Reduksi *Feedback Implosion* pada *Reliable Multicast Protocol* Menggunakan *Backup Node* Terdistribusi

Mahendra Data dan Waskitho Wibisono

***Abstrak*–Salah satu masalah dalam pengiriman data secara massal, seperti pendistribusian *update* perangkat lunak, adalah pemborosan *bandwidth*. *IP multicast* yang reliabel dapat dimanfaatkan sebagai solusi dalam mengatasi permasalahan ini, namun *IP multicast* yang reliabel memiliki peluang *feedback implosion* yang besar ketika terdapat paket yang hilang dalam jumlah besar.**

Dalam penelitian ini, penulis mengemukakan sebuah metode untuk meminimalisir kemunculan *negative-acknowledgment* (NAK) agar dapat menekan peluang terjadinya *feedback implosion*. Pada metode ini, tiap *node* dalam sebuah *multicast group* akan memetakan *node* lain yang berdasarkan tingkat reliabilitas dan menjadikannya sebagai *backup node*. Reliabilitas ini diukur berdasarkan besarnya ketersediaan *bandwidth* dan *packet loss* pada jaringan. *Node* yang mengalami *packet loss*, akan mengirimkan NAK ke *backup node* sesuai dengan urutan reliabilitasnya. Tujuannya agar tidak terjadi *bottleneck* di *node* pengirim, sehingga dapat menurunkan pengiriman NAK yang sama berulang kali.

Dari hasil percobaan, terbukti metode ini dapat menurunkan total NAK yang dikirimkan dengan cara mengurangi peluang *bottleneck* pada *node* pengirim sehingga dapat meningkatkan peluang terbalasnya NAK dan menurunkan jumlah pengiriman ulang NAK dengan signifikan.

***Kata Kunci—IP multicast*, reliabel, *feedback implosion*, NAK, *backup node*, ketersediaan *bandwidth*, *packet loss*.**

I. PENDAHULUAN

PENGIRIMAN data secara masal, seperti pendistribusian *update* perangkat lunak, memiliki banyak tantangan. Salah satunya adalah besarnya kebutuhan *bandwidth* karena *update* atau *patch* perangkat lunak tersebut harus dikirimkan ke banyak *node*, padahal data yang dikirimkan sama [1]. *IP Multicast* berpotensi untuk digunakan pada pendistribusian data masal semacam ini karena dapat menghemat *bandwidth* dan sumber daya jaringan. *IP Multicast protocol* dapat menghemat *bandwidth* karena data dikirimkan ke sebuah *multicast group*, tidak eksklusif untuk tiap *receiver* seperti halnya yang terjadi dalam pengiriman secara *unicast*, sehingga penggunaan sumber daya pada *node* pengirim dapat dikurangi secara signifikan [2]. *Node* pengirim dalam sebuah jaringan *multicast* hanya perlu melakukan satu kali pengiriman untuk seluruh *node* penerima yang tergabung dalam *multicast group*.

Peggunakan *IP multicast* tidak terbebas dari masalah. *IP Multicast* pada awalnya didesain untuk digunakan pada pengiriman yang tidak memprioritaskan reliabilitas seperti *audio video streaming*, dimana pada *audio video streaming* kehilangan beberapa paket data bukan merupakan masalah yang besar. Namun pada pengiriman data yang memerlukan reliabilitas tinggi, kehilangan satu paket saja merupakan masalah besar [3]. Misalnya ketika kita mengirimkan sebuah *file*. *File* harus diterima oleh *node* penerima sesuai dengan *file* aslinya. Kehilangan satu paket dapat menyebabkan *file* menjadi tidak sama atau bahkan tidak terbaca.

Sifat *IP multicast* yang tidak reliabel ini dikarenakan desain *IP multicast* yang menggunakan UDP sebagai *transport protocol*. UDP juga tidak menjamin paket data akan sampai ke tujuan dengan urutan yang sama dengan urutan pengiriman [4]. Desain *IP multicast* menyerahkan pengelolaan reliabilitas pengiriman ini ke *application layer*. Telah banyak penelitian dengan berbagai metode membahas masalah ini, dengan tujuan utama adalah menjadikan *multicast* menjadi sebuah protokol pengiriman data yang reliabel [5]. Secara garis besar pendekatan yang dilakukan dalam penelitian tersebut dapat dikelompokkan menjadi lima, yaitu *ACK-based protocols*, *NAK-based protocols*, *Ring-based protocols*, *Tree-based protocols* dan *FEC-based protocols* [5]-[6].

Dalam penelitian ini penulis mengemukakan sebuah rancangan protokol untuk meminimalisir penggunaan paket *acknowledgement* (ACK) dan paket *negative-acknowledgment* (NAK) sehingga dapat mengurangi peluang terjadinya *feedback implosion* pada *IP multicast*. Metode yang digunakan adalah pengembangan dari *NAK-based protocol* yang dikombinasikan dengan sebuah mekanisme yang memungkinkan tiap *node* dapat mengetahui tingkat reliabilitas dari *node* lain yang tergabung dalam

Mahendra Data adalah Mahasiswa Program Pascasarjana Teknik Informatika Institut Teknologi Sepuluh November, Surabaya, Indonesia (email mahendra.data12@mhs.if.its.ac.id).

Waskitho Wibisono adalah Dosen Program Pascasarjana Teknik Informatika Institut Teknologi Sepuluh November, Surabaya, Indonesia (email waswib@if.its.ac.id).

multicast group tersebut. Berbeda dengan mekanisme NAK pada umumnya yang langsung meminta perbaikan paket yang hilang ke *node* pengirim, maka dalam rancangan protokol baru ini *node* yang mengalami kegagalan penerimaan data akan meminta perbaikan dengan mengirimkan NAK ke *node* lain yang memiliki reliabilitas tertinggi. *Node* lain ini berperan sebagai *backup node* untuk perbaikan data.

Reliabilitas dari sebuah *backup node* diukur berdasarkan ketersediaan *bandwidth* dan *packet loss* dalam jaringan yang terjadi pada *backup node* tersebut. Ketersediaan *bandwidth* yang dimaksud disini adalah besar *bandwidth* yang tersedia dalam sebuah *end-to-end network path*, dimana nilainya ditentukan oleh bagian dari jaringan yang mengalami *bottleneck*, yaitu bagian yang memiliki ketersediaan *bandwidth* yang paling kecil dalam *path* tersebut [7]. Sedangkan *packet loss* adalah banyaknya paket yang tidak sampai pada *node* tujuan. *Packet loss* bisa disebabkan karena besarnya trafik yang terjadi dalam jaringan. Selain itu bisa juga disebabkan karena tingginya akses ke satu *node* sehingga menyebabkan *node* tersebut mengalami *overload*.

Packet loss dipilih sebagai salah satu parameter penilaian reliabilitas *backup node* karena walaupun ketersediaan *bandwidth* antara dua *node* besar, namun bila memiliki *packet loss* yang besar pula maka pengiriman data akan menjadi sia-sia. Maka dari itu kedua parameter ini dipilih untuk mengukur tingkat reliabilitas *backup node* karena bila dua faktor ini memiliki nilai yang baik, maka *backup node* tersebut merupakan *node* ideal bagi *node* lain untuk meminta perbaikan paket tanpa harus meminta langsung ke *node* pengirim.

Tujuan dari metode ini adalah untuk mempercepat proses permintaan kembali data yang mengalami kegagalan pengiriman dan meminimalisir peluang terjadinya *feedback implosion* pada *node* pengirim. Hal ini dimungkinkan karena nantinya tidak semua *node* akan meminta perbaikan paket ke *node* pengirim. Dengan kata lain beban yang diakibatkan pengiriman ulang paket yang hilang dapat dipecah dan tidak terpusat di *node* pengirim saja.

II. PROTOKOL MULTICAST YANG RELIABEL

A. *ACK-based protocols*

Pendekatan ACK-*based* pada dasarnya mengadopsi protokol unicast yang reliabel seperti TCP. *Node* penerima mengirimkan data ke receiver secara *multicast*. *Node* penerima akan membalas dengan mengirimkan ACK secara *unicast* bila menerima data dari *node* penerima atau mengirimkan NAK bila tidak menerima data dari *node* penerima. *Node* penerima baru akan menghapus *cache* data tersebut bila sudah menerima ACK dari semua receiver [5].

Kelebihan dari ACK-*based protocols* adalah sederhana, mudah diimplementasikan dan hemat penggunaan *cache*. Namun kelemahannya adalah kemungkinan terjadinya *feedback* ACK dan NAK *implosion* sangat besar sehingga tidak dapat digunakan untuk *multicast* skala besar [5].

B. *NAK-based protocols*

Pendekatan NAK-*based* memiliki perilaku yang berkebalikan dengan ACK-*based*. *Node* pengirim mengirimkan data ke receiver secara *multicast*. *Node* penerima akan membalas dengan mengirimkan NAK secara *unicast* hanya jika tidak menerima data dari *node* pengirim [5].

Pada perkembangannya NAK-*based protocol* ada yang menggunakan metode *polling*. Bila menggunakan metode *polling*, maka *node* pengirim akan memberikan tanda pada paket-paket tertentu secara periodik. *Node* penerima yang menerima paket dengan tanda tersebut harus mengirimkan ACK kepada *sender* secara *unicast* [5].

Pengembangan lain NAK-*based protocol* selain menggunakan metode *polling* adalah dengan menggunakan metode NACK *suppression*, yaitu *node* penerima akan menunggu dalam periode acak tertentu sebelum mengirimkan NAK kepada *node* pengirim secara *multicast*. *Node* penerima yang menerima NAK untuk paket yang sama dari Receiver lain akan berperilaku seakan telah mengirimkan NAK, sehingga pengiriman NAK dapat ditekan [5].

Kelebihan dari *NAK-based protocols* adalah sederhana, mudah diimplementasikan. Namun kelemahannya adalah boros penggunaan *cache*, peluang terjadinya feedback NAK *implosion* cukup besar bila terdapat banyak *node* yang tidak menerima data dan masih tidak dapat digunakan untuk *multicast* skala besar [5].

C. *Ring-based protocols*

Sebelum pengiriman data dimulai, dilakukan pembagian tanggung jawab pengiriman ACK untuk tiap *node* penerima. Satu *node* penerima hanya bertanggung jawab pada paket-paket yang telah ditentukan (selanjutnya disebut sebagai *token receiver*). *Node* pengirim akan mengirimkan data ke *node* penerima secara *multicast*. *Token receiver* akan membalas dengan mengirimkan ACK secara *multicast* ketika menerima data dari *node* pengirim. *Node* pengirim akan menunggu dalam periode tertentu dan akan melakukan transmisi ulang bila tidak menerima ACK dari *node* penerima. *Node* penerima akan mengirimkan NAK secara *unicast* kepada *token receiver* bila tidak menerima data. Lalu *token receiver* akan mengirimkan NAK secara *unicast* kepada *node* pengirim [5].

Kelebihan dari *ring-based protocols* adalah terhindar dari kemungkinan terjadi feedback ACK dan NAK *implosion*. Selain itu pendekatan *ring-based* relatif dapat digunakan untuk *multicast* skala besar (*scalable*). Namun kelemahannya adalah cukup kompleks dan sulit diimplementasikan. Selain itu pendekatan ini juga boros *cache* dan tidak efisien untuk pengiriman data berukuran kecil [5]. Salah satu contoh penerapan *ring-based protocols* ini adalah *Reliable* Data Center *Multicast* (RDCM) yang dikembangkan oleh Dan Li dkk.[8].

D. *Tree-based protocols*

Sebelum pengiriman data dimulai, dilakukan pembentukan *tree* dari *node* pengirim dan *forwarding router*. Tiap *forwarding router* bertangggung jawab terhadap beberapa *node* penerima atau *forwarding router* lain sehingga menghasilkan struktur berbentuk *tree* yang terdiri dari beberapa *group*. *Node* pengirim mengirimkan data ke *node* penerima secara *multicast*. *Group member* dari *multicast tree* mengirimkan ACK kepada *group leader* secara *unicast* ketika menerima data dari *node* pengirim. *Group leader* akan membuat *resume* ACK dari *group member* lalu mengirimkan satu *resume* ACK tersebut kepada *node* pengirim. Bila ada *group member* yang tidak menerima data, maka NAK akan dikirimkan ke *group leader*. *Group leader* akan membuat *resume* NAK dari *group member* lalu mengirimkan satu *resume* NAK tersebut kepada *node* pengirim [5].

Kelebihan dari *tree-based protocols* adalah terhindar dari kemungkinan terjadi feedback ACK dan NAK *implosion*. Selain itu pendekatan *tree-based* dapat digunakan untuk *multicast* dalam skala besar. Namun kelemahannya adalah cukup kompleks dan sulit diimplementasikan. Selain itu pendekatan ini, peluang terjadinya *delay* cukup besar karena harus dilakukan kompulasi ACK dan NAK terlebih dahulu dari banyak *group* serta pengelolaan bentuk *tree* tersebut harus dilakukan sepanjang waktu pengiriman [5].

E. *Forward Error Correction (FEC) based protocols*

Forward Error Correction (FEC) adalah mekanisme perbaikan data dengan menambahkan sejumlah *parity* yang dapat digunakan untuk perbaikan data yang rusak atau hilang [9]. Misalnya terdapat 10 paket data dan dibuat 2 paket *parity*, bila terjadi kegagalan pengiriman maksimal sejumlah 2 paket data, maka *node* penerima akan mampu memperbaiki data tersebut. FEC-*based protocols* tidaklah berdiri sendiri, dia merupakan tambahan protokol untuk keempat protokol sebelumnya. Tujuan dari penggunaan FEC adalah untuk meningkatkan reliabilitas pengiriman data tanpa harus melakukan proses pengiriman ulang. Metode FEC ini digunakan pada *Fcast Multicast File Distribution* [1].

Fcast sukses diterapkan untuk mengatasi masalah *download* dalam jumlah yang sangat besar atau yang biasa dijuluki sebagai "*midnight madness*". *Midnight madness* adalah kondisi trafik *download* yang sangat besar ketika sebuah perangkat lunak baru saja diluncurkan saat malam hari. Penyebab besarnya trafik tersebut antara lain karena kepopuleran perangkat lunak tersebut, perangkat lunak tersebut merupakan *update* penting atau merupakan *bug fixes* dari perangkat lunak sehingga akan sangat banyak orang ingin mendapatkan perangkat lunak tersebut [1].

Ide dari *Fcast* adalah mengkombinasikan pengiriman data menggunakan *IP multicast* untuk mengurangi jumlah trafik secara signifikan dan teknik *forward error correction* untuk meningkatkan reliabilitas dari *IP multicast*. Pada *Fcast*, data dipecah kedalam beberapa blok data, tiap blok data memiliki *parity* (data *redundant*) yang dapat digunakan untuk memperbaiki data yang hilang pada blok tersebut yang diakibatkan oleh *packet loss*. Pengiriman blok data dan *parity* dilakukan secara acak untuk mengurangi peluang hilangnya data dalam sebuah blok dalam jumlah besar yang nantinya mengakibatkan blok tersebut tidak dapat diperbaiki menggunakan *parity* yang ada [1].

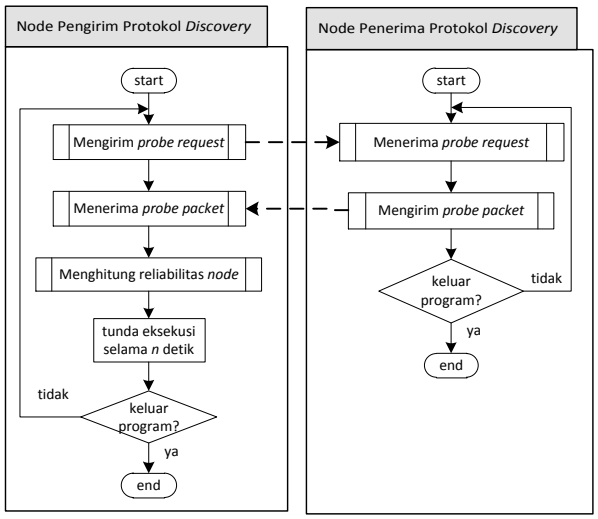
III. PERANCANGAN PROTOKOL

Protokol yang dirancang dalam penelitian ini akan berjalan pada *application layer* di sisi *end node*, tidak perlu mengganti *hardware* atau merubah protokol *IP multicast*, sehingga dapat diimplementasikan dalam jaringan nyata yang telah berjalan saat ini dengan cepat.

Ada 2 protokol yang dirancang dalam penelitian ini, yaitu protokol *discovery* dan protokol pengiriman *file*.

A. *Protokol Discovery*

Protokol *discovery* adalah protokol yang digunakan untuk menemukan *node* lain yang tergabung dalam *multicast group* yang sama. Tujuan dari protokol ini adalah untuk mendapatkan data tingkat reliabilitas *node* lain yang nantinya akan dijadikan sebagai *backup node* saat penerimaan data. Protokol *discovery* akan berjalan terus-menerus dengan periode eksekusi tertentu dan akan berhenti ketika pengiriman data sudah berakhir. Alur dari protokol ini dapat dilihat pada Gambar 1.

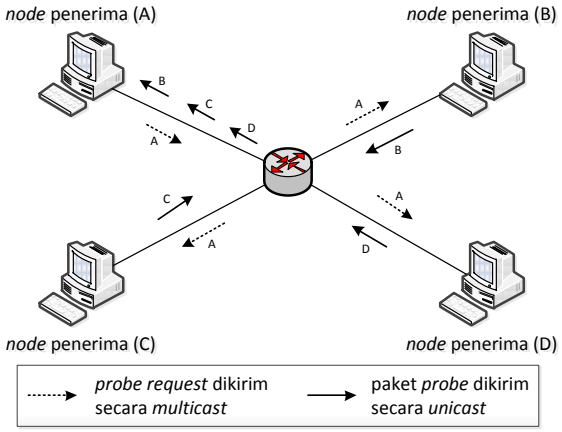


Gambar 1. *Flowchart* protokol *discovery*

Protokol ini diawali dengan pengiriman secara *multicast* sebuah *probe request*, yaitu sebuah paket yang berisi permintaan pengujian *bandwidth* ke setiap *node* dalam *multicast group*. *Node* yang menerima paket *probe request* akan membalas dengan mengirimkan paket *probe* dalam jumlah tertentu secara *unicast*. Paket *probe* inilah yang nantinya akan digunakan untuk menghitung reliabilitas dari *node* lain yang akan dijadikan sebagai *backup node*. Dari proses ini nantinya akan didapatkan sejumlah *N backup node*. Ilustrasi proses ini dapat dilihat pada Gambar 2.

Semua *node* yang tergabung dalam *multicast group* menjalankan protokol *discovery*, kecuali *node* pengirim. *Node* pengirim tidak melakukan pengiriman *probe request*. *Node* pengirim hanya akan merespon *probe*

request dari *node* lain. Hal ini dikarenakan *node pengirim* tidak akan pernah meminta perbaikan paket sehingga tidak perlu mengukur reliabilitas dari *node* lain.



Gambar 2. Ilustrasi protokol *discovery*

Proses *discovery* dilakukan dengan mengirimkan sejumlah *P probe packet* ke alamat *multicast group*. Paket tersebut berisi *probe id*, *total sequence, sequence number*, waktu pengiriman dari *node* pengirim dalam detik sejak *epoc time* dan *dummy payload* yang sengaja ditambahkan untuk menguji *bandwidth availability* pada jaringan. *Node* penerima akan menghitung selisih antar *probe packet* untuk memperkiraan *bandwidth availability* pada jaringan. Selain itu *node* penerima akan menghitung prosentase *packet loss* dari *probe packet* tersebut. Proses ini akan dilakukan secara periodik dalam rentang waktu tertentu.

Tingkat reliabilitas *node* ke-*n* dari sejumlah N *backup node* dinotasikan dengan R_n dengan $(1 \leq n \leq N)$. R_n bernilai antara [0 – 1] yang nilainya ditentukan berdasarkan rumus 1.

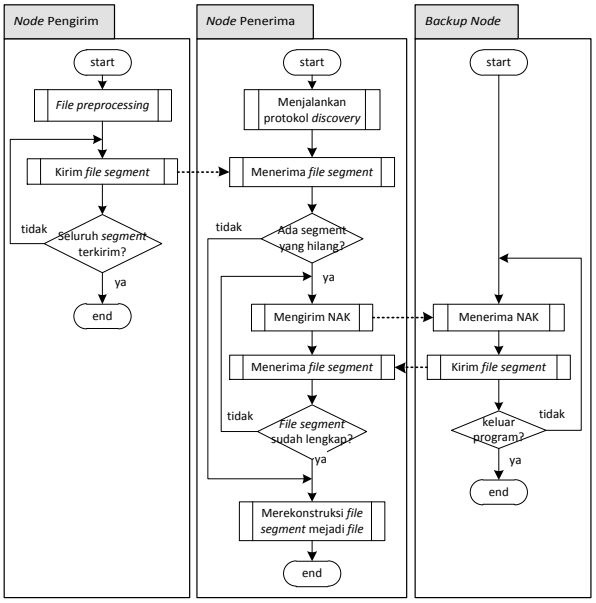
$$R_n = \left(b_1 \times \left(1 - \left(\frac{\Delta D_n}{\max\limits_{i=1\ldots N} \Delta D_i} \right) \right) \right) + \left(b_2 \times \frac{(P - l_n)}{P} \right) \quad (1)$$

ΔD adalah selisih waktu pengiriman dan kedatangan antar *probe packet* pada tiap *node*. *P* adalah jumlah total *probe packet* yang dikirimkan dan l_n adalah jumlah *probe packet* yang hilang (*loss*). b_1 adalah bobot nilai untuk prosentase selisih waktu antar *probe packet* sedangkan b_2 adalah bobot nilai untuk prosentase *packet loss* yang terjadi, dimana $(b_1 + b_2 = 1)$.

B. Protokol Pengiriman File

Protokol pengiriman *file* adalah protokol yang digunakan untuk mengirim data dari *node* pengirim ke *node-node* penerima. Protokol ini melibatkan tiga jenis *node*, yaitu *node* pengirim, *node* penerima dan *backup node*. Tujuan dari protokol ini adalah mengirimkan *file* dari *node* penerima ke *node* penerima dan mendistribusikan NAK ke *backup node* sehingga tidak terjadi *feedback implosion* pada *node* pengirim. Secara garis besar alur protokol dapat dilihat pada Gambar 3.

Tiap segmen hasil *preprocessing* ini akan dikirimkan menggunakan protokol *multicast* dan akan diterima oleh seluruh *node* penerima yang tergabung ke dalam *multicast* group tersebut. Proses ini akan dilanjutkan hingga seluruh segmen terkirim.

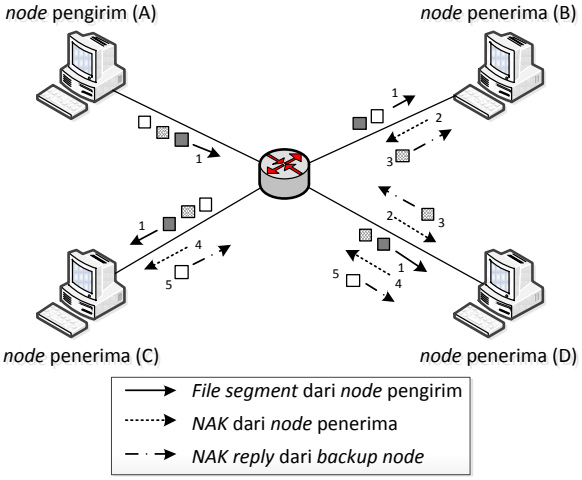


Gambar 3. Flowchart protokol pengiriman *file*

Tiap *node* penerima akan menjalankan protokol *discovery* agar dapat mengetahui *node-node* lain yang reliabel untuk dijadikan sebagai *backup node*. Urutan *backup node* bisa berbeda pada tiap *node* penerima. Hal ini disebabkan karena hasil perhitungan pada protokol *discovery* akan bervariasi untuk tiap *node* penerima.

Setelah protokol *discovery* dijalankan menggunakan *thread* yang terpisah, *node* penerima akan bersiap untuk menerima seluruh segmen *file* yang dikirimkan oleh *node* pengirim. Bila terdapat paket yang hilang saat pengiriman, maka *node* penerima akan mengirimkan NAK ke *backup node* secara *unicast*, dimulai dari *backup node* dengan tingkat reliabilitas tertinggi.

Backup node yang menerima NAK akan membalasnya dengan mengirimkan segmen yang diminta tersebut secara *unicast*. Namun bila tidak memiliki segmen yang diminta, maka *backup node* tidak akan membalas NAK tersebut.



Gambar 4. Ilustrasi protokol pengiriman *file*

Node penerima akan menerima balasan NAK dari *backup node* dan mengecek apakah masih terdapat

segmen yang hilang. Bila masih ada segmen yang hilang maka *node* penerima akan mengirimkan NAK lagi ke *backup node* lain dengan reliabilitas yang lebih rendah. Proses ini akan berlangsung hingga seluruh segmen diterima atau tidak ada lagi *backup node* yang dapat memberikan segmen yang hilang tersebut. Ilustrasi dari proses ini dapat dilihat pada Gambar 4.

IV. HASIL DAN ANALISIS

Pada penelitian ini, pengujian dilakukan menggunakan 12 node yang terbagi ke dalam 6 subnet yang berbeda. Selain itu ditambahkan pula 3 buah router yang bertugas melakukan routing pada tiap subnet sekaligus membangkitkan *packet loss* dan *delay* pada beberapa subnet. Tujuan dari pembangkitan *packet loss* dan *delay* ini adalah untuk mensimulasikan kondisi pada jaringan nyata.

Untuk membandingkan kinerja antara metode baru yang diusulkan dengan metode NAK yang umum digunakan, ada beberapa parameter hasil pengujian yang dibandingkan dan dianalisa, yaitu:

- Rata-rata NAK yang seharusnya dikirim dan rata-rata pengiriman ulang NAK pada tiap *node.*
- Rata-rata total waktu *download*.
- Rata-rata *bandwidth* yang terpakai pada tiap *node.*

Pada uraian selanjutnya metode baru yang diusulkan akan disebut dengan metode perbaikan terdistribusi, sedangkan metode NAK umum yang mengirimkan NAK langsung ke *node* pengirim disebut dengan metode perbaikan tersentral.

A. *Perbandingan rata-rata pengiriman ulang NAK*

Hasil percobaan menunjukkan bahwa metode perbaikan terdistribusi mampu menekan jumlah pengiriman ulang NAK dengan signifikan bila dibandingkan dengan metode perbaikan tersentral. Pada *file* berukuran 32 MB, metode perbaikan terdistribusi melakukan pengiriman ulang NAK sebanyak 18.6% dari jumlah NAK yang seharusnya dikirimkan, sedangkan metode perbaikan tersentral melakukan pengiriman ulang NAK sebanyak 112.4%. Prosentase ini semakin meningkat seiring dengan peningkatan ukuran *file* yang dikirimkan seperti yang terlihat pada Tabel I.

TABEL I
PERBANDINGAN PROSENTASE PENGIRIMAN ULANG NAK

Ukuran File (MB)	Perbaikan Tersentral (%)	Perbaikan Terdistribusi (%)
32	112.4	18.6
64	177.4	20.9
128	248.1	28.2
256	511.1	29.7
512	733.4	79.2

Hasil ini diperkuat dengan hasil pengujian pada jaringan yang mengalami *packet loss* dengan prosentase tertentu seperti pada Tabel II. Pada saat terjadi packet loss sebesar 20%, pada metode perbaikan tersentral terjadi pengiriman ulang NAK sebanyak 965.3% dari jumlah NAK yang seharusnya, sedangkan pada metode perbaikan terdistribusi hanya terjadi pengiriman ulang NAK sebanyak 215.7%.

TABEL II
PERBANDINGAN PROSENTASE PENGIRIMAN ULANG NAK

Besar Packet Loss (%)	Perbaikan Tersentral (%)	Perbaikan Terdistribusi (%)
0	248.1	28.2
10	718.6	107.6
20	965.3	215.7

Dari data ini dapat disimpulkan bahwa dengan mendistribusikan pengiriman NAK terbukti dapat mengurangi peluang *bottleneck* pada *node* pengirim sehingga meningkatkan peluang terbalasnya paket NAK yang dikirim. Pada metode perbaikan tersentral, semakin besar *file* yang dikirimkan semakin besar pula *bottleneck* yang terjadi, sehingga jumlah pengiriman ulang NAK semakin meningkat tajam. Namun hal ini tidak terjadi pada metode perbaikan terdistribusi, dimana NAK didistribusikan ke *backup node* yang berbeda antara satu *node* dengan *node* yang lain.

B. *Perbandingan rata-rata total waktu download*

Tabel III merupakan data rata-rata total waktu *download*. Dari data tersebut terlihat bahwa metode perbaikan terdistribusi memiliki total waktu *download* yang lebih sedikit bila dibandingkan dengan metode perbaikan tersentral. Ketika ukuran *file* semakin besar, selisih waktu dari kedua metode ini juga semakin membesar. Waktu *download* pada *file* berukuran 32 MB, selisih waktu *download* adalah 3.4 detik sedangkan waktu *download file* berukuran 512 MB adalah 30.0 detik. Sehingga dapat disimpulkan bahwa dengan meningkatkan peluang terbalasnya paket NAK yang dikirim dapat menurunkan waktu *download* dari sebuah *file*.

TABEL III
PERBANDINGAN RATA-RATA WAKTU DOWNLOAD

Ukuran File (MB)	Perbaikan Tersentral (detik)	Perbaikan Terdistribusi (detik)
32	26.8	23.4
64	46.1	36.7
128	77.1	63.8
256	147.8	122.4
512	262.7	232.7

TABEL IV
PERBANDINGAN RATA-RATA WAKTU DOWNLOAD

Besar Packet Loss (%)	Perbaikan Tersentral (detik)	Perbaikan Terdistribusi (detik)
0	77.1	63.8
10	96.1	71.5
20	135.5	85.0

Pada saat pengujian menggunakan pada jaringan yang mengalami *packet loss* dengan prosentase tertentu, metode perbaikan terdistribusi memiliki waktu download yang lebih optimal bila dibandingkan dengan metode perbaikan tersentral berapapun besar packet loss yang terjadi. Data hasil pengujian ini bisa dilihat pada

Tabel IV.

C. *Perbandingan rata-rata penggunaan bandwidth*

Tabel V menunjukkan bahwa metode perbaikan terdistribusi menggunakan *bandwidth* yang lebih banyak daripada metode perbaikan tersentral. Perbedaan yang paling mencolok terletak pada pengiriman paket *discovery*. Metode perbaikan terdistribusi menggunakan *bandwidth* yang cukup banyak dalam melakukan proses pencarian *node* reliabel, sedangkan metode perbaikan tersentral tidak memerlukan proses ini sehingga tidak ada *bandwidth* yang terpakai untuk proses tersebut.

TABEL V
PERBANDINGAN RATA-RATA PENGGUNAAN BANDWIDTH

Ukuran File (MB)	Perbaikan Tersentral (MB)	Perbaikan Terdistribusi (MB)
32	33.069	40.956
64	66.176	76.799
128	132.467	147.784
256	265.110	292.116
512	530.411	581.327

Fenomena ini juga terjadi ketika pengujian menggunakan pada jaringan yang mengalami *packet loss* dengan prosentase tertentu. Metode perbaikan terdistribusi menghabiskan *bandwidth* lebih banyak dengan prosentase yang cukup signifikan. Hasil lengkap dari percobaan ini dapat dilihat pada Tabel VI.

TABEL VI
PERBANDINGAN RATA-RATA PENGGUNAAN BANDWIDTH

Besar Packet Loss (%)	Centralized Recovery (MB)	Decentralized Recovery (MB)
0	132.467	147.784
10	131.078	151.736
20	127.972	158.25

V. KESIMPULAN

Dari hasil percobaan dan analisa yang telah dilakukan, dapat ditarik beberapa kesimpulan. Kesimpulan yang pertama adalah metode perbaikan terdistribusi dapat mengurangi peluang *bottleneck* pada *node* pengirim sehingga meningkatkan peluang terbalasnya NAK yang dikirim. Dampaknya adalah menurunnya pengiriman ulang NAK sehingga mengurangi jumlah total NAK yang dihasilkan.

Kesimpulan yang kedua adalah metode perbaikan terdistribusi memiliki total waktu pengiriman yang lebih sedikit bila dibandingkan dengan metode perbaikan tersentral. Hal ini disebabkan karena dengan peningkatan peluang terbalasnya NAK, maka waktu yang terbuang akibat proses pengiriman ulang dapat dikurangi, sehingga total waktu pengiriman *file* dapat berkurang.

Kesimpulan yang ketiga adalah teknik *discovery* untuk pencarian dan pengukuran reliabilitas *backup node* yang digunakan pada metode perbaikan terdistribusi menghabiskan cukup banyak *bandwidth*. Hal ini disebabkan karena untuk menentukan *bandwidth availability* tiap *node* akan melakukan *flooding probe packet* ke jaringan. *Bandwidth availability* dihitung dari hasil *flooding probe packet* paket ini.

Kesimpulan yang keempat adalah metode perbaikan terdistribusi memiliki kemampuan menangani *packet loss* yang lebih baik daripada metode perbaikan tersentral. Pada saat *packet loss* yang semakin besar, prosentase NAK yang dibangkitkan tidak langsung meningkat secara tajam. Hal ini sangat berbeda dengan metode perbaikan tersentral yang peningkatan NAK-nya melonjak tajam seiring dengan semakin besarnya *packet loss* yang terjadi.

REFERENCES

[1] J. Gemmell, J. Gray, dan E. Schooler, "Fcast multicast file distribution," *IEEE Network*, vol. 14, no. 1, hal. 58–68, 2000.

[2] Y. Chawathe, S. A. Fink, S. McCanne, dan E. A. Brewer, "A proxy architecture for reliable multicast in heterogeneous environments," dalam *Proceedings of the sixth ACM international conference on Multimedia*, New York, NY, USA, 1998, hal. 151–159.

[3] D. Koutsonikolas, Y. C. Hu, dan C.-C. Wang, "Pacifier: high-throughput, reliable multicast without 'Crying babies' in wireless mesh networks," *IEEE/ACM Trans. Netw.*, vol. 20, no. 5, hal. 1375–1388, Okt. 2012.

[4] J. Postel, "User Datagram Protocol," 28-Aug-1980. [Online]. Available: http://tools.ietf.org/html/rfc768. [Diakses: 05-Sep-2013].

[5] R. G. Lane, S. Daniels, dan X. Yuan, "An empirical study of protokol reliable multicast protocol over Ethernet-connected networks," dalam *International Conference on Parallel Processing, 2001*, 2001, hal. 553–560.

[6] M. Wu, S. S. Karande, dan H. Radha, "Network-embedded FEC for optimum throughput of multicast packet video," *Signal Processing: Image Communication*, vol. 20, no. 8, hal. 728–742, Sep. 2005.

[7] K. Koitani, G. Hasegawa, dan M. Murata, "Measuring available bandwidth of multiple parts on end-to-end network path," dalam *2012 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*, 2012, hal. 1–6.

[8] D. Li, M. Xu, M. Zhao, C. Guo, Y. Zhang, dan M.-Y. Wu, "RDCM: Reliable data center multicast," dalam *2011 Proceedings IEEE INFOCOM*, 2011, hal. 56–60.

[9] L. Vicisano, M. Luby, M. Handley, J. Gemmell, J. Crowcroft, and L. Rizzo, "The Use of Forward Error Correction (FEC) in Reliable Multicast," Dec-2002. [Online]. Available: https://tools.ietf.org/html/rfc3453. [Diakses: 06-Sep-2013].