ANALISIS PERFORMANSI *AQM ROUTERS* YANG MENDUKUNG ALIRAN TCP DENGAN MENGGUNAKAN PENGONTROL *PROPORTIONAL-INTEGRAL-DERIVATIVE* (PID)

I Kadek Haddy W.[1], Hafidudin[2], Adiwijaya[3]
[1,2] Jurusan Teknik Elektro
[3]Program Perkuliahan Dasar dan Umum
Sekolah Tinggi Teknologi Telkom
Jl. Telekomunikasi Dayeuhkolot Bandung 40253
E-mail: [1]haddy_wiryana@yahoo.com, [2]hfd@stttelkom.ac.id, [3]adw@stttelkom.ac.id

ABSTRAKSI

Active Queue Management (AQM) adalah proses penandaan source TCP dari pusat router dengan mempertimbangkan penggunaan queue dan delay. Penggunaan AQM pada router akan memegang peranan penting dalam peningkatan kinerja aplikasi-aplikasi internet. Seperti aplikasi yang termasuk didalamnya voice over IP (VoIP), class of service (CoS) dan video streaming di mana besar paket dan durasinya menunjukkan variasi yang sangat signifikan. Hal ini sesungguhnya merupakan permasalahan kontroling. Didasarkan pada pembuatan model dinamik dari TCP's congestion-avoidance terdapat beberapa hal penting yang perlu mendapatkan perhatian, pertama parameter kunci network seperti TCP load, link capacity, dan round-trip-time yang menjadi penyebab utama masalah kontroling. Model standar AQM yang ada sekarang ini yaitu Random Early Detection (RED) sementara ini memang mampu mengatasi permasalahan TCP congestion.

Dalam penelitian ini dianalisis dan disimulasikan suatu metoda AQM alternatif dengan menggunakan pengontrol Proportional-Integral-Derivative (PID). Membandingkan pengontrol Proportional-Integral-Derivative (PID) dengan metoda Random Early Detection (RED) dan Proportional Integral (PI) dengan menggunakan simulasi Network Simulator (NS) dan menunjukkan hasil bahwa pengontrol Proportional-Integral-Derivative (PID) lebih baik dalam hal throughput, paket loss dan index fairness.

***Kata kunci**: AQM, RED, PID, PI, throughput, paket loss dan index fairness*

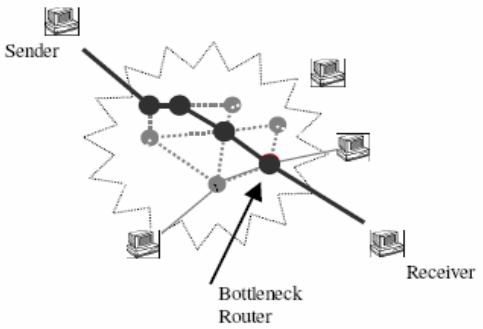
1. PENDAHULUAN

Pembuatan *Active Queue Management* (AQM) *routers* memegang peranan penting dalam peningkatan permintaan akan kinerja dari aplikasi internet seperti: *Voice over IP* (VoIP), *Class of Service* (CoS), dan *Video streaming*, dimana besar paket data dan durasi *session*-nya menunjukkan variasi yang sangat signifikan. Namun disisi lain terdapat kelemahan yang terjadi dalam komunikasi data ini, diantaranya masalah kongesti dalam pengiriman paket antar jaringan.

Pada gambar 1 terdapat sebuah contoh koneksi *sender-receiver* melewati sebuah *bottleneck router* berbasiskan TCP, pengirim mengecek ketersediaan *bandwidth* jaringan dengan meningkatan kecepatan sampai paket data hilang (*lost*) (gambar 2).

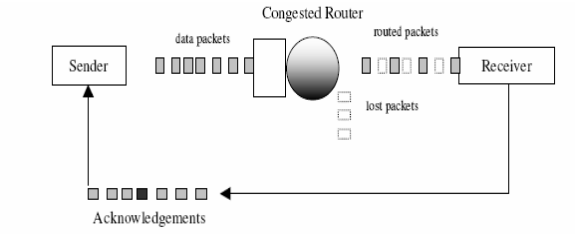
Atas paket-paket yang hilang tersebut, sinyal penerima dari pengirim akan dikurangi kecepatannya dan ini akan menyebabkan ketidakefisienan jaringan. Dengan adanya ketidakefisienan terhadap parameter kunci seperti *TCP load*, *link capacity*, dan *round-trip-time* dalam jaringan ini, maka skema RED (gambar 3) diperkenalkan untuk mengijinkan *router* melakukan manajemen TCP terhadap kinerja dalam jaringan.

Selanjutnya RED melakukan tindakan pendahuluan dengan mengukur panjang antrian pada *router* dan melakukan penstabilan kecepatan pengiriman agar sesuai. Namun RED (*Random Early Detection*) masih banyak memiliki kelemahan, diantaranya: panjang antrian bergantung pada level aliran TCP, menghasilkan *bandwidth* sistem lebih kecil yang akan mengurangi respon *transient* dari sistem tersebut, memiliki *margin phase* rendah yang akan menimbulkan terjadinya osilasi yang lebih banyak.



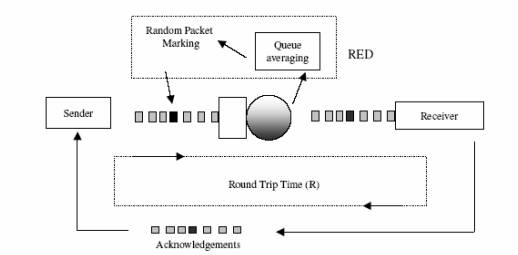
Gambar 1. Koneksi sender-receiver melewati bottleneck router

Oleh karena itu, pengontrol yang digunakan adalah pengontrol *Proportional-Integral-Derivative* (PID) untuk mengatasi kelemahan dari pengontrol RED.



Gambar 2. Skema koneksi sender-receiver tanpa AQM

$$G_p(j\omega_g)\left[K_p + j(K_d\omega_g - \frac{K_i}{\omega_g})\right] = -e^{jP_m}$$

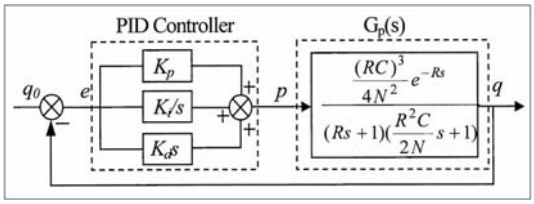


Gambar 3. Antisipasi congestion paket secara acak pada RED

2. DESAIN AQM SUATU ALIRAN TCP

2.1 Desain AQM menggunakan Pengontrol PID

Skema pengontrol PID digambarkan pada gambar 4.



Gambar 4. Diagram blok dari Pengontrol PID[13]

di mana:
C : *Link Capacity* (packets/s)
q_o : *Queue reference value*
N : *Load Factor* (number of active TCP sessions)
q : *Instantaneous queue*
R : *Round-Trip-Time* (RTT), di mana $R = 2(q/C+T_p)$,
T_p : delay propagasi (*fixed propagation delay*)
p : probabilitas *dropping/marking*
K_p,K_i dan K_d adalah konstanta *proportional*, *integral* dan *derivative* pada pengontrol PID

Pada kenyataannya, sistem antrian memerlukan format pengontrol terpisah waktu sampling (kT) digantikan menjadi *continuous time* (t), *integral* dan *differential* digantikan dengan *difference* dan *sum* yang berturut-turut, sehingga didapatkan suatu pemodelan:

$$p(k) = K_p e(k) + K_i T\sum_{j=0}^{k} e(j) + K_d\frac{e(k)-e(k-1)}{T}$$

$$= K_p\left\{e(k) + \frac{T}{T_i}\sum_{j=0}^{k} e(j) + \frac{T_d}{T}\left[e(k)-e(k-1)\right]\right\}$$

di mana

$$T_i = \frac{K_p}{K_i}, T_d = \frac{K_d}{K_p}$$

T = waktu *sampling*, dengan demikian dihasilkan:

$$\Delta p(k) = p(k) - p(k-1)$$

$$= K_p\left\{(1+\frac{T}{T_i}+\frac{T_d}{T})e(k) - (1+\frac{2T_d}{T})e(k-1) + \frac{T_d}{T}e(k-2)\right\}$$

Asumsikan bahwa spesifikasi dari PID sebagai pengontrol AQM memberikan *gain margin* (A_m) dan *phase margin* (P_m) yang stabil, yaitu $A_m > 1$ dan $0 < P_m < \pi/2$ sehingga didapatkan:

$$G_p(j\omega_p)\left[K_p + j(K_d\omega_p - \frac{K_i}{\omega_p})\right] = -\frac{1}{A_m}$$

di mana ω_p dan ω_g adalah frekuensi *phase* dan *gain crossover* dari *closed-loop* yang berulang. Pada saat penyetingan pengontrol, *bandwidth close-loop* sebaiknya dipilih secara hati-hati. Karena jika terlalu besar akan mengakibatkan saturasi, namun jika terlalu kecil akan menyebabkan respon lembam (*sluggish response*). *Bandwidth close-loop* biasanya dipilih mendekati *bandwidth open-loop* sebab pengontrol biasanya bekerja di bawah frekwensi *crossover*. Untuk itu dipilih

$\omega_p = \alpha\omega_c$, di mana $\alpha \in [0.5,2]$

Namun nilai α biasanya adalah 1, dan ω_c adalah frekwensi *phase crossover* dari *plant* $G_p(s)$ yang dirumuskan sebagai

$\angle G_p(j\omega_c) = -\pi$

Selanjutnya yaitu penentuan parameter K_p, K_i dan K_d. Dari persamaan sebelumnya didapat:

$$K_p = \mathrm{Re}\left[\frac{-1}{A_m G_p(j\omega_p)}\right]$$

$$K_i = (X_p\omega_g - X_g\omega_p)(\frac{\omega_p}{\omega_g} - \frac{\omega_g}{\omega_p})^{-1}$$

$$K_d = (\frac{X_p}{\omega_g} - \frac{X_g}{\omega_p})(\frac{\omega_p}{\omega_g} - \frac{\omega_g}{\omega_p})^{-1}$$

di mana

$$X_p = \mathrm{Im}\left[\frac{-1}{A_m G_p(j\omega_p)}\right], \quad X_g = \mathrm{Im}\left[\frac{-e^{jP_m}}{G_p(j\omega_g)}\right]$$

dan

$$\frac{\left|G_p(j\omega_p)\right|\cos(P_m - \angle G_p(j\omega_p))}{\left|G_p(j\omega_g)\right|\cos(\angle G_p(j\omega_p))} = \frac{1}{A_m}$$

2.2 Implementasi Digital Pengontrol PID

Dari persamaan sebelumnya didapatkan persamaan probabilitas PID sebagai berikut:

 ISSN: 1907-5022

$$p(k) = Kp\{(1+\frac{T}{T_i}+\frac{T_d}{T})e(k) - (1+\frac{2T_d}{T})e(k-1) + \frac{T_d}{T}e(k-2)\} + p(k-1)$$

dimana

$$T_i = \frac{K_p}{K_i}, T_d = \frac{K_d}{K_p}.$$

Dalam penulisan pada Network Simulator, persamaan probabilitas PID dapat ditulis sebagai berikut:

```
p=edp_.kp*(1+((1/edp_.w)/(edp_.kp/edp_.ki)))*(qle
  n-edp_.qref)-
  ((edp_.kd/edp_.kp)/(1/edp_.w))*((qlen-
  edv_.qold)-
  edp_.qref)+edp_.kp*(1+(2*(edp_.kd/edp_.kp)/(1/e
  dp_.w)))+edp_.kp*((edp_.kd/edp_.kp)/(1/edp_.w))
  *((qlen-(2*edv_.qold))-edp_.qref)+edv_.v_prob
p_old   = Δ p
q_old   =  q
```

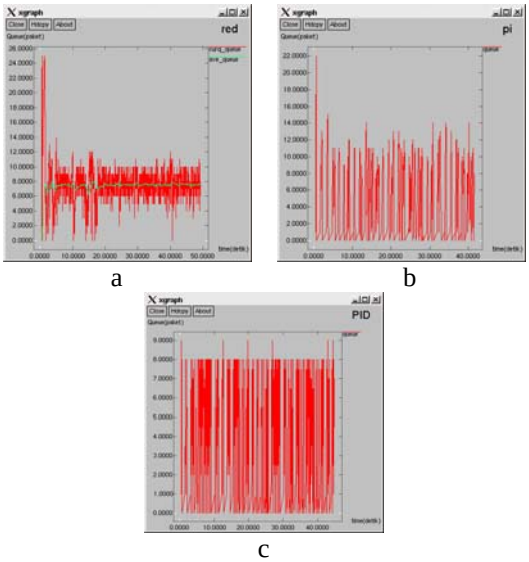
3. ANALISA PERFORMANSI SISTEM

3.1 Analisa Interaksi dua sumber dengan Pengontrol RED, PI dan PID

Pada bagian ini akan dianalisa bagaimana interaksi dua sumber dengan algoritma TCP Reno pada suatu *bottleneck link* yang menggunakan pengontrol RED, PI dan PID. Topologi yang digunakan seperti yang dijelaskan pada bab 3.1.

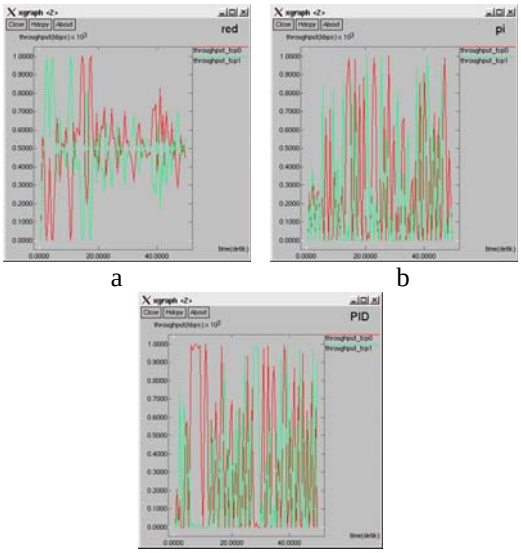
Tabel 1. Perbandingan nilai *throughput*, *index fairness* dan *packet loss*

Pengontrol	*Paket diterima*	*Paket loss (%)*	*Total Throughput (kbps)*	*Index fairness*
PI	6302	233 (3,70%)	549,832	0,9924
RED	11458	369 (3,22%)	999,810	0,9998
PID	7337	198 (2,70%)	640,159	0,97717



a b c

Gambar 5. Grafik antrian pada gateway dengan pengontrol RED (a), PI (b) dan PID (c)



a b c

Gambar 6. Grafik throughput pada gateway dengan pengontrol RED (a), PI (b) dan PID (c)

Pengontrol PID menghasilkan paket *loss* yang lebih kecil dibandingkan dengan RED dan PI. Hal ini dikarenakan pengontrol PID selalu menjaga agar antrian selalu berada dalam q_referensi, sehingga jaringan tidak terlalu terbebani oleh paket-paket.

Total *throughput* yang dihasilkan oleh pengontrol PID tidak sebesar RED. Ini disebabkan karena antrian pengontrol PID selalu berada dalam kisaran q_referensinya, yang menyebabkan paket yang berada di jaringan sedikit. Hal ini mempengaruhi jumlah throughput yang dihasilkan oleh pengontrol PID

3.2 Analisa Perubahan propagasi *delay* pada *bottleneck link*

Pada bagian ini akan dianalisa mengenai pengaruh perubahan propagasi *delay* pada sisi bottleneck link terhadap besarnya panjang antrian, *throughput*, *index fairness* dan *packet loss*, dengan menggunakan pengontrol PID, PI dan RED secara bergantian.

Tabel 2. Data perubahan propagasi delay pada gateway dengan Pengontrol RED, PI dan PID

Propagasi delay	*paket diterima*	*paket loss*	*Packet Loss (%)*	*Pengontrol*
0,05	*5695*	*241*	*4,23%*	*PI*
	6953	*185*	*2,66%*	*PID*
	11458	*369*	*3,22%*	*RED*
0,08	*8383*	*237*	*2,83%*	*PI*
	6969	*182*	*2,61%*	*PID*
	11458	*369*	*3,22%*	*RED*
0,10	*6302*	*233*	*3,70%*	*PI*
	7337	*198*	*2,70%*	*PID*
	11458	*369*	*3,22%*	*RED*

Tabel 3. Throughput dan Index fairness RED

Propa-gasi delay (ms)	Throughput (kbps)		Total Throug-hput (kbps)	Index fairness
	Reno1	Reno2		
0,05	505,054	494,756	999,81	0,999893922
0,08	505,054	494,756	999,81	0,999893922
0,10	505,054	494,756	999,81	0,999893922
rata-rata	505,054	494,756	999,81	0,999893922

Tabel 4. Throughput dan Index fairness PI

Propa-gasi delay (ms)	Throughput (kbps)		Total Through-put (kbps)	Index fairness
	Reno1	Reno2		
0,05	569,374	430,523	999,897	0,981081249
0,08	569,374	430,523	999,897	0,981081249
0,10	569,374	430,523	999,897	0,981081249
rata-rata	569,374	430,523	999,897	0,981081249

Tabel 5 . Throughput dan Index fairness PID

Propagasi delay (ms)	Throughput (kbps)		Total Through-put (kbps)	Index fairness
	Reno1	Reno2		
0,05	308,603	301,084	609,687	0,999847931
0,08	307,961	303,192	611,153	0,999939112
0,10	368,996	271,163	640,159	0,977177229
rata-rata	328,52	291,813	620,333	0,992321424

Rrespons time dari panjang antrian pada *gateway* yang menggunakan pengontrol PID lebih cepat daripada PI dan RED dengan berubahnya propagasi *delay*.

Pada pengontrol RED dan PID, *throughput* rata-rata dan total *throughput* yang dihasilkan tetap, walaupun *delay* propagasi berubah-ubah. Hal ini dikarenakan selisih perubahan *delay* propagasi yang terlalu kecil tidak berpengaruh pada pengontrol RED dan PID. Sedangkan pada pengontrol PI tampak terlihat pada tabel 4, bahwa perubahan nilai *delay* propagasi yang semakin besar akan menghasilkan *throughput* yang lebih besar dan *packet loss* yang semakin kecil, dikarenakan dengan semakin besarnya *delay* propagasi, maka paket yang masuk ke dalam *bottleneck link* akan semakin banyak sehingga panjang antrian pada *buffer* akan menurun.

3.3 Analisa Perubahan parameter *gateway*

Pada bagian ini akan dianalisa mengenai pengaruh perubahan nilai parameter q_{ref} (panjang antrian referensi) terhadap besarnya panjang antrian, *throughput*, *index fairness* dan *packet loss* pada *gateway* yang menggunakan pengontrol PID.

Tabel 6. Loss Data perubahan parameter gateway yang menggunakan pengontrol PID dan PI

Parameter qref	paket diterima	paket loss	Paket Loss (%)	Pengontrol
4 paket	4722	308	6,52%	PI
	4516	231	5,12%	PID
6 paket	5937	285	4,80%	PI
	5354	192	3,59%	PID
8 paket	6302	233	3,70%	PI
	7337	198	2,70%	PID
10 paket	7332	248	3,38%	PI
	8503	175	2,06%	PID

Table 7. Throughput dan index fairness PID

Parameter qref	Throughput (kbps)		Total Throughput (kbps)	Index fairness
	Reno1	Reno2		
4	195,759	198,203	393,962	0,999961516
6	243,161	226,392	469,553	0,998726228
8	368,996	271,163	640,159	0,977177229
10	363,061	382,723	745,784	0,999305412
rata-rata	292,744	269,620		0,993792596

Table 8. Throughput dan index fairness PI

Para-meter qref	Throughput (kbps)		Total Through-put (kbps)	Index fairness
	Reno1	Reno2		
4	198,552	213,388	411,940	0,998704603
6	265,378	257,884	523,262	0,999794931
8	298,916	250,916	549,832	0,992436459
10	365,679	274,043	639,722	0,979893860
rata-rata	282,1313	249,0578		0,992707463

Pada pengontrol PID dan PI berusaha mengatur panjang antrian berada disekitar qref walaupun parameter tersebut diubah-ubah. Dan tampak pada tabel 8, pengontrol PID yang memiliki nilai qref lebih besar akan menghasilkan *throughput* yang lebih besar dan *packet loss* yang semakin kecil. Hal ini dikarenakan pengontrol PID berusaha mengatur panjang antrian di *bottleneck link* berada di sekitar qref sehingga dapat menampung jumlah paket yang lebih banyak dan menghasilkan jumlah paket yang diterima menjadi lebih banyak jika parameter qref diperbesar. Oleh karena itu, *throughput* yang diperoleh lebih besar daripada menggunakan parameter qref yang lebih kecil.

3.4 Analisa Interaksi N sumber dengan Pengontrol PID, RED dan PI

Pada bagian ini akan dianalisa pengaruh banyaknya jumlah user terhadap panjang antrian, *throughput*, *index fairness* dan *packet loss* pada *gateway* yang menggunakan pengontrol PID, RED dan PI.

Tampak pada gambar panjang antrian yang .4 bahwa pengontrol PID berusaha melakukan manajemen *trade off* antara paket yang di *drop* dengan nilai *throughput* yang besar.

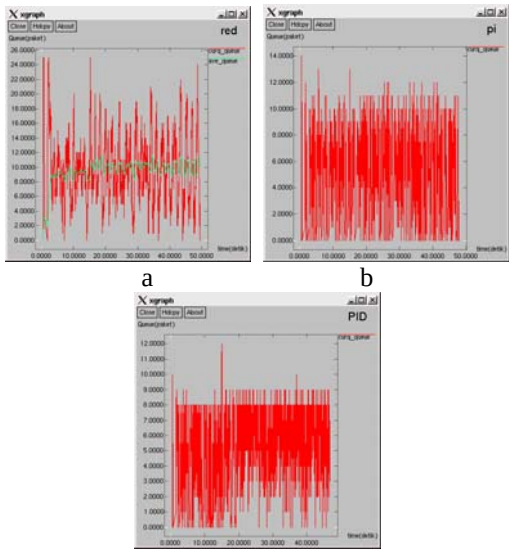
Tabel 9. Data perubahan interaksi N sumber dengan pengontrol RED, PI dan PID

N User	*Paket diterima*	*paket loss*	*Paket Loss (%)*	*Pengontrol*
5	*8817*	*583*	*6,61%*	*PI*
	10162	*385*	*3,79%*	*PID*
	11462	*622*	*5,43%*	*RED*
10	*11442*	*1360*	*11,89%*	*PI*
	11039	*695*	*6,30%*	*PID*
	11251	*938*	*8,34%*	*RED*
20	*11442*	*1360*	*11,89%*	*PI*
	11135	*1114*	*10,00%*	*PID*
	11219	*1498*	*13,35%*	*RED*

Pada tabel 9 terlihat bahwa jumlah user yang lebih besar pada jaringan *bottleneck link* dengan *router*/*gateway* menggunakan pengontrol PID mempunyai nilai *packet loss* yang lebih kecil dibandingkan dengan pengontrol RED dan PI, namun menghasilkan throughput yang lebih kecil. Hal ini dikarenakan pengontrol PID melakukan manajemen terhadap paket yang datang dengan jumlah antriannya dan berusaha mengatur panjang antrian agar selalu berada di sekitar qref. Pengontrol PID juga melakukan *trade off* antara paket yang di *drop* dan jumlah paket yang *loss* dengan besarnya *throughput* yang dihasilkan. Walaupun jumlah user menjadi lebih banyak namun pengontrol PID dapat meminimalisasi besarnya probabilitas *dropping* dan memiliki nilai *throughput* yang dihasilkan.

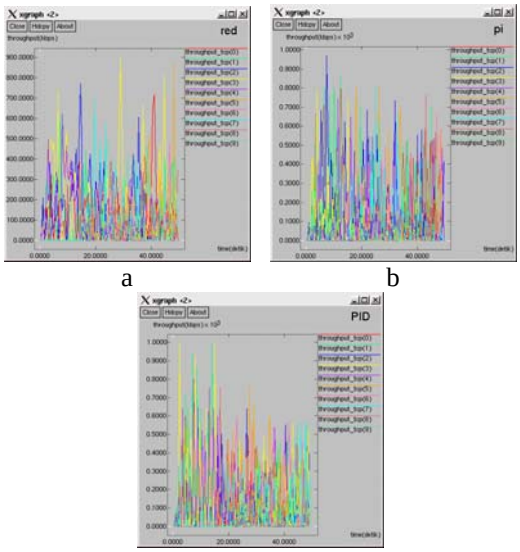
3.5 Analisis Pengaruh Perbedaan Waktu Pengiriman Paket

Pada bagian ini dianalisa tentang perubahan waktu start pengiriman paket untuk setiap pengontrol AQM dengan jumlah node sebanyak 10 node dengan spesifikasi waktu start yang berbeda. Yaitu node 0 sampai 4 waktu start pada detik ke-0.5s, sedangkan node 5 sampai 9 menggunakan waktu start pada detik ke-15s. Hal ini mensimulasikan tentang pengujian terhadap pengontrol AQM, bagaimana suatu pengontrol dapat melakukan manajemenisasi terhadap jumlah user yang bertambah setiap saat. Yaitu pada detik ke 0.5 terdapat 5 user dan pada detik ke 15 terjadi penambahan user sebanyak 5. jadi total user pada detik ke 15 sebanyak 10 user.



a b c

Gambar 7. Grafik antrian untuk perubahan waktu start paket dengan pengontrol RED (a), PI (b) dan PID (c)



a b c

Gambar 8. Grafik throughput untuk perubahan waktu start paket dengan pengontrol RED (a), PI (b) dan PID (c)

Table 10. Data pengaruh perubahan perbedaan waktu start pengiriman paket

Pengontrol	*Paket diterima*	*Paket loss*	*Paket Loss (%)*	*Total troughput*
RED	*7023*	*518*	*7,38%*	*989,875*
PI	*9928*	*856*	*8,62%*	*911,170*
PID	*10601*	*600*	*5,66%*	*975,636*

Dari table 10 dapat dilihat bahwa jumlah paket *loss* dari pegontrol PID lebih kecil dibandingkan dengan pengontrol PI dan RED. Jumlah total *throughput* yang dihasilkan pengontrol PID sebanding dengan pengontrol PI dan RED. Ini juga menandakan bahwa pengontrol PID melakukan

manajemen *trade off* terhadap paket yang di *drop* dengan total *throughput* yang dihasilkan. Pada gambar 7 dan 8, perubahan waktu pengiriman paket juga menyebabkan semua pengontrol mendapat respon yang besar saat detik ke 15s. hal ini dikarenakan pada detik ke 15s terjadi peningkatan jumlah paket yang mengalir dalam jaringan.

4. KESIMPULAN

Kesimpulan yang dapat diambil dalam Penelitian ini adalah:

a. Pengontrol PID merupakan pengontrol AQM yang memiliki paket *loss* yang kecil, *throughput* yang besar sebanding dengan pengontrol PI dan RED. Pengontrol PID memiliki nilai antrian yang selalu berada dalam q referensinya. Hal ini dikarenakan pengontrol PID melakukan manajemenisasi *trade off* yang terbaik terhadap paket yang di *drop* dengan jumlah *throughput* yang dihasilkan.
b. *Index fairness* yang dihasilkan dengan pengontrol PID tidak berbeda jauh dengan pengontrol PI dan RED. Ketiga pengontrol memiliki nilai *index fairness* mendekati 1. Hal ini menandakan bahwa ketiga pengontrol melakukan pembagian *bandwidth* yang cukup adil terhadap semua node.
c. Untuk pengontrol PI dan PID, semakin besar nilai q referensi maka semakin baik paket *loss*, total *throughput* dan *index fairness* yang dihasilkan. Namun nilai yang dihasilkan pengontrol PID relatif lebih baik dibandingkan dengan pengontrol PI.
d. Penambahan jumlah user yang terhubung dengan *router/gateway* akan memberikan nilai paket *loss* yang besar juga. Namun nilai paket *loss* untuk pengontrol PID relatif lebih kecil dibandingkan dengan pengontrol PI dan RED.

DAFTAR PUSTAKA

[1] Adiwijaya, R. Saragih, dan B. Riyanto T. Sistem Kontrol Umpan Balik untuk Aliran TCP pada Router Suatu Jaringan Komputer. *Jurnal Telekomunikasi*, Vol. 8 no. 2. Desember 2003.
[2] Chung, J. dan Claypool, M., *Analysis of Active Queue Management*, Computer Science Department, Worcester Polytechnic Institute, Worcester, MA 01609, USA
[3] Floyd, S., Jacobson, V., Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Transactions on Networking*, Vol. 1, August 1997.
[4] Hollot, C. V., Misra, V., Towsley, D. dan Gong, W. B., Analysis and Design of Controllers for AQM Routers Supporting TCP Flows. Systems and Control Methods for Communication Networks,. *IEEE TAC's*, March 2002.
[5] Hollot, C. V., Misra, V., Towsley, D. dan Gong, W. B., A Control Theoretical Analysis of RED, *IEEE INFOCOM*, 2001.
[6] Kumar, A., Traffic Sensitive Quality of Service Controller, *A Thesis submitted to the Faculty of the WORCESTER POLYTECHNIC INSTITUTE In partial fulfillment of the requirements for the Degree of Master of Science in Computer Science*, 2003.
[7] Misra, V., Gong, W. B., dan Towsley, D., *Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED*, ACM/SIGCOMM, 2000.
[8] *Network Simulator Home Page*, http://www.isi.edu/nsnam/ns/
[9] Nurhalim, *Penghindaran kongesti pada aliran TCP dengan menggunakan pengontrol Proporsional Integaral*, STTTELKOM, Bandung, 2005.
[10] Ogata, K., *Teknik Kontrol Automatik.* Erlangga, Indonesia, 1996,
[11] Seungwan, R., *Active Queue Management (AQM) based Internet Congestion Control*, University of Buffalo, October 2001.
[12] Wirawan, A. B. dan Indarto, E., *Mudah Membangun Simulasi dengan Network Simulator-2*, Andi Yogyakarta, 2004.
[13] Yanfei, F., Fengyuan, R. dan Chuang, L., *Design a PID Controller for Active Queue Management*, Computer Science and Technology Department, Tsinghua University, Beijing, China.