OPEN ACCESS
Indo-JC
Indonesian Journal of Computing
ISSN 2460-9056
socj.telkomuniversity.ac.id/indojc

Ind. Journal on Computing
Vol. 2, Issue. 1, Maret 2017. pp. 119-130
doi:10.21108/indojc.2017.21.139

Active Queue Management (AQM) Performance Analysis Based On Controled Delay (CoDel) Algorithm On User Datagram Protocol (UDP)

Satria Mandala [*1], Muhammad Noer Iskandar [2], Muhammad Arief Nugroho [3]

Fakultas Informatika - Universitas Telkom
Jl. Telekomunikasi No. 1 Bandung, Indonesia

[1] satriamandala@.telkomuniversity.ac.id
[2] noeris@student.telkomuniversity.ac.id
[3] arif.nugroho@telkomuniversity.ac.id

Abstract

Bufferbloat merupakan salah satu kondisi buffer dengan ukuran besar yang cenderung selalu penuh dan menyebabkan antrian panjang didalam buffer, jika hal ini terjadi secara terus-menerus maka dapat menyebabkan jeda transmisi yang tinggi. *Bufferbloat* sering terjadi pada aplikasi berbasis real-time. *Active Queue Management* (AQM) merupakan salah satu cara untuk menangani terjadinya *bufferbloat.*, AQM umumnya menggunakan algoritma Drop Tail untuk menangani kondisi antrian panjang dalam buffer router di jaringan. Namun demikian, performansi AQM berbasis Drop Tail kurang dapat diandalkan karena jeda transmisi dalam keadaan *bufferbloat* masih tinggi. Telah banyak studi dilakukan untuk menangani *bufferbloat*, seperti Drop Tail, Random Early Detection (RED) dan Controlled Delay (CoDel). Dari riset yang telah dilakukan tersebut masih sulit ditemukan performasi algoritma terbaik dalam menangani *bufferbloat.* Untuk hal tersebut, paper ini menyajikan studi performansi penanganan bufferbloat menggunakan ketiga algoritma diatas. Dalam studi ini, video streaming digunakan sebagai *traffic* uji untuk menentukan performansi algoritma terbaik dalam mengatasi *bufferbloat*. Sedangkan metriks uji yang digunakan dalam riset ini adalah *latency*, *throughput* dan *packet-loss*. Analisa hasil pengujian mengambil 3 hasil terbaik dalam setiap percobaan. Hasil pengujian menunjukan performansi algoritma CoDel jauh lebih baik dalam menangani *latency* yang tinggi pada kondisi bufferbloat dibandingkan RED dan Drop Tail. Namun untuk *packet-loss* dan *throughput* performansi RED dan Drop Tail masih unggul dibanding algoritma CoDel.
Keywords: AQM, *Bufferbloat*, CoDel, *latency*, *throughput*, *packet-loss*

I. Introduction

Penggunaan internet sekarang ini bisa dilakukan dengan bermacam cara seperti *browsing/searching*, *download/upload*, bahkan bisa digunakan secara real-time seperti *streaming. Streaming* merupakan teknologi didalam internet yang dapat menjalankan file audio maupun video secara real-time yang berada di *streaming-server* tanpa harus men-*download* file tersebut [15]. Pada proses *streaming* sering kali terjadi *buffer* dengan

Received on Feb 2017. Accepted on Mar 2017

beberapa penyebab seperti *bandwith* kecil, file *streaming* terlalu besar dan lain sebagainya. Pada proses streaming ini juga menggunakan RTP (Real-Time Transport Protocol) untuk pengiriman packet dari streaming-server ke client. RTP (Real-Time Transport Protocol) merupakan suatu protokol standar untuk mengirimkan data multimedia seperti audio dan video secara real-time [13]. Penyedia layanan internet juga sedang mengalami tingginya permintaan *bandwith*, karena idealnya dengan bandwith yang besar harus disertai dengan *latency* yang baik.

Buffer merupakan area memori yang dapat menyimpan antrian paket-paket data ketika paket tersebut sedang dipindahkan dari *streaming-server* terhadap *client*. Buffer diperlukan untuk menampung aliran paket yang deras (*burst packet*). Jika suatu saat bersamaan terjadi aliran paket yang deras dan disaat kondisi buffer penuh, akan sangat mengurangi fungsionalitas dari buffer. Bufferbloat merupakan salah satu kondisi buffer dengan ukuran besar yang cenderung selalu penuh dan menyebabkan antrian panjang didalam *buffer* [3], jika hal ini terjadi secara terus-menerus maka dapat menyebabkan jeda transmisi menjadi tinggi. Bufferbloat sering terjadi pada aplikasi berbasis real-time. Meningkatnya permintaan bandwidth dan buffer yang cenderung penuh menyebabkan tingginya *end-to-end latency* dan pengurangan throughput dalam jaringan. *Active Queue Management* (AQM) berbasis *Controlled Delay* (CoDel) merupakan salah satu cara menangani terjadinya *bufferbloat* [7]. Umumnya pada router menggunakan algoritma Drop Tail untuk menangani kondisi antrian atau buffer dalam jaringan termasuk kondisi bufferbloat. Pada implementasinya performansi algoritma Drop Tail masih mengalami jeda transmisi yang tinggi dalam menangani kondisi bufferbloat untuk studi kasus video streaming. Telah banyak dilakukan studi yang dilakukan untuk menangani bufferbloat seperti Drop Tail, Random Early Detection (RED) dan Controlled Delay (CoDel). Dari riset yang telah dilakukan masih sulit ditentukan performasi algoritma terbaik dalam menangani bufferbloat.

Ali Ahammed dan Resma Banu (2010) melakukan riset tentang performansi AQM. Dalam riset tersebut membandingkan performansi RED, FRED, dan SFB dengan protocol yang digunakan UDP. Hasil riset tersebut menjelaskan bahwa performansi RED lebih unggul menggunakan protokol UDP daripada algoritma lain yang diujikan akan tetapi sulit dalam konfigurasi. Ganesh Patil dan Sally McClean (2011) melakukan riset perbandingan performansi Drop Tail dan RED. Pada hasil risetnya menjelaskan bahwa performansi Drop Tail lebih baik dalam menangani buffer yang kecil dibandingkan RED.

Praveen Kumar dan Sanjay Kumar (2013), melakukan riset perbandingan performansi AQM berbasis Drop Tail dan RED. Pada hasil risetnya menjelaskan bahwa performansi RED lebih unggul dalam menangani *throughput* dan *packet-loss* dibandingkan dengan Drop Tail. Tanvi Sharma (2014), menjelaskan bahwa performansi algoritma CoDel unggul dibeberapa parameter pengujian karena CoDel menggunakan *timestamp* pada setiap paket untuk menjaga antrian tidak penuh dan menjaga latency tetap rendah. Pada paper tersebut juga dijelaskan algoritma RED dan Drop Tail unggul dalam beberapa parameter pengujian. Kemudian usaha ini dilanjutkan oleh Jae Chung dan Mark Claypool (2015), mereka menjelaskan bahwa algoritma RED lebih unggul dalam menangani kondisi bufferbloat.

II. LITERATURE REVIEW

2.1 Bufferbloat

Buffer merupakan suatu area memori yang digunakan untuk menyimpan data ketika data tersebut sedang dipindahkan dari suatu device ke device lain. Bufferbloat merupakan fenomena penumpukan yang terjadi saat buffer pada suatu jaringan yang sedang dalam kondisi cenderung penuh. Penumpukan yang berlebihan menyebabkan paket-paket yang dikirimkan menghabiskan banyak waktu dalam buffer, padahal paket tersebut memiliki batas waktu. Akibatnya banyak paket-paket yang sudah melebihi batas waktu menyebabkan keterlambatan paket dan menurunkan performa jaringan. Bufferbloat dapat didefinisikan dengan meningkatnya *latency* dan menurunkan throughput.

2.2 Active Queue Management

Untuk menangani kondisi seperti bufferbloat, anjuran untuk menggunakan Active Queue Management (AQM) sudah lama direkomendasikan [1]. Bila berbicara mengenai AQM maka hal tersebut tidak lepas dari kongesti. Kongesti adalah kondisi ketika agregat kebutuhan bandwidth melebihi kapasitas link yang tersedia. Akibatnya adalah banyaknya packet loss, utilisasi link yang rendah (throughput rendah), delay queue yang tinggi, dan congestion collapse.

AQM merupakan satu cara congestion control pada closed-loop atau packet switched network yang eksplisit. AQM berperan penting dalam mengatur buffer dan mengurangi *latency*. Secara umum berfungsi dalam mengontrol queue agar tidak full atau tumbuh terlalu besar dengan memonitor queue paket dengan menandai (marking) atau mendropnya. Pada kenyataanya, AQM tidak banyak digunakan pada router dan benar-benar tidak diterapkan pada banyak perangkat [2].

2.3 CoDel : *Active Queue Management for Low Latency*

CoDel adalah AQM baru yang dikembangkan untuk mengatasi masalah bufferbloat (seperti pada router) dan menggantikan algoritma RED karena dinilai sulit dikonfigurasi karena perlu dilakukan tuning dengan tingkat ketelitian tinggi [7]. CoDel bersifat adaptif "no-knobs" karena dikembangkan dengan tujuan agar dapat dikonfigurasikan dengan mudah (parameterless) atau tidak banyak parameter yang perlu diatur, tidak berdasar pada queue size, dapat menangani good queue dan bad queue secara berbeda, mengontrol delay, menjaga delay/latency rendah saat terjadi burst traffic, dapat beradaptasi dengan link dinamis tanpa berdampak buruk pada throughput, dan cukup sederhana untuk diimplementasikan pada router dengan skala kecil ataupun besar.

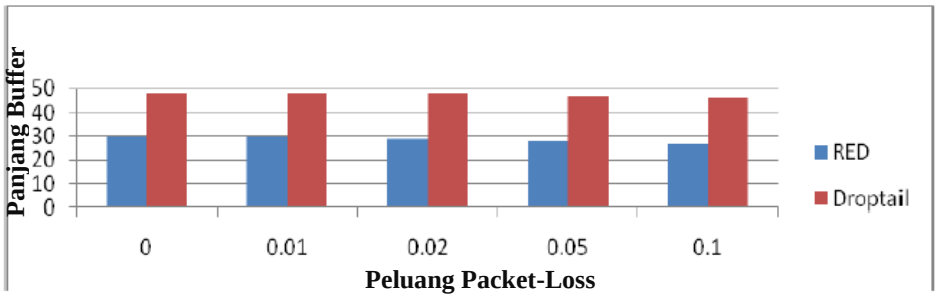
```
On arrival of every packet:
if current_queue_size < queue_limit then
    Enqueue the packet
    Attach a timestamp in packet header
end
else
    Drop the Packet
end
On departure of every packet:
dequeue_time = timestamp for dequeue time
soujourn_time = dequeue_time – enqueue_time
if inside the dropping state then
    if soujourn_time < target or
    current_queue_size < MTU then
        Do not drop packets
        Come out of dropping state
    end
    else
        while dequeue_time ≥ next_drop_time do
            Drop the packet
            count = count + 1
            next_drop_time + = interval /√count
        end
    end
end
else if outside dropping state and first packet is
being dropped then
    Enter the dropping state
end
```

Gambar 1 CoDel Algorithm [11]

2.4 Riset Tentang Penanganan Bufferbloat

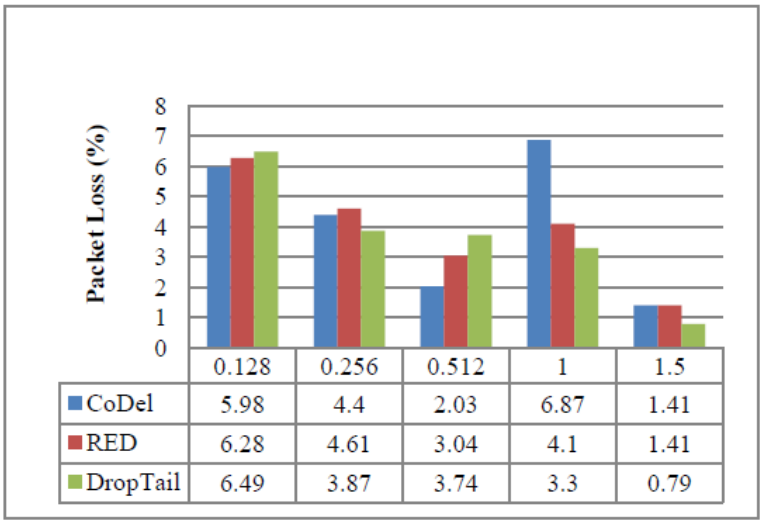
Analyzing the Performance of Active Queue Management Algorithm oleh Ali Ahammed dan Resma Banu (2010) menjelaskan bahwa performansi RED lebih unggul daripada algoritma lain yang diujikan akan tetapi sulit dalam konfigurasi. Drop Tail and RED Queue Management With Small Buffers oleh Ganesh Patil dan Sally McClean (2011), menjelaskan bahwa performasi algoritma Drop Tail lebih baik dalam menangani buffer kecil dibandingkan algoritma RED.

Performance Evaluation of Queue Management Techniques : Drop Tail and RED over Wireless Networks oleh Praveen Kumar dan Sanjay Kumar (2013), menjelaskan bahwa performansi RED lebih unggul dibandingkan dengan dengan algoritma Drop Tail dalam menangani *Packet-Loss.*



Gambar 2 Perbandingan performansi RED dan Drop Tail

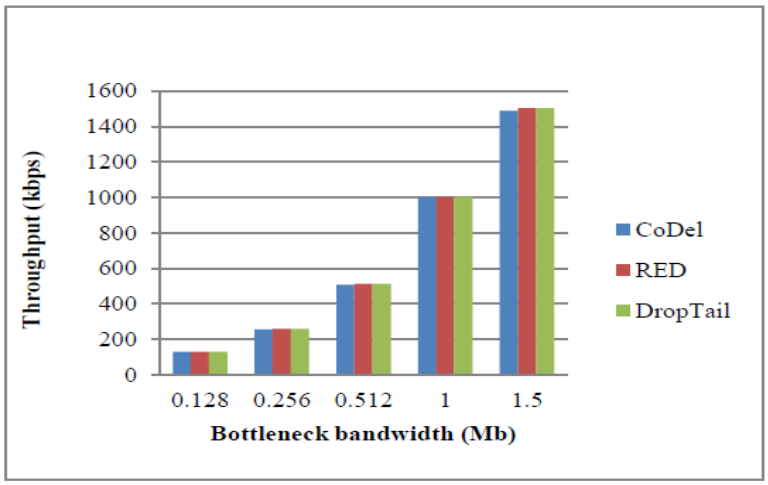
Pada gambar 2 menjelaskan perbandingan parameter packet loss pada performasi algoritma Drop Tail lebih besar 20% dibandingkan performansi algoritma RED. Controlling Queue Delay (CoDel) to counter the Bufferbloat Problem in Internet oleh Tanvi Sharma (2014), menjelaskan bahwa performansi algoritma CoDel unggul dibeberapa parameter pengujian. Pada paper tersebut juga dijelaskan algoritma RED dan Drop Tail unggul dalam beberapa parameter pengujian.



	0.128	0.256	0.512	1	1.5
CoDel	5.98	4.4	2.03	6.87	1.41
RED	6.28	4.61	3.04	4.1	1.41
DropTail	6.49	3.87	3.74	3.3	0.79

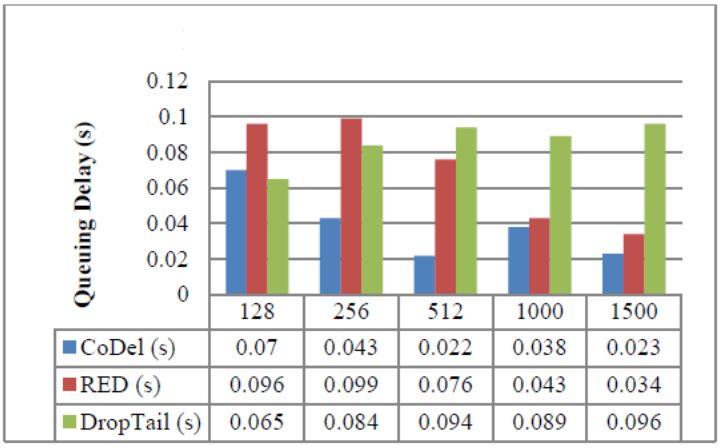
Gambar 3 Perbandingan Packet-Loss performansi CoDel, RED, dan Drop Tail

Pada ganbar 3 menjelaskan bahwa nilai packet-loss pada beberapa kondisi bottleneck. Setiap performansi ketiga algoritma menunjukan nilai yang unggul dalam beberapa kondisi bottleneck. Pada kondisi bottleneck kurang dari 1 mbps, performansi CoDel menunjukan keunggulan dalam menangani packet-loss. Sedangkan pada kondisi bottleneck 1 mbps atau lebih, performansi Drop Tail menunjukan keunggulan dalam menangani packet-loss.



Gambar 4 Perbandingan Throughput performansi CoDel, RED, dan Drop Tail

Pada gambar 4 menjelaskan bahwa nilai throughput pada beberapa kondisi bottleneck. Gambar ini menjelaskan bahwa performansi algoritma Drop Tail lebih unggul dibandingkan dengan RED dan CoDel. Selisih nilai throughput pada ketiga performansi menunjukan selisih nilai yang kecil. Semakin besar nilai bottleneck, maka semakin besar throughput yang ditunjukan



	128	256	512	1000	1500
CoDel (s)	0.07	0.043	0.022	0.038	0.023
RED (s)	0.096	0.099	0.076	0.043	0.034
DropTail (s)	0.065	0.084	0.094	0.089	0.096

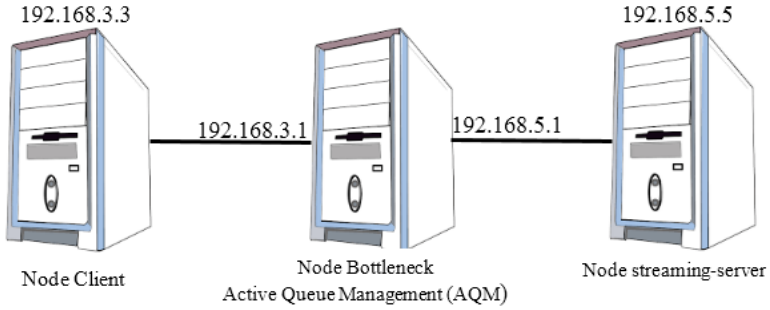
Gambar 5 Perbandingan *Latency* performansi CoDel, RED, dan Drop Tail

Pada ganbar 5 menjelaskan bahwa nilai latency pada beberapa kondisi bottleneck. Setiap performansi ketiga algoritma menunjukan nilai yang unggul dalam beberapa kondisi bottleneck. Pada kondisi bottleneck kurang dari 256 kbps, performansi Drop Tail menunjukan keunggulan dalam menjaga rendahnya *latency*. Sedangkan pada kondisi bottleneck 256 kbps atau lebih, performansi CoDel menunjukan keunggulan dalam menjaga *latency* tetap rendah.

Kemudian riset dilanjutkan pada paper berjudul "Analysis of Active Queue Management" Oleh Jae Chung dan Mark Claypool (2015), menjelaskan bahwa algoritma RED lebih unggul dalam menangani kondisi bufferbloat.

III. Research Method

3.1 Topologi Jaringan



Gambar 6 Topologi Jaringan untuk performansi CoDel, RED, dan Drop Tail

Gambar 6 adalah rencana topologi uji performansi 3 algoritma, yaitu Drop Tail, RED, dan CoDel. Seperti digambar 3 node diinstall dalam 2 network yang berbeda, yaitu 192.168.3.0 dan 192.168.5.0. Pengalamatan node Client dinetwork yang berbeda dengan node router bottleneck dan node streaming server dimaksudkan agar simulasi menggambarkan situasi jaringan internet yang sebenarnya, dimana antara client dan server berada di jaringan yang tidak sama

Percobaan akan dimulai dengan mensetup Node Client agar mengirimkan paket ke node streaming-server melewati node bottleneck. Kemudian streaming-server menjawab permintaan packet dari node client. Node streaming-server akan mengirimkan paket streaming menggunakan Real-Time Transport Protocol (RTP) secara terus-menerus ke node bottleneck. Node bottleneck menerima aliran paket yang sangat deras dari streaming-server. Performansi Active Queue Management (AQM) berbasis Controlling Delay (CoDel), RED (Random Early Detection) dan Drop Tail diujikan pada node router bottleneck.

3.2 Skenario Perancangan Uji

a. Kondisi normal tanpa Bottleneck

Pada skenario ini dilakukan pada router yang menghubungkan antara client dan server. Pengujian dilakukan tanpa adanya kondisi bottleneck dan mengikuti setiap link interface pada PC-Router berbasis linux yaitu 100 Mbps full duplex. Kemudian dilakukan proses streaming antara client dan server, serta melakukan perhitungan untuk parameter yang diujikan seperti *latency*, *throughput* dan *packet-loss*

b. Kondisi Bottleneck

Kondisi ini dibuat pada sisi router yang umumnya memiliki 100 Mbps full duplex, diperkecil dengan beberapa skenario pengujian. Dan pada node client dan server tetap menggunakan interface default, sehingga akan terjadi bottleneck ketika packet melalui router.

- **Bottleneck 200 kbps dan 200 kbps [15]**
  Pada skenario ini dilakukan pada router yang menghubungkan antara client dan server. Pertama diatur fungsi bottleneck pada kedua interface. Interface yang menghubungkan client diatur dengan kecepatan 200 kbps dan interface yang menghubungkan server diatur dengan kecepatan 200 kbps. Kemudian pada kedua interface ditambahkan algoritma Drop Tail, RED dan Controlling Delay dengan limit, target dan interval berbeda.
- **Bottleneck 200 kbps dan 200 kbps [15]**
  Pada skenario ini dilakukan pada router yang menghubungkan antara client dan server. Pertama diatur fungsi bottleneck pada kedua interface. Interface yang menghubungkan client diatur dengan kecepatan 200 kbps dan interface yang menghubungkan server diatur dengan kecepatan 200 kbps. Kemudian pada kedua interface ditambahkan

algoritma Drop Tail, RED dan Controlling Delay dengan limit, target dan interval berbeda.

- **Bottleneck 150 kbps dan 150 kbps [15]**
  Pada skenario ini dilakukan pada router yang menghubungkan antara client dan server. Pertama diatur fungsi bottleneck pada kedua interface. Interface yang menghubungkan client diatur dengan kecepatan 150 kbps dan interface yang menghubungkan server diatur dengan kecepatan 150 kbps. Kemudian pada kedua interface ditambahkan algoritma Drop Tail, RED dan Controlling Delay dengan limit, target dan interval berbeda.

3.3 Parameter Uji

Terdapat tiga parameter uji yang digunakan dalam menganalisa mekanisme Controlling Delay dan Drop Tail.

a. Latency

Latency adalah satuan waktu yang dibutuhkan paket saat berada didalam jaringan pada selang waktu paket masuk antrian pada node pengirim sampai paket diterima oleh client, seperti yang ditunjukan pada persamaan (1).

Latency = Packet Receive time – Packet Send time ….. (persamaan 1)

b. Throughput

Througput adalah jumlah total byte yang diterima oleh receiver pada selang waktu saat paket pertama dan terakhir dikirimkan, seperti persamaan berikut:

$$throughput = \frac{jumlah\_data\_yang\_dikirim}{waktu\_pengiriman\_data} \quad \text{….(persamaan 2)}$$

c. Packet Loss

Packet Loss mengindikasikan rata-rata packet loss yang dirasakan pada sisi client. Yang diukur dalam tugas akhir ini adalah perbandingan jumlah paket yang dibuang dalam jaringan dengan jumlah paket yang dikirmkan selama selang waktu pengujian dilakukan. Dapat dilakukan pada persamaan berikut ini.

$$packet-loss = \frac{packet\_transmitted - packet\_received}{packet\_transmitted} x100\% \quad \text{…(persamaan 3)}$$

IV. RESULTS AND DISCUSSION

Pada sub bab ini menjelaskan hasil yang didapat dari percobaan sesuai skenario uji. Pada sub bab 4.1 menjelaskan hasil percobaan kondisi tanpa bottleneck dan pada sub bab 4.2 menjelaskan hasil percobaan kondisi dengan bottleneck.

4.1 Kondisi Tanpa Bottleneck

Percobaan	Latency (ms)	Packet Loss (% loss)	Throughput (kbits / s)
1	0.65	0	3871
2	0.45	0	3164

3	0.55	0	2258

Tabel 1 Percobaan Kondisi Tanpa Bottleneck

Pada kondisi ini menjalankan *traffic* video tanpa adanya kondisi bottleneck, table 1 menjelaskan parameter *latency*, *throughput*, dan *packet-loss* dalam keadaan yang sangat baik. Pada tabel diatas menunjukan nilai *latency* yang sangat rendah yaitu dibawah 1 ms, packet-loss menunjukan tidak ada packet yang didrop dalam simulasi tanpa bottleneck, dan throughput menunjukan angka yang sangat tinggi.

4.2 Kondisi Bottleneck

a. Performansi Drop Tail

Traffic Shaping (kbps)		Latency (ms)			Throughput	Dropped Packets		
		Min	Max	Mean	kbits / s	xmt	rcv	%loss
250	250	727	6574	4668	1999	100	92	8
200	200	3143	9434	6180	1599	100	82	18
150	150	4040	9106	8863	1199	100	76	24

Tabel 2 Percobaan Performansi Drop Tail

Pada tabel 2 menjelaskan performansi Drop Tail pada setiap scenario bottleneck yang diujikan yaitu 250 kbps, 200 kbps dan 150 kbps. Tabel tersebut juga menjelaskan bahwa semakin kecil nilai skenario akan semakin besar nilai *latency* dan *packet-loss*, sedangkan *throughput* akan semakin kecil.

b. Performansi RED

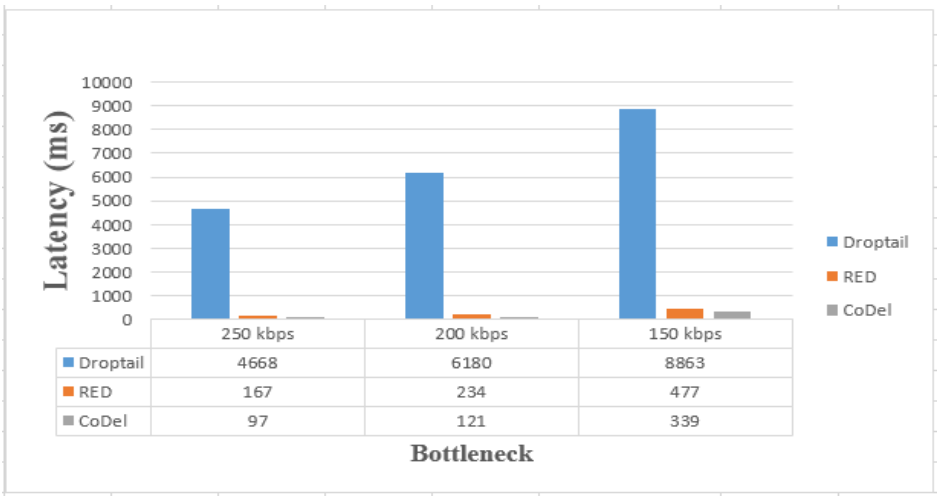
Traffic Shaping (kbps)		Latency (ms)			Throughput	Dropped Packets		
		Min	Max	Mean	kbits / s	xmt	rcv	%loss
250	250	0.52	1442	167	1911	100	77	23
200	200	0.46	458	234	1521	100	73	27
150	150	239	664	447	1198	100	66	34

Tabel 3 Percobaan Performansi RED

Pada tabel 3 menjelaskan performansi Drop Tail pada setiap skenario yang diujikan. Tabel tersebut juga menjelaskan bahwa semakin kecil nilai skenario akan semakin besar nilai *latency* dan *packet-loss*, sedangkan *throughput* akan semakin kecil

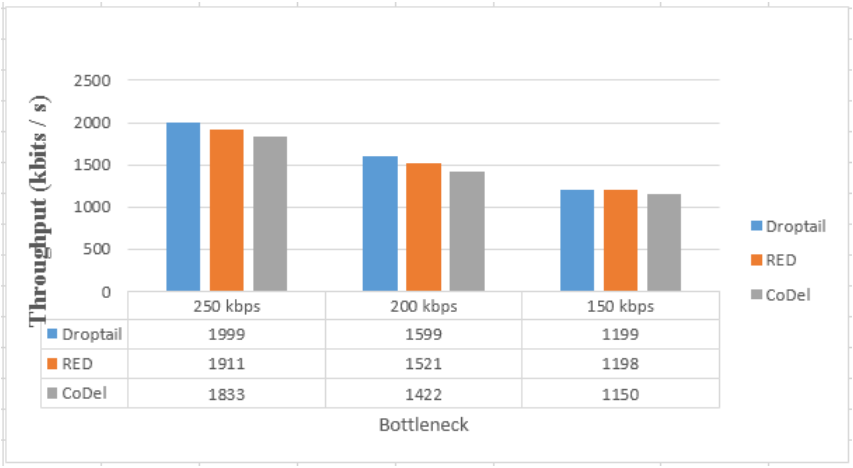
c. Perbandingan CoDel, RED, dan Drop Tail

Sub bab ini akan menyajikan perbandingan performansi CoDel, RED, dan Drop Tail. Parameter uji yang akan digunakan untuk analisis adalah *latency*, *throughput*, dan *packet-loss.*



Gambar 7 Latency CoDel, RED, dan Drop Tail

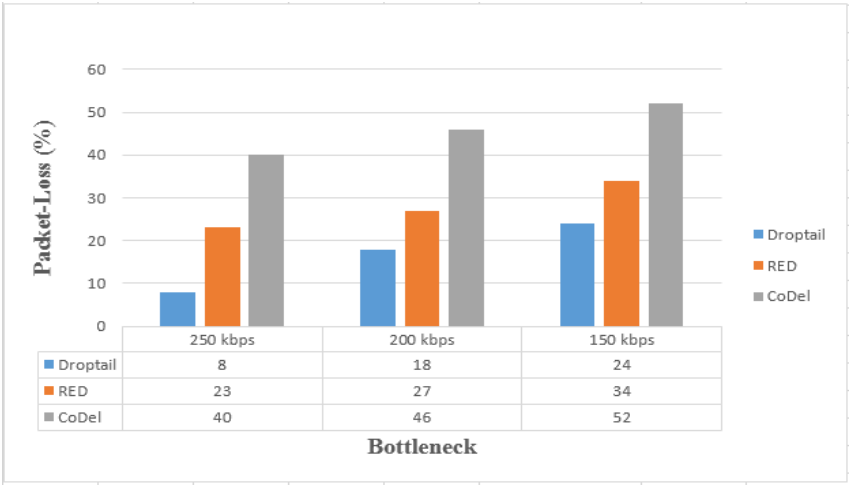
Gambar 7 menyajikan perbandingan latency pada algoritma CoDel, RED, dan Drop Tail. Gambar tersebut menjelaskan bahwa performansi CoDel lebih unggul dalam menjaga *latency* tetap rendah dibandingkan dengan algoritma RED dan Drop Tail. Hal ini disebabkan karena pada performansi algoritma CoDel menggunakan *timestamp* untuk menandai setiap paket didalam antrian. Maka lamanya paket didalam antrian harus sesuai dengan *timestamp*. Ketika paket melebihi *timestamp* akibatnya paket tersebut akan di-drop agar menjaga antrian tidak penuh dan menjaga *latency* tetap rendah. Hasil yang sama juga dijelaskan oleh Tanvi Sharma (2011) bahwa CoDel unggul dalam menjaga latency.



Gambar 8 Throughput CoDel, RED, dan Drop Tail

Gambar 8 menyajikan perbandingan *throughput* pada algoritma CoDel, RED, dan Drop Tail. Gambar tersebut menjelaskan bahwa performansi Drop Tail lebih unggul dalam parameter uji *throughput* dibandingkan dengan algoritma RED dan CoDel. Hal ini disebabkan pada performansi CoDel menggunakan *timestamp* pada setiap paket, ketika terjadi bufferbloat atau antrian yang cenderung penuh akan semakin banyak paket yang di-*drop* sehingga menyebabkan paket yang

diterima semakin sedikit. Hasil yang sama juga dijelaskan oleh Ganesh Patil dan Sally McClean (2010) bahwa Drop Tail lebih unggul dalam parameter uji *throughput*.



Gambar 7 Packet-Loss CoDel, RED, dan Drop Tail

Gambar 7 menyajikan perbandingan *packet-loss* pada algoritma CoDel, RED, dan Drop Tail. Gambar tersebut menjelaskan bahwa performansi Drop Tail lebih unggul dalam parameter uji *packet-loss* dibandingkan dengan algoritma RED dan CoDel. Hal ini disebabkan pada performansi CoDel menggunakan *timestamp* pada setiap paket, ketika terjadi bufferbloat atau antrian yang cenderung penuh akan semakin banyak paket yang di-*drop*. Oleh karena itu, nilai *packet-loss* yang dihasilkan akan semakin tinggi.

V. Conclusion

Tujuan riset untuk melakukan studi anlisis terhadap performansi algoritma CoDel, RED, dan Drop Tail telah dicapai dengan baik. Hasil perbandingan terhadap ketiga algoritma tersebut disajikan dalam sub bab 4.2. Dari hasil tersebut dijelaskan bahwa performansi algoritma CoDel unggul 90 % dalam menjaga *latency* tetap rendah dibandingkan dengan performansi RED, sedangkan nilai yang sangat signifikan besar ditunjukan oleh algoritma DropTail. Hal ini dikarenakan kelebihan algoritma CoDel yang menggunakan *timestamp* untuk menjaga antrian agar tidak penuh dan latency tetap rendah. Namun algoritma CoDel masih memiliki kelemahan didalam menangani *throughput* dan *packet-loss*. Drop Tail dan RED lebih unggul 20% dalam menangani *throughput* dan nilai *packet-loss* yang ditunjukan 10% lebih kecil dibandingkan dengan algoritma CoDel.

Kedepan riset ini focus pada peningkatan throughput dalam algoritma CoDel. Ada beberapa hal yang menjadi focus riset berikutnya sehingga throughput pada CoDel bias ditingkatkan, seperti memperbesar nilai target pada CoDel dan memperbesar nilai limit paket antrian pada CoDel.

References

[1] Braden, R., dan teman teman. 1998. "Recommendations on Queue Management and Congestion Avoidance in the Internet, RFC2309 (Informational), Internet Engineering Task Force." Internet Engineering Task Force, RFC2309 (Informational). April. http://www.ietf.org/rfc/rfc2309.txt.

[2] Gettys, J., Kathleen N. 2011. "Bufferbloat: Dark Buffers in the Internet."AQM Queue. November. http://queue.acm.org/detail.cfm?id=2071893.

[3] Gettys, J., Kathleen N., dan teman-teman. 2014. http://www.bufferbloat.net. Agustus 12. http://www.bufferbloat.net/.

[4] Greg W., Dan R. 2013. "Active Queue Management Algorithms DOCSIS 3.0." CableLabs. April.

http://www.cablelabs.com/wp-content/uploads/2013/11/Active_Queue_Management_Algorithms_DOCSIS_3_0.pdf.

[5] Hoiland-Jorgensen, Toke. 2012. "Battling Bufferbloat: An experimental comparison of four approaches to queue management in Linux Master module project Computer Science". RUDAR (Roskilde University Digita Archive. Desember. http://rudar.ruc.dk/handle/1800/9322.

[6] Høiland-Jørgensen, Toke. 2014. "Netperf Wrapper-Python wrapper to run multiple simultaneous netperf instances and aggregate the results". Accessed November 2014. github.com/tohojo/netperf-wrapper

[7] Jacobson, V., Kathleen, N. 2012. "Controlling Queue Delay - A modern AQM is just one piece of the solution to bufferbloat". Asscociation for Computing Machinery (ACM Queue). Mei. http://queue.acm.org/detail.cfm?id=2209336.

[8] Naeem, K., David, R., Michael, W. 2014. "The new AQM kids on the block: An experimental evaluation of CoDel and PIE". IEEE Xplore 85-90.

[9] Nichols, K., Jacobson, V. 2014. "Controlled Delay Active Queue Management draft-ietf-aqm-codel-00". Internet Engineering Task Force. Oktober 24. http://www.ietf.org/id/draft-ietf-aqm-codel-00.txt.

[10] Preethi Rao V., Mohit P. Tahiliani, Udaya Kumar K. Shenoy. 2014. "Analysis of sfqCoDel for Active Queue Management". IEEE Xplore 262-267

[11] Raghuvanshi, D.M., B. Annappa, and Mohit P. T. 2013. "On the Effectiveness of CoDel for Active Queue Management". IEEE Computer Society, In Proceedings of Third International Conference on Advanced Computing & Communication Technologies, ACCT107114.

[12] Ryu, Seungwan. 2002. "Active Queue Management (AQM) based Internet Congestion Control". University at Buffalo. Oktober. http://www.cse.buffalo.edu/~qiao/cse620/fall04/AQM-Fall04.pdf.

[13] Sally, F., Van, J. 1993. "Random Early Detection Gateways for Congestion Avoidance". Lawrence Berkeley Laboratory. Agustus. http://www.icir.org/floyd/papers/early.twocolumn.pdf.

[14] Sharma, Tanvi. 2014. "Controlling Queue Delay (CoDel) to counter the Bufferbloat Problem in Internet". INPRESSCO International Journal of Current Engineering and Technology. Juni. http://inpressco.com/wp-content/uploads/2014/07/Paper1992210-2215.pdf.

[15] Taht, Dave. 2012. "RFC: Realtime Response Under Load (rrul) test specification". GMANE. September. http://article.gmane.org/gmane.network.routing.bufferbloat/940/.

[16] Tannenbaum, A.S. 2011. "Computer Network 5th Edition". New Jersey: Prentice Hall, Inc

.