Metode Deteksi Terputusnya Koneksi *Tcp* Pada *Receiving Host* Berdasarkan *Packet Inter-Arrival Timeout*

Pangestu Widodo[1], Waskitho Wibisono[2]
[1,2] Program Studi Pascasarjana Teknik Informatika, Fakultas Teknologi Informasi,
Institut Teknologi Sepuluh Nopember
Email: [1]idemahal@gmail.com, [2]waswib@if.its.ac.id

Abstract. Novel Method to Detect Failed TCP Connection on Receiving Host Based on Packet Inter-arrival Timeout. *TCP use retransmission timeout (RTO) as a standard mechanism to detect connection failure. Nevertheless, RTO can only be used by the sending host, and not by the receiving host. Until today there is no standardized mechanism for the receiving host to detect connection failure. Meanwhile a study on internet traffic shows that majority of internet traffic is unidirectional. This means the receiving host in majority of internet traffic does not have a standardized method to detect connection failure. This paper propose a novel method to detect failed TCP connection based on history of packet inter-arrival time. Simulation using NS2 shows the effectiveness of the proposed method.*

.

Keywords: *Failed TCP connection detection, receiving host, packet inter-arrival timeout*

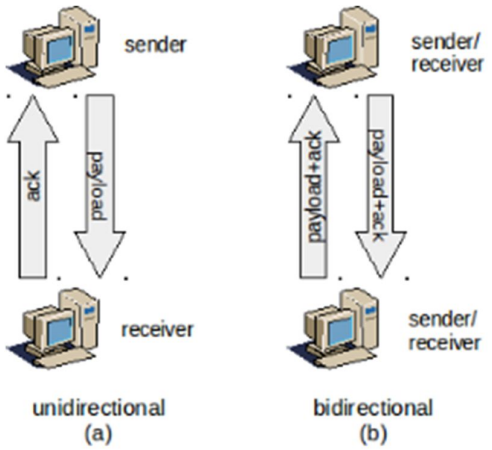
Abstrak. *TCP menggunakan retransmission timeout (RTO) sebagai mekanisme standar untuk mendeteksi terputusnya koneksi. Namun RTO hanya dapat dimanfaatkan pihak pengirim data (sending host), sedangkan pihak penerima data (receiving host) tidak memiliki mekanisme untuk mendeteksi terputusnya koneksi. Sementara itu sebuah studi menunjukkan bahwa sebagian besar traffic internet bersifat satu arah. Ini berarti pihak penerima data pada sebagian besar traffic internet tidak memiliki metode yang terstandarisasi untuk mendeteksi terputusnya koneksi. Penelitian ini mengajukan sebuah metode baru bagi receiving host untuk mendeteksi terputusnya koneksi TCP berdasarkan jeda waktu antar paket data (packet inter-arrival timeout) yang diterima. Simulasi menggunakan NS2 menunjukkan efektivitas metode yang diajukan.*

Kata Kunci: *Deteksi terputusnya koneksi TCP, receiving host, packet inter-arrival timeout*

1. Pendahuluan

Dewasa ini perangkat bergerak yang memiliki lebih dari satu network interface sudah menjadi hal yang biasa. Meskipun demikian, pada umumnya perangkat bergerak saat ini hanya memungkinkan penggunaan sebuah network interface saja setiap saat untuk terhubung ke internet. Ketika koneksi yang menggunakan sebuah network interface terputus, maka perangkat bergerak harus segera melakukan peralihan ke network interface lain untuk membuat koneksi baru atau melanjutkan koneksi yang lama jika memungkinkan. Keputusan peralihan ke network interface lain ini harus dilakukan secepat mungkin agar dapat menjaga kualitas koneksi yang dirasakan oleh pengguna. Semakin cepat perangkat bergerak mendeteksi terputusnya koneksi, semakin kecil gangguan koneksi yang dirasakan oleh pengguna.

TCP (Transport Control Protocol) merupakan salah satu protokol pada transport layer yang paling banyak digunakan di internet. Koneksi TCP dikatakan bersifat dua arah (bidirectional, Gambar 1b) karena kedua pihak yang terhubung dapat mengirimkan data atau menerima data sekaligus. Namun demikian, penelitian yang dilakukan oleh Sinha dkk (Sinha dkk, 2005) menunjukkan bahwa sebagian besar koneksi internet justru bersifat satu arah (unidirectional, Gambar 1a). Pada koneksi satu arah ini, salah satu pihak berperan sebagai pengirim data (sending host) sementara pihak yang lain disebut sebagai penerima data (receiving host). Ketika receiving host menerima sebuah paket data dari sending host, ia akan mengirimkan sebuah paket acknowledgement (ack) kepada sending host sebagai notifikasi atas diterimanya paket data tersebut.



Gambar 1. Jenis koneksi TCP berdasarkan arah aliran data

Pada TCP standar terdapat metode untuk mendeteksi terputusnya koneksi yaitu retransmission timeout (RTO). Ketika sebuah host mengirimkan data, maka ia akan menjalankan sebuah timer untuk membatasi lama waktu tunggu diterimanya acknowledgement dari receiving host. Ketika terjadi beberapa kali timeout, maka sending host dapat mengggunakan informasi ini dalam proses deteksi terputusnya koneksi (Internet Engineering Task Force, 1989).

Kelemahan penggunaan RTO untuk mendeteksi terputusnya koneksi adalah bahwa RTO hanya dapat digunakan oleh sending host. Sementara bagi receiving host hingga saat ini belum ada mekanisme standar yang dapat digunakan untuk mendeteksi terputusnya koneksi terutama pada tingkat transport layer. Hal ini menjadi penting apabila dihubungkan dengan hasil penelitian yang dilakukan Sinha dkk diatas bahwa sebagian besar traffic internet bersifat satu arah.

Paper ini mengajukan metode baru yang dapat dimanfaatkan oleh receiving host untuk mendeteksi terputusnya koneksi TCP di transport layer. Metode yang diajukan berusaha mendeteksi terputusnya koneksi berdasarkan jeda waktu antar paket data (packet inter-arrival time) yang diterima receiving host.

2. Kajian Literatur

Pada bagian ini akan dibahas mekanisme pada TCP standar untuk mendeteksi terputusnya koneksi. Yang dimaksud dengan TCP standar adalah spesifikasi TCP yang terkandung dalam dokumen-dokumen RFC yang termasuk dalam kategori Standard Protocols, Draft Standard Protocols, atau Proposed Standard Protocols di dalam dokumen Internet Official Protocol Standards STD-1 (RFC Editor, 2008).

2.1 Retransmission Timer

Retransmission Timer merupakan mekanisme pokok dalam *TCP* standar untuk menangani terjadinya *packet loss*. Untuk setiap paket data (*payload*) yang dikirimkan, *host sender* akan menghitung *Retransmission Timeout* untuk paket tersebut, yaitu jangka waktu maksimal diterimanya *acknowledgement* dari *host receiver* untuk paket tersebut. Ketika terjadi *timeout*, *host sender* akan melakukan transmisi ulang paket yang mengalami *timeout*, dan *Retransmission Timer* diulang kembali (*Information Science Institute University of South California*, 1981).

Perhitungan *Retransmission Timeout* (*RTO*) dilakukan berdasarkan *Smoothed Round Trip Time* (*SRTT*) dan *Round Trip Time Variance* (*RTTVAR*). *SRTT* dan *RTTVAR* dihitung berdasarkan *Round Trip Time* (*RTT*) yang terukur dengan perhitungan sebagai berikut sesuai (Paxson dkk, 2011):

1. Untuk *RTT* pertama yang didapat, dilakukan perhitungan sebagai berikut:

$$SRTT = RTT \quad (1)$$

$$RTTVAR = \frac{RTT}{2} \quad (2)$$

$$RTO = SRTT + max(G, 4 \cdot RTTVAR) \quad (3)$$

2. Untuk setiap *RTT* berikutnya, dilakukan perhitungan ulang sebagai berikut:

$$RTTVAR = (1-\beta) \cdot RTTVAR + \beta \cdot (SRTT - RTT) \quad (4)$$

$$SRTT = (1-\alpha) \cdot SRTT + \alpha \cdot RTT \quad (5)$$

$$RTO = SRTT + max(G, 4 \cdot RTTVAR) \quad (6)$$

dimana $\alpha = 1/8$, $\beta = 1/4$ dan G = *clock granularity* pada sistem terkait.

2.2 Penggunaan Retransmission Timer untuk Mendeteksi Terputusnya Koneksi

Retransmission Timeout dapat digunakan oleh *host sender* sebagai dasar untuk mendeteksi adanya kegagalan koneksi (terputusnya *link*), misalnya berdasarkan jumlah transmisi ulang untuk paket yang sama. RFC 1122 (*Internet Engineering Task Force*, 1989) mensyaratkan penggunaan dua buah *threshold*, yaitu R1 dan R2, untuk membatasi jumlah transmisi ulang paket yang sama. R1 adalah batas waktu atau jumlah transmisi ulang secara mandiri oleh *TCP* sebelum meminta *network layer* (*IP*) untuk memeriksa kondisi *path* yang digunakan. R2 yang lebih besar dari R1 digunakan sebagai batas akhir waktu atau transmisi ulang sebelum koneksi ditutup secara paksa. Nilai minimal yang disarankan untuk R1 adalah 3 kali transmisi dan R2 adalah 100ms.

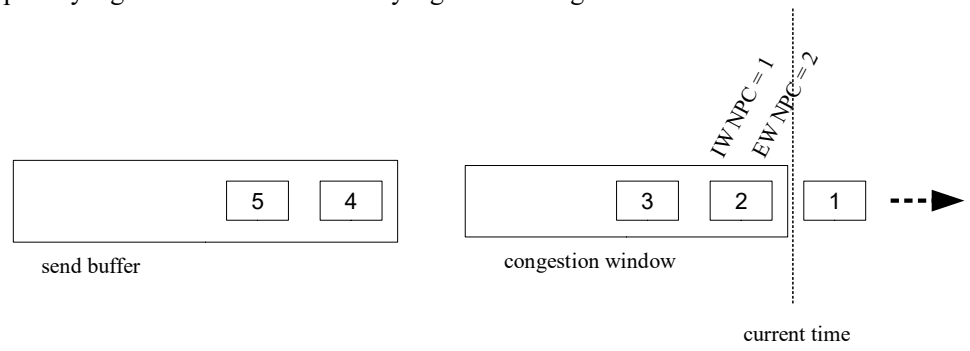
3. Perancangan Metode

Pada bagian ini dijelaskan rancangan metode Packet Inter-arrival Timeout (PITO) yang diajukan dalam penelitian ini.

3.1 Desain Dasar Metode PITO

Berikut ini adalah desain dasar metode yang diajukan:

1. Bagi *receiver*, untuk mendeteksi terputusnya *subflow* dibutuhkan informasi mengenai adanya paket yang akan diterima di waktu yang akan datang.



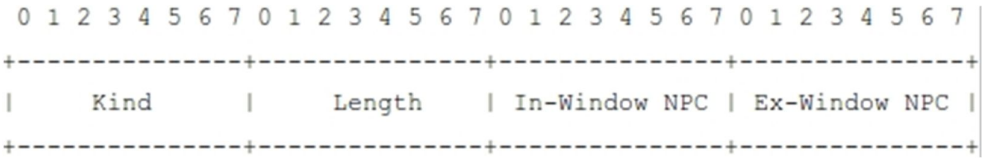
Gambar 2. Jenis koneksi TCP berdasarkan arah aliran data

2. Dalam setiap paket yang akan dikirimkan, *sender* dapat menyisipkan informasi dalam bentuk *TCP option* berapa jumlah paket yang akan dikirimkan setelah paket tersebut (*next packets count*). Informasi yang disisipkan ada dua yaitu *In-Window Next Packets Count (IW NPC)* dan *Ex-Window Next Packets Count (EW NPC)*. *In-Window Next Packets Count* merupakan informasi berapa banyak paket dalam *congestion window* pada *host sender* yang masih akan dikirimkan, sedangkan *Ex-Window Next Packets Count* merupakan informasi berapa banyak paket di luar *congestion window* pada *host sender* (*send buffer*) yang akan dikirimkan. Ilustrasi Perhitungan IW NPC dan EW NPC dapat dilihat pada Gambar 2.

3. Setiap kali *receiver* menerima paket yang berurutan dari *sender*, *receiver* akan menghitung *timeout* untuk paket berikutnya berdasarkan informasi *IW NPC* dan *EW NPC* yang diterimanya, dan mencatat jumlah paket yang akan diterima berikutnya (apabila diperlukan).
4. Apabila tidak ada paket yang diterima *receiver* hingga terjadi timeout, maka *receiver* dapat melakukan aksi sesuai *policy* yang dimiliki. Contoh aksi yang dapat dilakukan misalnya mengaktifkan *subflow backup*, membuat *subflow* baru, mengirimkan sinyal *REMOVE_ADDR* untuk menghapus *subflow* yang mengalami *timeout*, dan lain-lain.

3.2 *Format TCP Option Next Packets Count*

Format TCP option untuk mengirimkan informasi *next packets count* dapat dilihat pada Gambar 3. Field *In-Window NPC* berisi informasi *In-Window Next Packets Count (IW NPC)* dan field *Ex-Window NPC* berisi informasi *Ex-Window Next Packets Count (EW NPC)*.



Gambar 3. ***Format TCP option Next Packets Count***

3.3 Perhitungan *Packet Inter-arrival Timeout*

Perhitungan *Packet Inter-arrival Timeout (PITO)* diadaptasi dari perhitungan *Retransmission Timeout* pada (Paxson dkk, 2011) yang dimodifikasi. Perhitungan *PITO* dilakukan berdasarkan *Smoothed Packet Inter-arrival Time (SPIT)* dan *Packet Inter-arrival Time Variance (PITVAR)*. *SPIT* dan *PITVAR* dihitung berdasarkan *Packet Inter-arrival Time (PIT)* yang terukur.

Terdapat dua macam *timeout* dalam metode *PITO* yaitu *In-Window timeout* dan *Ex-Window timeout*. Kedua jenis *timeout* tersebut dihitung secara terpisah namun menggunakan perhitungan yang serupa yaitu sebagai berikut:

1. Apabila belum ada data *Packet Inter-arrival Time (PIT)* yang didapat, informasi *retransmission timeout* yang dimiliki TCP digunakan sementara hingga didapatkan informasi *Packet Inter-arrival Time.*
2. Untuk *Packet Inter-arrival Time* pertama yang didapat, dilakukan perhitungan sebagai berikut (berlaku untuk *In-Window timeout* maupun *Ex-Window timeout*):

$$SPIT = PIT \tag{7}$$

$$PITVAR = PIT / 2 \tag{8}$$

$$PITO = \gamma \cdot (SPIT + (4 \cdot PITVAR)) \tag{9}$$

3. Setiap kali *receiver* menerima paket dengan *sequence number* yang berurutan dengan paket yang diterima sebelumnya serta memenuhi syarat nilai *In-Window NPC* pada paket sebelumnya > 0 atau nilai PIT yang baru didapatkan lebih kecil dari $PITO_{INW}/2$ maka dilakukan perhitungan ulang *In-Window timeout* sebagai berikut:

$$PITVAR_{INW} = (1 - \beta) \cdot PITVAR_{INW} + \beta \cdot (SPIT_{INW} - PIT) \tag{10}$$

$$SPIT_{INW} = (1 - \alpha) \cdot SPIT_{INW} + \alpha \cdot PIT \tag{11}$$

$$PITO_{INW} = \gamma \cdot (SPIT_{INW} + (4 \cdot PITVAR_{INW})) \tag{12}$$

dimana $\alpha = 1/32$, $\beta = 1/16$, dan $\gamma = 4$.

4. Setiap kali *receiver* menerima paket dengan *sequence number* yang berurutan dengan paket yang diterima sebelumnya serta memenuhi syarat nilai *In-Window NPC* pada paket sebelumnya = 0 dan nilai PIT yang baru didapatkan lebih besar dari $PITO_{INW}/2$ maka dilakukan perhitungan ulang *Ex-Window timeout* sebagai berikut:

$$PITVAR_{EXW} = (1 - \beta) \cdot PITVAR_{EXW} + \beta \cdot (SPIT_{EXW} - PIT) \tag{13}$$

$$SPIT_{EXW} = (1 - \alpha) \cdot SPIT_{EXW} + \alpha \cdot PIT_{EXW} \quad (14)$$

$$PITO_{EXW} = \gamma \cdot (SPIT_{EXW} + (4 \cdot PITVAR_{EXW})) \quad (15)$$

dimana $\alpha = 1/64$, $\beta = 1/32$, dan $\gamma = 1, 1.1, 1.2, \ldots 4$. Penambahan nilai g dilakukan apabila $PITO_{EXW} < PITO_{INW}$.

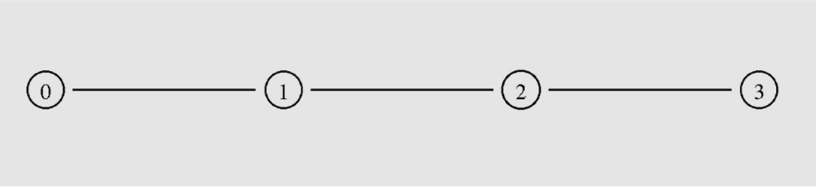
5. Apabila paket yang diterima memiliki sequence number yang melompat atau merupakan awal dari rangkaian paket retransmisi (sehingga tidak terurut) maka tidak dilakukan perhitungan ulang (dilewatkan).
6. Apabila paket yang diterima merupakan retransmisi (karena *retransmission timeout* ataupun *fast restransmit*) maka *Ex-Window timeout* dihitung ulang menggunakan *PIT* pertama yang didapat pada langkah 2 diatas.
7. Apabila paket yang diterima saat ini memiliki *In-Window NPC* = 0 maka $PITO_{EXW}$ digunakan sebagai timer. Sebaliknya apabila nilai *In-Window NPC* > 0 maka $PITO_{INW}$ digunakan sebagai timer.

4. Pengujian Metode

Pada bab ini akan dipaparkan implementasi dan pengujian metode *Packet Inter-arrival Timeout (PITO)*. Pengujian dilakukan dengan mengubah parameter-parameter jaringan seperti *latency*, *jitter*, *packet loss*, dan *bandwidth*. Evaluasi terhadap hasil pengujian disajikan pada bagian pembahasan di akhir bab.

4.1 Topologi Jaringan dalam Pengujian

Topologi jaringan yang digunakan dalam pengujian disajikan pada Gambar 4. Pada jaringan tersebut Node 0 merupakan sending host yang terhubung dengan Node 4 yang merupakan receiving host.



Gambar 4. Topologi jaringan dalam pengujian

4.2 Pengujian Kehandalan Metode PITO

Yang dimaksud dengan pengujian kehandalan disini adalah menguji seberapa besar ketahanan metode PITO terhadap variasi parameter jaringan agar tidak menimbulkan kesalahan deteksi terputusnya koneksi. Parameter yang dimasukkan ke dalam pengujian adalah latency, bandwidth, Maximum Segment Size (MSS), jitter, dan packet loss.

Pada pengujian kehandalan ini, transfer data dilakukan secara kontinyu dari sending host ke receiving host, berikutnya dilakukan variasi nilai terhadap parameter jaringan satu per satu, kemudian dilakukan pengamatan apakah metode PITO mampu melewati periode waktu tertentu tanpa mendeteksi terputusnya koneksi. Apabila dalam periode waktu tersebut ternyata metode PITO mendeteksi terputusnya koneksi (false alarm), maka pengujian tersebut dinyatakan gagal. Pada pengujian ini periode waktu yang digunakan adalah 60 detik.

4.2.1 Latency

Pengujian kehandalan metode PITO terhadap perubahan nilai latency dilakukan dengan melakukan 10 kali pengujian pada setiap nilai latency. Parameter yang digunakan adalah sesuai Tabel 1. Hasil pengujian disajikan pada Tabel 2. Nilai precision pada tabel hasil pengujian dihitung berdasarkan jumlah percobaan yang berhasil (tidak terjadi false alarm) dan jumlah percobaan yang gagal (terjadi false alarm).

Tabel 1. Parameter Pengujian Kehandalan Metode *PITO* terhadap Variasi *Latency*

Parameter	Nilai
Maximum Segment Size	1440 bytes
Bandwidth	10Mbps
Latency	30ms, 150ms
Maximum Jitter	0ms
Packet Loss	0%
Durasi	60s

Tabel 2. Hasil Pengujian Kehandalan Metode *PITO* terhadap Variasi *Latency*

Nilai *Latency*	*Precision*
150ms	1
30ms	1

4.2.2 Maximum Segment Size (MSS)

Pengujian kehandalan metode PITO terhadap perubahan nilai MSS dilakukan dengan melakukan 10 kali pengujian pada setiap nilai MSS. Parameter yang digunakan adalah sesuai Tabel 3. Hasil pengujian disajikan pada Tabel 4. Nilai precision pada tabel hasil pengujian dihitung berdasarkan jumlah percobaan yang berhasil (tidak terjadi false alarm) dan jumlah percobaan yang gagal (terjadi false alarm).

Tabel 3. Parameter Pengujian Kehandalan Metode *PITO* terhadap Variasi *MSS*

Parameter	Nilai
Maximum Segment Size	1440, 536 bytes
Bandwidth	10Mbps
Latency	150ms
Maximum Jitter	0ms
Packet Loss	0%
Durasi	60s

Tabel 4. Hasil Pengujian Kehandalan Metode *PITO* terhadap Variasi *MSS*

Nilai *MSS*	*Precision*
1440byte	1
536byte	1

4.2.3 Bandwidth

Pengujian kehandalan metode PITO terhadap perubahan nilai bandwidth dilakukan dengan melakukan 10 kali pengujian pada setiap nilai bandwidth. Parameter yang digunakan adalah sesuai Tabel 5. Hasil pengujian disajikan pada Tabel 6. Nilai precision pada tabel hasil pengujian dihitung berdasarkan jumlah percobaan yang berhasil (tidak terjadi false alarm) dan jumlah percobaan yang gagal (terjadi false alarm).

Tabel 5. Parameter Pengujian Kehandalan Metode *PITO* terhadap Variasi *Bandwidth*

Parameter	Nilai
Maximum Segment Size	1440 bytes
Bandwidth	10Mbps, 1Mbps
Latency	150ms
Maximum Jitter	0ms
Packet Loss	0%
Durasi	60s

Tabel 6. Hasil Pengujian Kehandalan Metode *PITO* terhadap Variasi *Bandwidth*

Nilai *Bandwidth*	*Precision*
10Mbps	1
1Mbps	1

4.2.4 Jitter

Pengujian kehandalan metode PITO terhadap perubahan nilai jitter dilakukan dengan melakukan 10 kali pengujian pada setiap nilai jitter. Parameter yang digunakan adalah sesuai Tabel 7. Hasil pengujian disajikan pada Tabel 8. Nilai precision pada tabel hasil pengujian dihitung berdasarkan jumlah percobaan yang berhasil (tidak terjadi false alarm) dan jumlah percobaan yang gagal (terjadi false alarm).

Tabel 7. Parameter Pengujian Kehandalan Metode *PITO* terhadap Variasi *Jitter*

Parameter	Nilai
Maximum Segment Size	1440 bytes
Bandwidth	10Mbps
Latency	150ms
Maximum Jitter	0ms, 0.1ms, 1ms, 10ms
Packet Loss	0%
Durasi	60s

Tabel 8. Hasil Pengujian Kehandalan Metode *PITO* terhadap Variasi *Jitter*

Nilai *Jitter*	*Precision*
0ms	1
0.1ms	1
1ms	1
10ms	1

4.2.4 Packet Loss

Packet loss merupakan parameter jaringan yang berpengaruh besar terhadap kinerja metode PITO. Hal ini disebabkan karena terjadinya packet loss menimbulkan variasi yang besar dalam jeda waktu antar paket (packet inter-arrival time) yang diterima oleh receiving host. Oleh karena itu pengujian terkait parameter packet loss lebih kompleks daripada pengujian terkait parameter lainnya.

Pengujian kehandalan metode PITO terhadap perubahan nilai packet loss dilakukan dalam 2 tahap: 1) evaluasi keterkaitan antara jitter, packet loss, dan false alarm yang ditimbulkan, dan 2) evaluasi keterkaitan antara latency, jitter, packet loss, dan false alarm yang ditimbulkan.

Pengujian tahap 1 dilakukan dengan latency tetap sebesar 150ms, dan dimulai dengan tingkat packet loss 0% dan jitter 0ms. Apabila nilai precision dibawah 1, maka dilakukan pengujian dengan skenario berikutnya yaitu menaikkan nilai jitter. Hal ini terus dilakukan selama nilai precision masih menunjukkan perubahan atau nilai jitter mencapai 10ms. Tujuan pengujian tahap 1 ini untuk menyelidiki pengaruh jitter terhadap kehandalan metode PITO pada jaringan dengan tingkat packet loss tertentu, sekaligus untuk menyelidiki seberapa besar ketahanan metode PITO terhadap tingkat packet loss dalam jaringan. Output dari pengujian tahap 1 ini adalah nilai jitter minimal yang dibutuhkan untuk mengoptimalkan nilai precision pada nilai packet loss tertentu.

Pengujian tahap 2 dilakukan dengan tujuan untuk menyelidiki pengaruh latency terhadap kehandalan metode PITO pada jaringan dengan tingkat packet loss dan jitter tertentu. Parameter

yang digunakan pada pengujian tahap 2 ini adalah nilai-nilai jitter dan packet loss yang didapatkan dari output pengujian tahap 1.

Untuk setiap skenario pada setiap tahap dilakukan 10 kali percobaan. Parameter yang digunakan untuk pengujian terkait packet loss ini disajikan pada Tabel 9. Hasil pengujian tahap 1 disajikan pada Tabel 10, sedangkan hasil pengujian tahap 2 disajikan pada Tabel 11. Nilai precision pada tabel hasil pengujian dihitung berdasarkan jumlah percobaan yang berhasil (tidak terjadi false alarm) dan jumlah percobaan yang gagal (terjadi false alarm).

Tabel 9. Parameter Pengujian Kehandalan Metode *PITO* terhadap Variasi *Packet Loss*

Parameter	Nilai
Maximum Segment Size	1440 bytes
Bandwidth	10Mbps
Latency	150ms, 90, 30ms
Maximum Jitter	>=0ms
Packet Loss	0%, 0.001%, 0.01%, 0.1%, 1%
Durasi	60s

Tabel 10. Hasil Pengujian Tahap 1 Parameter *Packet Loss*

Nilai *Packet Loss*	Nilai *Latency*	Nilai *Jitter*	Precision
0%	150ms	0ms	1
0,001%	150ms	0ms	1
0,01%	150ms	0ms	1
0,1%	150ms	0ms	0.8
0,1%	150ms	0.1ms	1
1%	150ms	0ms	0.1
1%	150ms	0.1ms	0.5
1%	150ms	1ms	0.6
1%	150ms	10ms	0.7

Tabel 11. Hasil Pengujian Tahap 1 Parameter *Packet Loss*

Nilai *Packet Loss*	Nilai *Latency*	Nilai *Jitter*	Precision
0,1%	90ms	0ms	0.8
0,1%	90ms	0.1ms	0.8
0,1%	90ms	1ms	0.8
0,1%	90ms	5ms	0.9
0,1%	90ms	6ms	1
0,1%	30ms	1ms	0.8
0,1%	30ms	5ms	0.8
0,1%	30ms	6ms	1

4.2.4 Hasil Pengujian Kehandalan Metode *PITO*

Dari pengujian kehandalan metode PITO diatas, didapatkan kombinasi nilai parameter jaringan yang masih yang masih dapat ditangani dengan baik oleh metode PITO, yaitu kombinasi nilai parameter jaringan yang tidak membuat metode PITO menghasilkan false alarm. Kombinasi parameter tersebut disajikan pada Tabel 12.

Tabel 12. Nilai Parameter Jaringan yang Masih Dapat Ditangani Dengan Baik oleh Metode *PITO*

Packet Loss	*Jitter*	*Latency*	*Bandwidth*
0,001%	0ms	150ms	10Mbps
0,001%	0ms	150ms	1Mbps
0,001%	0ms	90ms	10Mbps
0,001%	0ms	90ms	1Mbps

Packet Loss	*Jitter*	*Latency*	*Bandwidth*
0,001%	0ms	30ms	10Mbps
0,001%	0ms	30ms	1Mbps
0,01%	0ms	150ms	10Mbps
0,01%	0ms	150ms	1Mbps
0,01%	0ms	90ms	10Mbps
0,01%	0ms	90ms	1Mbps
0,01%	0ms	30ms	10Mbps
0,01%	0ms	30ms	1Mbps
0,1%	6ms	150ms	10Mbps
0,1%	6ms	150ms	1Mbps
0,1%	6ms	90ms	10Mbps
0,1%	6ms	90ms	1Mbps
0,1%	6ms	30ms	10Mbps
0,1%	6ms	30ms	1Mbps

4.3 Pengujian Akurasi Metode *PITO* dalam Mendeteksi Terputusnya Koneksi

Pengujian akurasi metode PITO dalam mendeteksi terputusnya koneksi dilakukan dengan memutus segmen jaringan antara Node 1 dan Node 2 pada Gambar 4 setelah simulasi berjalan selama 50 detik. Ground truth yang digunakan dalam pengujian akurasi ini adalah titik waktu paket data terakhir yang berhasil melewati segmen antara Node 1 dan Node 2 diterima oleh receiving host. Percobaan dikatakan berhasil apabila metode PITO hanya menghasilkan timeout setelah titik ground truth tersebut (true positive). Sebaliknya percobaan dikatakan gagal apabila metode PITO menghasilkan timeout sebelum titik ground truth tersebut (false positive). Skenario pengujian pada pengujian akurasi ini didapat dari nilai parameter jaringan yang terdapat pada Tabel 12. Untuk setiap kombinasi parameter pada tabel tersebut dilakukan 10 kali pengujian. Hasil pengujian disajikan pada Tabel 13. Nilai precision pada tabel hasil tersebut dihitung berdasarkan jumlah true positive dan false positive pada setiap skenario pengujian.

Tabel 13. Hasil Pengujian Akurasi Metode *PITO* dalam Mendeteksi Terputusnya Koneksi

Parameter				Akurasi	*Precision*
Packet Loss	*Jitter*	*Latency*	*Bandwidth*		
0,001%	0ms	150ms	10Mbps	100%	1
0,001%	0ms	150ms	1Mbps	100%	1
0,001%	0ms	90ms	10Mbps	100%	1
0,001%	0ms	90ms	1Mbps	100%	1
0,001%	0ms	30ms	10Mbps	100%	1
0,001%	0ms	30ms	1Mbps	100%	1
0,01%	0ms	150ms	10Mbps	100%	1
0,01%	0ms	150ms	1Mbps	100%	1
0,01%	0ms	90ms	10Mbps	100%	1
0,01%	0ms	90ms	1Mbps	100%	1
0,01%	0ms	30ms	10Mbps	100%	1
0,01%	0ms	30ms	1Mbps	100%	1
0,1%	6ms	150ms	10Mbps	100%	1
0,1%	6ms	150ms	1Mbps	100%	1
0,1%	6ms	90ms	10Mbps	100%	1
0,1%	6ms	90ms	1Mbps	100%	1
0,1%	6ms	30ms	10Mbps	100%	1
0,1%	6ms	30ms	1Mbps	100%	1

4.3 Evaluasi Timeout yang Dihasilkan Metode *PITO*

Pada bagian ini akan dilakukan perbandingan antara timeout yang dihasilkan metode PITO di sisi receiving host pada setiap percobaan yang dilakukan pada pengujian akurasi diatas dengan retransmission timeout yang terjadi pada sending host. Selisih kedua nilai tersebut adalah penghematan waktu yang dihasilkan metode PITO dalam mendeteksi terputusnya koneksi. Penghematan waktu tersebut dinyatakan dalam satuan round-trip-time untuk melakukan normalisasi akibat keterkaitan erat antara retransmission timeout dengan round-trip-time. Hasil evaluasi tersebut dapat dilihat pada Tabel 14.

Tabel 14. Hasil Evaluasi Pengurangan *Delay Timeout*

Parameter				Pengurangan *Delay Timeout* (% dibanding *RTT*)
Packet Loss	***Jitter***	***Latency***	***Bandwidth***	
0,001%	0ms	150ms	10Mbps	0,214s (71,5% *RTT*)
0,001%	0ms	90ms	10Mbps	0,219s (121,76% *RTT*)
0,001%	0ms	30ms	10Mbps	0,194s (323,5% *RTT*)
0,001%	0ms	150ms	1Mbps	0,323s (107,7% *RTT*)
0,001%	0ms	90ms	1Mbps	0,201s (111,% *RTT*)
0,001%	0ms	30ms	1Mbps	0,136s (223,6% *RTT*)
0,01%	0ms	150ms	10Mbps	-0,137s (-45,6% *RTT*)
0,01%	0ms	90ms	10Mbps	-0,015s (-0,08% *RTT*)
0,01%	0ms	30ms	10Mbps	0,194s (323,5% *RTT*)
0,01%	0ms	150ms	1Mbps	0,32s (106,7% *RTT*)
0,01%	0ms	90ms	1Mbps	0,193s (107,5% *RTT*)
0,01%	0ms	30ms	1Mbps	0,139s (231,3% *RTT*)
0,1%	6ms	150ms	10Mbps	-0,51s (-169,9% *RTT*)
0,1%	6ms	90ms	10Mbps	-0,215s (-119,5% *RTT*)
0,1%	6ms	30ms	10Mbps	0,134s (223,8% *RTT*)
0,1%	6ms	150ms	1Mbps	0,975s (325,07% *RTT*)
0,1%	6ms	90ms	1Mbps	0,775s (430,3% *RTT*)
0,1%	6ms	30ms	1Mbps	1,425s (2374,9% *RTT*)

4.3 Pembahasan Hasil Pengujian Metode *PITO*

Berdasarkan hasil seleksi parameter yang telah dilakukan, metode PITO dapat diaplikasikan pada berbagai macam kondisi jaringan. Hal ini terbukti dari tingkat keberhasilan dalam percobaan dengan melakukan variasi nilai parameter latency, Maximum Segment Size, bandwidth, dan jitter yang mencapai 100%. Hal ini membuktikan kemampuan metode PITO dalam menghindari kesalahan deteksi terputusnya koneksi. Satu-satunya parameter yang tidak selalu mencapai tingkat keberhasilan 100% adalah packet loss, namun tingkat keberhasilan metode PITO terkait packet loss dapat ditingkatkan apabila pada jaringan terdapat jitter yang cukup. Semakin tinggi tingkat paket loss, semakin besar pula jitter yang dibutuhkan untuk mencapai tingkat keberhasilan yang tinggi dalam menghindari kesalahan deteksi terputusnya koneksi.

Berdasarkan hasil seleksi nilai parameter, tingkat packet loss yang masih dapat ditoleransi metode PITO adalah 0,1%, yang membutuhkan nilai jitter sebesar 6ms (untuk dua arah, sehingga nilai maksimum jitter yang dibutuhkan 12ms). Untuk tingkat packet loss 0,01% atau kurang bahkan tidak dibutuhkan adanya jitter. Pencapaian ini cukup bagus mengingat pada koneksi nirkabel kemungkinan terjadi jitter yang besar cukup tinggi. Kehandalan metode PITO pada kondisi jaringan dengan tingkat packet loss seperti dijelaskan diatas dibuktikan dengan akurasi

100% untuk seluruh kasus pada pengujian akurasi metode PITO dalam mendeteksi terputusnya koneksi.

Keuntungan penggunaan metode PITO terlihat dari hasil evaluasi pengurangan timeout delay. Dari total 18 skenario percobaan, 14 diantaranya menunjukkan pengurangan waktu deteksi terputusnya koneksi. Secara keseluruhan rata-rata pengurangan waktu deteksi terputusnya koneksi adalah 263,73% round trip time (RTT). Apabila proses pengiriman informasi vertical handover dari host receiver ke host sender membutuhkan waktu sebesar 50% RTT, maka rata-rata penghematan waktu bersih yang dihasilkan metode PITO adalah sebesar 213,73% RTT. Hasil ini menunjukkan bahwa metode PITO efektif dalam mengurangi delay pada proses vertical handover akibat proses deteksi terputusnya koneksi.

Terdapat pola yang jelas pada hasil evaluasi pengurangan timeout delay: untuk nilai packet loss, jitter, dan bandwidth yang sama, jika latency semakin kecil maka pengurangan delay timeout yang dihasilkan semakin besar. Pola ini terjadi karena latency yang kecil akan menyebabkan nilai Ex-Window timeout yang kecil. Ex-Window timeout merupakan batas bawah timeout yang dihasilkan oleh metode PITO, sehingga latency yang kecil secara keseluruhan akan mengurangi waktu deteksi terputusnya koneksi.

5 Kesimpulan dan Saran

5.1 Kesimpulan

Beberapa kesimpulan yang dapat diambil dari penelitian metode *Packet Inter-arrival Timeout* untuk mendeteksi terputusnya koneksi adalah sebagai berikut:

1. *Bandwidth overhead* akibat penggunaan metode *Packet Inter-arrival Timeout (PITO)* pada jaringan berbasis *ethernet* berkisar antara 0,533% hingga 1,35% apabila. Apabila *TCP option Next Packets Count* dikirimkan menggunakan interval beberapa paket, maka *bandwidth overhead* dapat diturunkan hingga menjadi hanya 0,02%.
2. Akurasi metode *PITO* relatif tidak terpengaruh oleh nilai parameter *latency*, *Maximum Segment Size*, *bandwidth*, dan *jitter*.
3. Akurasi metode *PITO* sangat dipengaruhi oleh tingkat *packet loss* dalam jaringan, namun akurasi metode *PITO* terkait *packet loss* dapat ditingkatkan apabila terdapat *jitter* yang cukup dalam jaringan. Semakin besar tingkat *packet loss* maka semakin besar pula nilai *jitter* yang dibutuhkan.
4. Tingkat packet *loss tertinggi* yang masih dapat ditoleransi oleh metode *PITO* adalah 0,1% dengan nilai *jitter* 6ms (satu arah, 12ms untuk dua arah).
5. Kecepatan deteksi terputusnya koneksi menggunakan metode *PITO* sangat ditentukan oleh *latency* jaringan. Semakin kecil *latency* maka waktu yang dibutuhkan untuk mendeteksi terputusnya koneksi juga akan semakin kecil.
6. Dari total 180 kali pengujian dengan tingkat *packet loss* antara 0,001% hingga 0,1%, penggunaan metode *PITO* menghasilkan pengurangan waktu deteksi terputusnya koneksi sebesar rata-rata 263,73% *round trip time* jaringan dibandingkan dengan menggunakan *retransmission timeout*.

5.2 Saran

Metode *PITO* menjanjikan peningkatan performa yang besar dalam mendeteksi terputusnya koneksi. Namun metode *PITO* menggunakan perhitungan yang cukup rumit dan terdapat perhitungan yang menggunakan angka *floating point*. Hal ini dapat berakibat menyita energi yang cukup besar dalam operasinya. Oleh karena itu perlu dilakukan evaluasi penggunaan energi oleh metode *PITO*.

Referensi

Paxson, V., Allman, M., Chu, J., Sargent, M. (2011), *Computing TCP's Retransmission Timer*, Internet Engineering Task Force.

Internet Engineering Task Force. (1989), *Requirements for Internet Hosts – Communication Layers*, Internet Engineering Task Force.

Paxson, V., Allman, M., Chu, J., Sargent, M. (2011), *Computing TCP's Retransmission Timer*, Internet Engineering Task Force.

RFC Editor. (2008), *STD 1 – Internet Official Protocol Standards*, University of Southern California/Informations Sciences Institute, Marina del Rey.

Sinha, R., Papadopoulos, C., Heidemann, J. (2007), *Internet Packet Size Distributions: Some Observations*, Technical Report ISI-TR-2007-643, University of Southern California/Informations Sciences Institute, Marina del Rey.