirim, maka latensi komunikasi dapat di kurangi.

Pendekatan yang sederhana yang dapat digunakan untuk menentukan derajat kedekatan antar node adalah latensi komunikasi antar node pengirim dan node target. Sebuah *slice leader* dipilih sebagai node yang paling dekat dengan node yang mendeteksi event berdasarkan kriteria node dengan latensi paling kecil dibandingkan dengan seluruh node lainnya yang berada di *slice* yang sama.

Sebuah *unit leader* dipilih sebagai node yang paling dekat dalam sebuah *unit* dari *slice leader*-nya. Untuk mendeteksi event, setiap node menjalankan prosedur stabilisasi yaitu dengan mengirimkan pesan *ping* secara periodik dalam interval waktu t_{stab} ke node *successor* dan node *predecessor*. Jika pesan *ping* tersebut menjadi *time out* karena penerima tidak memberi balasan apapun setelah interval waktu tertentu, maka node yang mengirimkan pesan *ping* tersebut mendeteksi bahwa node *predecessor* atau node *successor* telah meninggalkan jaringan. Node yang mendeteksi event tersebut bertugas sebagai inisiator untuk mendistribusikan notifikasi event tersebut keseluruh node di jaringan. Jika notifikasi event belum diterima oleh suatu node, maka node tersebut akan memberikan hasil lookup yang salah karena node yang dimaksud sebenarnya sudah tidak bergabung lagi di jaringan.



Gambar 2. Mekanisme Distribusi Event

Mekanisme pendistribusian event diperlihatkan pada Gambar 2. Jaringan yang terdiri dari 10 node dibagi menjadi 3 *slice* dan setiap slice dibagi menjadi 3 *unit*. Jika node *s* mendeteksi sebuah event, misalnya node *f* telah meninggalkan jaringan karena kegagalan koneksi, ketika menjalankan prosedur stabilisasi (langkah 1), maka berikutnya node *s* harus mengirimkan notifikasi event tersebut ke seluruh segmen lainnya termasuk segmennya sendiri. Untuk mendistribusikan event ke *slice*-nya, node *s* memilih sebuah node yang bertindak sebagai *slice leader* yang selanjutnya akan meneruskan distribusi event. Node *slice leader* tersebut dipilih berdasarkan kriteria kedekatan, yaitu node yang paling dekat dengan node *s* dibandingkan dengan seluruh node-node lainnya yang berada di segmennya. Setelah menemukan node yang bertindak sebagai *slice leader*, selanjutnya node *s* mengirimkan notifikasi event ke node *slice leader* tersebut (langkah 2). Selanjutnya, node *s* menjadi inisiator dalam mendistribusikan event ke *slice* lainnya. Dengan memilih *slice leader* dari suatu *slice* berdasarkan kriteria kedekatan tertentu terhadap node *s*, maka node *s* dapat mengirimkan notifikasi event ke seluruh *slice leader* lainnya.

Jika sebuah *slice leader* menerima notifikasi event tertentu, maka node tersebut harus meneruskan distribusi notifikasi event ke seluruh *unit* yang berada pada *slice*-nya. Setiap *unit* direpresentasikan oleh sebuah node *unit leader*, yaitu node yang memiliki latensi paling kecil terhadap node *slice leader*. Node *slice leader* selanjutnya mengirimkan notifikasi event ke seluruh *unit leader* yang ada di dalam *slice*-nya (langkah 3).

Ketika sebuah node yang bertindak sebagai *unit leader* menerima event, maka node tersebut bertanggung jawab untuk mengumpulkan event-event lainnya selama selang waktu tertentu. Event-event yang telah dikumpulkan dalam selang waktu tertentu selanjutnya dikirimkan ke node *predecessor* dan *successor* dengan menumpangkan notifikasi event tersebut ke pesan ping yang dikirimkan pada prosedur stabilisasi (langkah 4). Event tersebut kemudian diteruskan ke seluruh node dalam satu arah melalui prosedur stabilisasi yang sama yang dilakukan oleh seluruh node. Proses meneruskan notifikasi event akan berhenti pada perbatasan *unit*. Jika sebuah node menerima notifikasi event dari *predecessor*-nya maka node tersebut akan menumpangkan event tersebut ke pesan *ping* yang dikirimkan ke node *successor*-nya. Jika sebuah node menerima notifikasi event dari node *successor*-nya, maka node tersebut akan menumpangkan event tersebut pada pesan *ping* yang dikirimkan ke node *predecessor*-nya. Dengan aturan sederhana tersebut maka duplikasi komunikasi akibat pengiriman event yang sama oleh node lain dapat dihindari.

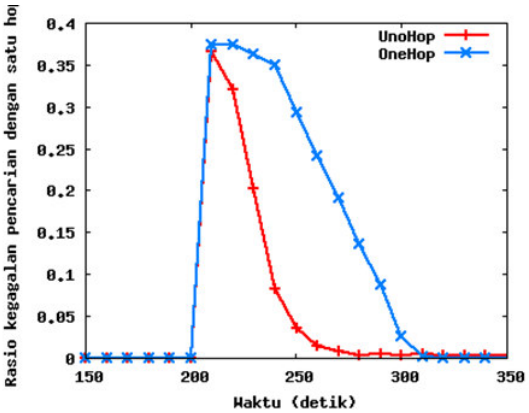
III. SIMULASI

Dalam melakukan simulasi protokol UnoHop yang dikembangkan, penelitian ini menggunakan simulator level paket p2psim [19]. Simulator tersebut mengkhususkan pada simulasi kinerja lookup yang dinyatakan oleh latensi lookup dan rasio kegagalan lookup. Selain itu dapat diperlihatkan pula lebar pita yang digunakan untuk oleh prosedur pengelolaan tabel *routing*. Oleh karena itu, p2psim sangat sesuai untuk melakukan analisa kualitatif terhadap kinerja routing dari algoritma DHT. Simulator dapat mengumpulkan data penting lainnya seperti jumlah total lookup, jumlah lookup yang gagal, penggunaan lebar pita per node, dan data lainnya yang berguna untuk analisis lanjut.

Dalam melakukan simulasi, parameter protokol yang digunakan adalah *slice* berjumlah 10 dan *unit* berjumlah 5 pada setiap *slice*. Penelitian ini menggunakan beberapa topologi jaringan dalam melakukan simulasi. Pertama, simulasi menggunakan jaringan yang terdiri dari 1024 node. Matriks latensi dari 1024 node dikumpulkan dengan menggunakan teknik King yang memanfaatkan DNS query rekursif untuk mengaproksimasi latensi antara dua node [20]. Kedua, simulasi menggunakan jaringan yang terdiri dari 400, 600, dan 3000 node dimana setiap node memiliki koordinat Euclidean yang dipilih secara acak. Latensi antara node dihitung sebagai jarak Euclidean antara dua node. Kedua topologi jaringan yang digunakan dalam simulasi memiliki nilai rata-rata *round trip delay* sebesar 178 mdet. Dalam simulasi, lookup event dibangkitkan secara acak dalam interval waktu yang dipilih secara acak pula, yaitu dengan menggunakan distribusi eksponensial dengan nilai rata-rata 1 lookup per detik.

Untuk memperlihatkan efisiensi distribusi event yang digunakan oleh UnoHop dibandingkan dengan OneHop, maka dalam simulasi dibangkitkan pula leave event pada waktu 200 detik setelah simulasi dimulai yang mengakibatkan 40% dari node meninggalkan jaringan (leave). Node yang leave dipilih secara acak setiap 1 mdet. Untuk jaringan dengan topologi yang terdiri dari 1024 node, seluruh 40% node yang dipilih secara acak akan meninggalkan jaringan setelah 410 mdet. Setelah node-node tersebut leave, maka lookup ke node-node tersebut akan gagal dan meningkatkan rasio kegagalan sistem dan menurunkan kinerja lookup.

Untuk memperlihatkan efisiensi dalam menangani event join, dalam simulasi dibangkitkan event rejoin bagi setiap node-node yang sebelumnya leave. Setiap node di-rejoin setiap selang waktu 1 mdet di sekitar 1000 detik setelah simulasi dimulai. Setelah rejoin, terdapat perpindahan kunci ke node-node yang baru bergabung. Karena node-node lainnya memiliki tabel routing yang lama dan belum mengetahui bahwa node-node baru telah bergabung, maka lookup yang gagal akan bertambah.



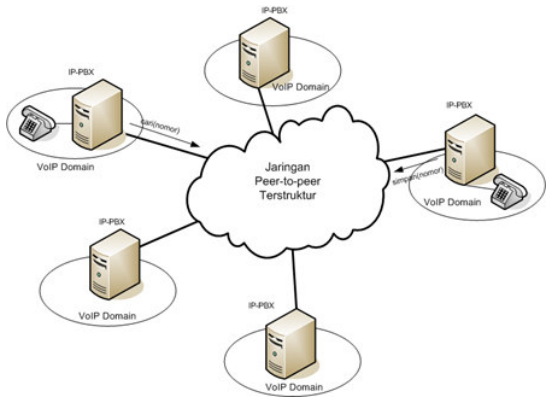
Gambar 3. Perbandingan rasio lookup gagal dengan satu hop setelah 40% node meninggalkan jaringan.

Gambar 3. Memperlihatkan peningkatan lookup gagal pada waktu 200 detik dan 1000 detik, yaitu pada saat 40% node keluar dari jaringan dan saat node-node tersebut bergabung kembali ke jaringan. Peningkatan lookup gagal terjadi karena routing table yang dimiliki oleh node-node lainnya sudah tidak akurat merepresentasikan kondisi jaringan terakhir.

Setelah prosedur distribusi event dijalankan, rasio lookup gagal menurun secara bertahap. Efisiensi dari mekanisme pendistribusian event diperlihatkan oleh waktu yang dibutuhkan oleh protokol untuk memperbaharui routing table dari seluruh node lainnya. Semakin pendek latensi distribusi, maka semakin efisien algoritma pendistribusian yang digunakan. UnoHop membutuhkan 50 detik untuk menurunkan rasio lookup gagal dari 35% ke 0.5%.

IV. ARSITEKTUR

Penelitian ini mengusulkan penggunakan jaringan p2p terstruktur dengan algoritma DHT sebagai infrastruktur pendistribusian dan pencarian nomor dan URI. Setiap server switching, yaitu elemen yang melakukan fungsi switching dan pensinyalan, adalah sebuah node di jaringan p2p. Server switching dapat mempublikasikan seluruh nomor pelanggan yang dilayaninya dengan mengirimkan pesan komunikasi store_route(number). Pesan komunikasi tersebut berisi informasi URI menuju nomor telepon beserta dengan beberapa informasi tambahan lainnya. Informasi tersebut selanjutnya dialokasikan pada node-node lainnya dengan menggunakan algoritma DHT.



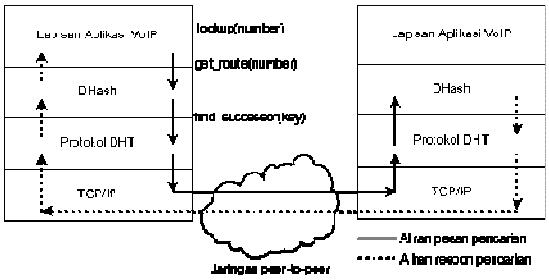
Gambar 4. Jaringan p2p terstruktur sebagai infrastruktur pendistribusian dan pencarian URI berdasarkan nomor telepon.

Jika server switching menerima permintaan panggilan dari pelanggannya, maka pertama-tama server switching memeriksa apakah nomor tujuan ada di domain lokalnya. Jika nomor tersebut ada di domain lokalnya, maka panggilan dapat diteruskan ke tujuan dengan pensinyalan langsung menggunakan protokol VoIP seperti SIP, IAX, atau H.323. Jika nomor tersebut ada di domain eksternal, maka server switching dapatmengirimkan pesan pencarian get_route(number) ke jaringan p2p. Dengan menggunakan protokol DHT, server switching akan menerima respon yang berisi alamat node yang menyimpan informasi tersebut. Selanjutnya, informasi URI dapat diunduh secara langsung dari node yang menyimpan informasi tersebut. Gambar 4. memperlihatkan skema pendistribusian dan pencarian yang dapat dilakukan dengan menggunakan jaringan p2p terstruktur.

Penelitian ini mengusulkan arsitektur terdistribusi untuk mempublikasikan dan melakukan pencarian nomor telepon, URI, dan status registrasi klien. Setiap server switching dan klien berlaku sebagai suatu node di jaringan p2p. Infrastruktur routing di jaringan p2p dibangun oleh protokol DHT. Akan tetapi, protokol DHT hanya menyediakan antarmuka pencarian kunci dan tidak menyediakan fasilitas

penyimpanan data dan replikasi data. Karena itu penelitian ini menggunakan Dhash [16] sebagai lapisan yang menangani antarmuka penyimpanan data dan replikasi data.

Gambar 5 memperlihatkan lapisan perangkat lunak pada sistem pendistribusian dan pencarian nomor telepon, URI, dan status registrasi klien. Lapisan paling atas adalah lapisan yang menangani fungsi pensinyalan dan pengaliran panggilan. Penelitian ini menggunakan aplikasi IP-PBX (IP Private Branch eXchange) Asterisk [17] sebagai implementasi switching server dan SIP client.



Gambar 5. Lapisan aplikasi.

Server Asterisk memiliki basis data yang berisi seluruh nomor lokal yang dilayaninya. Informasi tersebut dapat dipublikasikan ke jaringan p2p pada saat pertama kali aplikasi dijalankan. Selanjutnya, jika terjadi panggilan ke nomor tujuan di luar domain lokal yang dilayaninya, maka switching server melakukan pencarian terhadap rute menuju nomor tersebut dengan memanggil fungsi lookup(number). Selanjutnya, pencarian diteruskan dengan memanggil fungsi get_route(number) yang disediakan oleh lapisan DHash.

Nomor telepon tujuan di-hash dengan menggunakan SHA-1 untuk memperoleh identitas kunci yang dapat dicari oleh protokol DHT. Selanjutnya, lapisan Dhash memanggil fungsi find_successor(key) yang disediakan oleh protokol DHT untuk menemukan node-node yang bertanggung jawab menyimpan kunci tersebut.

Jika protokol DHT dapat menemukan alamat IP dari node-node yang bertanggung jawab menyimpan kunci tersebut, maka DHash akanmelakukan komunikasi dengan node-node tersebut untuk mengambil data yang direpresentasikan oleh kunci. Dalam melakukan replikasi blok data, DHash menggunakan teknik erasure coding untuk menyimpan blok data berukuran 8192 byte ke dalam 14 buah *erasure-coded fragment* berukuran 1171 byte.

Setiap fragment yang diwakili oleh kunci tertentu diletakan pada node tertentu. Lokasi node tersebut ditentukan oleh protokol DHT, yaitu diletakkan pada node suksesor dari kunci. Untuk merekonstruksi kembali blok data, DHash hanya membutuhkan 7 fragmen saja. Setelah DHash dapat menemukan node-node mana saja yang menyimpan data yang direpresentasikan oleh sebuah kunci, selanjutnya DHash melakukan komunikasi dengan 7 node untuk mengambil seluruh fragment yang dapat digunakan untuk merekonstruksi data. Blok data yang diperoleh oleh DHash selanjutnya diproses untuk memperoleh informasi nomor telepon, URI, dan status registrasi klien dari nomor telepon yang dituju.

```
Extension: 7001
Destination: context:passwd@server_ip/7001
Options: nopartial
Status: ok
```

Gambar 6. Format informasi nomor telepon, URI, dan status registrasi klien yang diterima oleh extension logic dari lapisan Dhash.

Gambar 6 memperlihatkan format nomor telepon, URI, dan status registrasi klien yang diterima. Selanjutnya, rute untukmencapai nomor tujuan dapat diekstrak dari field dengan header Destination. Informasi yang terdapat pada field tersebut terdiri dari alamat IP dari server switching yang melayani nomor telepon tujuan beserta informasi keamanan, yaitu username dan password, untuk melakukan panggilan menuju nomor tersebut. Status registrasi klien dapat diekstrak dari field dengan header Status.

V. EKSPERIMEN

Jaringan LAN (Local Area Network) adalah lingkungan yang terlalu ideal dan tidak dapat merepresentasikan jaringan Internet rill sehingga tidak sesuai untuk melakukan pengujian suatu aplikasi p2p. Pada umumnya, jaringan LAN yang baik memiliki karakteristik kapasitas koneksi yang besar (100-1000 Mbps), latensi yang kecil (< 1 mdet), kongesti paket yang minimal, dan rasio hilang paket yang sangat kecil mendekati 0%. Semua kondisi ideal tersebut tidak terdapat pada jaringan Internet riil. Hal tersebut disebabkan karena di jaringan LAN tidak terdapat router dengan ukuran buffer antrian yang terbatas dan kapasitas koneksi antar domain yang beragam. Interkoneksi antar beberapa segmen LAN saja dengan menggunakan sejumlah router masih belum dapat merepresentasikan jaringan Internet di tingkat Autonomous System (AS) karena jumlah router dan domain yang terlalu sedikit jika dibanding-kan dengan kondisi jaringan Internet yang riil.

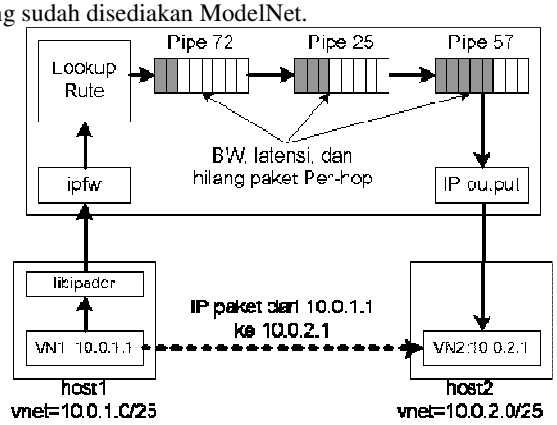
Penelitian ini menggunakan lingkungan emulasi ModelNet [8] yang memungkinkan para peneliti untuk menggelar prototipe aplikasinya di lingkungan yang dapat di kontrol seperti LAN atau testbed kampus. Lingkungan tersebut dapat mengemulasikan jaringan riil Internet dengan ribuan router dan host pada lingkungan LAN. ModelNet juga dapat menerima konfigurasi yang merepresentasikan keadaan jaringan riil seperti parameter latensi, kongesti paket, antrian paket, dan rasio hilang paket. Interkoneksi setiap elemen di lingkungan ModelNet dinyatakan dalam berkas topologi dengan format XML. Setiap tag yang merepresentasikan koneksi dapat memiliki atribut kapasitas lebar pita koneksi, latensi, ukuran antrian, dan rasio hilang paket.

ModelNet [8] dirancang untuk melakukan emulasi di tingkat Internet terhadap sistem terdistribusi seperti aplikasi p2p. Lingkungan yang dibutuhkan oleh ModelNet adalah jaringan LAN yang terdiri dari minimal satu server emulator dan satu host yang terhubung melalui switch LAN dengan kecepatan tinggi (100-1000 Mbps). Semakin banyak jumlah host yang tersedia maka jumlah node yang dapat dijalankan juga semakin banyak. Dengan demikian, peneliti dapat melakukan pengujian terhadap skalabilitas dari aplikasi p2p

atau protokol p2p yang dikembangkannya. Satu server emulator digunakan oleh ModelNet untuk melakukan emulasi trafik dan host lainnya digunakan sebagai node untuk menjalankan aplikasi yang ingin diuji.

Server emulator yang digunakan untuk mengemulasi trafik adalah node inti pada sistem ModelNet. Program ModelNet yang dijalankan di server emulator tersebut memiliki konfigurasi topologi dan routing antar node edge. Jika peneliti membutuhkan emulasi jaringan dengan jumlah node yang sangat besar, maka dapat pula menggunakan beberapa server emulator. Topologi jaringan dan emulasi paket dapat didistribusikan server-server emulator tersebut. Yang perlu diperhatikan jika menggunakan beberapa server emulator adalah munculnya latensi tambahan sebagai akibat komunikasi antar node emulator untuk mengemulasikan trafik antar node yang melalui topologi yang berbeda.

Server emulator dan seluruh host yang terlibat di sistem ModelNet harus memiliki berkas topologi yang sama yang berisi konfigurasi jaringan virtual. Berkas tersebut harus diletakkan di susunan direktori yang seragam. Berkas topologi tersebut berisi seluruh koneksi dan node pada jaringan virtual. Setiap link dapat dikonfigurasi dengan kapasitas lebar pita, ukuran antrian paket, latensi, dan laju hilang paket. Berkas topologi tersebut dapat ditulis dengan manual dalam format XML atau dapat pula dibangkitkan dengan aplikasi generator yang sudah disediakan ModelNet.



Gambar 7. Aliran paket dari satu virtual edge node (VN) ke VN lainnya melalui server emulator ModelNet.

Satu host dapat menjalankan beberapa instan aplikasi yang disebut sebagai virtual edge node (VN). Setiap host memiliki antarmuka fisik ke jaringan LAN. Antarmuka tersebut harus memiliki alamat IP yang berbeda dari alamat IP yang digunakan oleh VN. ModelNet menggunakan IP aliasing untuk memberikan alamat IP yang berbeda bagi setiap VN. Dengan cara tersebut, satu host dapat menjalankan banyak instan aplikasi dengan alamat IP yang berbeda. Seluruh paket yang dikirimkan oleh setiap VN diarahkan ke server emulator untuk diemulasikan seolah-olah paket tersebut melalui jaringan Internet yang terdiri dari banyak domain dan router. Caranya adalah dengan memasang librari khusus pada seluruh host yang tugasnya menangkap (intercept) paket dari alamat IP VN tertentu untuk diteruskan ke server emulator.

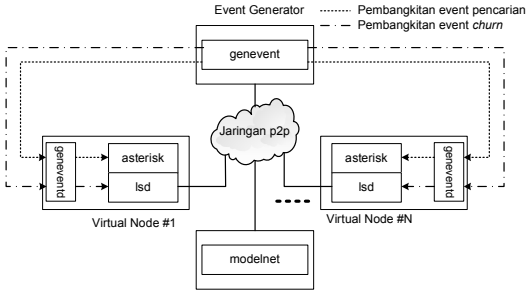
ModelNet menggunakan alamat IP privat 10.0.0.0/8 untuk seluruh node virtual yang dijalankan di satu host. Gambar VI.6 memperlihatkan aliran trafik data dari satu VN ke VN lain. Kedua VN tersebut dapat ada pada host yang sama atau dapat pula dijalankan pada host yang berbeda. Emulasi trafik dilakukan dengan melewatkan paket melalui pipe yang memiliki karakteristik lebar pita, latensi, panjang antrian, dan laju paket hilang tertentu. Setiap pipe tersebut merepresentasikan aliran paket melalui satu link tertentu. Penelitian ini menggunakan satu server emulator dan enam host yang dikonfigurasi seperti yang diperlihatkan oleh Gambar 7.

Topologi yang digunakan pada eksperimen ini dibangkitkan secara otomatik dengan menggunakan generator topologi Internet, Inet-3.0 [18] untuk selanjutnya dikonversikan ke format berkas topologi ModelNet dengan menggunakan aplikasi yang sudah disediakanoleh ModelNet sendiri. Topologi tersebut terdiri dari 4000 core router dan sejumlah node virtual yang didistribusikan secara merata ke enam host. Jika topologi terdiri dari 300 node virtual, maka setiap host menjalankan 50 instan aplikasi.

TABEL 1. KONFIGURASI PADA BERKAS TOPOLOGI YANG DIGUNAKAN DALAM MELAKUKAN EKSPERIMEN.

Parameter	**Nilai**
Jumlah core nodes	4000
Jumlah virtual edge nodes (VN)	100, 200, 300
Jumlah host	6
Kapasitas lebar pita koneksi antara VN dan gateway	512 kbps
Rata-rata latensi antar VN	167 mdet
Kapasitas lebar pita koneksi antar node	150 Mbps

Tabel 1 memperlihatkan konfigurasi topologi yang digunakan dalam melakukan eksperimen. Untuk membangkitkan event pencarian dan leave/join maka penelitian ini membangun sebuah event generator (genevent) yang dapat berkomunikasi menggunakan TCP/IP dengan setiap instan VNmelalui proses *geneventd* dijalankan pada setiap host. Proses generator event tersebut dijalankan di server terpisah agar tidak membebani node emulator dan virtual node. Gambar 8 memperlihatkan hubungan antar proses pada eksperimen.



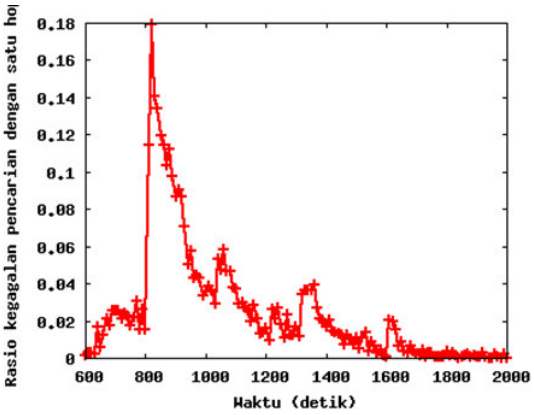
Gambar 8. Proses-proses yang terlibat dalam pembangkitan event pencarian dan event join/leave.

Setiap instan VN terdiri dari dua proses: *asterisk* dan *geneventd*. Proses *asterisk* adalah aplikasi IP-PBX yang dapat melakukan pensinyalan dan switching panggilan telepon Internet. Proses *geneventd* adalah proses yang menjalankan program Dhash dan DHT. Generator event dapat mengirimkan event pencarian ke proses *asterisk* melalui komunikasi TCP/IP dengan proses *geneventd*. Proses *geneventd* menerima event yang berisi nomor tujuan yang dipilih secara acak. Selanjutnya event tersebut dikirimkan ke proses *asterisk* untuk selanjutnya digunakan untuk melakukan panggilan ke nomor tujuan tersebut.

Sementara itu, event event leave/join di bangkitkan oleh *genevent* dan dikomunikasikan dengan *geneventd* yang dijalankan di setiap host. Proses *geneventd* menerima informasi mengenai VN mana yang akan mengalami event keluar atau bergabung. Selanjutnya, *geneventd* akan melakukan eksekusi remote terhadap proses *lsd* untuk menjankan dan menghentikan proses tersebut. Protokol DHT dan DHash akan melakukan mekanisme pendeteksian event dan propagasi event untuk menjaga keakuratan tabel rute di setiap VN lainnya.

Pemrosesan panggilan tertentu pada proses asterisk akan memicu pencarian data di jaringan p2p oleh proses *lsd*. Jika *lsd* dapat menemukan nomor telepon, URI, dan status registrasi klien menuju nomor tujuan, maka asterisk dapat melanjutkan mengalirkan panggilan ke URI yang terdapat pada informasi nomor telepon, URI, dan status registrasi klien yang ditemukan oleh *lsd*. Di awal eksperimen, setiap VN di aktifkan satu persatu dalam interval 1 detik. Hal tersebut dilakukan agarmenghindari lonjakan yang drastis pada utilisasi CPU akibat banyaknya proses yang dijalankan pada waktu yang hampir bersamaan. Eksperimen dilakukan selama 2000 detik dan data-data statistik dikumpulkan mulai dari detik ke-500. Sementara itu pencarian dilakukan setelah detik ke-500.

Generator event membangkitkan event keluar (leave) terhadap sejumlah node yang dipilih secara acak mulai dari detik ke-800. Event keluar tersebut memicu sistem untuk mematikan sejumlah instan proses *lsd* dalam interval setiap 10 mdetik. Setelah detik ke-1600 generator event membangkitkan event bergabung (join) terhadap node-node yang telah keluar pada detik ke-800. Event bergabung tersebut memicu sistem untuk menjalankan kembali sejumlah instan proses *lsd* dalam interval setiap 10 mdetik.



Gambar 9. Rasio lookup gagal dengan satu hop pada eksperimen dengan 200 node.

Pada kondisi normal, yaitu tidak terjadi event leave/join, maka rata-rata latensi pencarian adalah 217 mdet. Sementara itu, dari Tabel 1, latensi rata-rata antar VN adalah 167 mdet. Selisih 50 mdet diakibatkan karena waktu tunda pemrosesan paket yang terjadi di emulator dan waktu tunda pemrosesan yang terjadi di masing-masing host. Kontribusi waktu tunda tersebut diakibatkan karena utilisasi CPU yang cukup tinggi, yaitu rata-rata beban kerja lebih dari 1, jika menggunakan jumlah VN lebih dari 200 node. Selanjutnya nilai rata-rata latensi pencarian sebesar 217 mdet adalah batas bawah pada setiap latensi pencarian dengan menggunakan eksperimen ModelNet.

Gambar 9 memperlihatkan rasio lookup gagal dengan satu hop meningkat pada saat terjadinya sejumlah event keluar. Mekanisme pendistribusian event akan memperbaharui tabel rute di setiap node sehingga rasio lookup gagal akan terus berukurang. Pada detik ke-1600 terjadi sejumlah event bergabung, sehingga rasio lookup gagal menjadi naik kembali.

VI. KESIMPULAN

Protokol DHT adalah sistem terdistribusi yang efisien yang dapat digunakan sebagai infrastruktur pembangun aplikasi. Penelitian ini mengusulkan pemanfaatan DHT dalam melakukan lookup domain VoIP untuk menemukan server switching yang bertanggung jawab menangani routing panggilan di domain yang dituju. Aplikasi dari penelitian ini adalah untuk peering jaringan VoIP multi-domain.

Penelitian ini mengusulkan protokol UnoHop dengan kinerja pencarian satu hop sebagai protokol DHT yang digunakan pada jaringan overlay. Dalam simulasi dengan 1024 node menggunakan simulator p2psim, dilakukan perbandingan antara protokol UnoHop dan OneHop. Untuk melihat kecepatan propagasi event pada kondisi event keluar, dibangkitkan event keluar terhadap 40% node yang dipilih secara acak. Protokol UnoHop dapat mengurangi rasio lookup gagal dengan satu hop ke sekitar 0% dalam waktu 70 detik. Sementara itu, sebagai perbandingan, protokol OneHop membutuhkan waktu 110 detik.

REFERENSI

[1] [1] J. Rosenberg, H. Schulzrinne, U. Camarillo, A. Johnson, J. Peterson, R. Sparks, M. Handley, E. Schooler, "SIP: Session Initiation Protocol", IETF, June 2002.

[2] [2] Mule, J," SPEERMINT Requirements for SIP-based Session Peering", IETF, October 2009.

[3] [3] Uzelac, A., "SPEERMINT Peering Architecture, IETF, November 2009.

[4] [4] Creighton, T., Livingood, J. , "Use of DNS SRV and NAPTR Records for SPEERMINT", IETF, September 2009

[5] [5] Uzelac, A., Lee, Y., "VoIP SIP Peering Use Cases", IETF, January 2010.

[6] [6] Falstrom, P., Mealling, M., "The E.164 to Uniform Resource Identifiers (URI) Dynamic Delegation Discovery System (DDS) Application (ENUM), IETF, April 2004.

[7] [7] S. Androutsellis-Theotokis, dan D. Spinellis, "A survey of peer-to-peer content distribution technologies". ACM Computing Surveys (CSUR) 36: 335–371, 2004.

[8] [8] Vahdat, A., Yocum, K.,Walsh, K., P., Kostic, Mahadevan, D., Chase, J. dan Beck D., "Scalability and Accuracy in a Large-Scale Network Emulator In Proceedings of the 5th ACM/USENIX Symposium on Operating System

[9] Design and Implementation (OSDI). Boston, MA, 2002

[10] [9] Stoica, I., Morris, R., Karger, D., Kaashoek, F. M. dan Balakrishnan, H., "Chord: A scalable peer-to-peer lookup service for internet applications". In SIGCOMM '01: Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, 31. New York, NY, USA: ACM Press, 149–160.

[11] [10] P. Maymounkov and D. Maziãres. "Kademlia: A peer-to- peer information system based on the xor metric". First International Workshop, IPTPS 2002 Cambridge, MA, USA, March 7-8, 2002.

[12] [11] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. D. Kubiatowicz. "Tapestry: A resilient global-scale overlay for service deployment". IEEE Journal on Selected Areas in Communications,22(1):41–53,2004.

[13] [12] A. Gupta, B. Liskov, and R. Rodrigues. "Efficient routing for peer-to-peer overlays". In Proc. First Symposium on Networked Systems Design and Implementation (NSDI '04), March 2004.

[14] [13] John F. Buffor, Heather Yu, Eng Keong Luar, "Peer-to-Peer Networking and Application", Morgan Kauffman, 2009.

[15] [14] Herry Sitepu, Carmadi Machbub, Armein Z.R. Langi & Suhono H. Supangkat, "UnoHop: Efficient Distributed Hash Table with O(1) Lookup Performance", ITB Journal Vol. 2C No. 1, 2008

[16] [15] Saroiu, S., Goummadi, P. dan Gribble, S., "A Measurement Study of Peer-to-Peer File Sharing Systemdet", In Proceedings of Multimedia Computing and Networking 2002 (MMCN '02). San Jose, CA, USA.

[17] [16] Dabek, F., "A Distributed Hash Table". Ph.D. thesis, Massachusetts Institute of Technology. 2005

[18] [17] Asterisk.org, 2009

[19] [18] Winnick, J. dan Jamin, S., "Inet-3.0: Internet Topology Generator". Tech. rep., University of Michigan, 2002.

[20] [19] Li, J., Stribling, J., Morris, R., Kaashoek, M. F. dan Gil, T. M., "A performance vs. cost framework for evaluating DHT design tradeoffs under churn.", In Proceedings of the 24th Infocom. Miami, FL, 2005.

[21] [20] Gummadi, K. P., Saroiu, S. dan Gribble, S. D. "King: Estimating Latency between Arbitrary Internet End Hosts.", In Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002). Marseille, France. 2002.