BAB 5 PENGUJIAN DAN ANALISIS HASIL

Dalam bab pengujian dan analisis hasil ini akan dibahas mengenai hasil pengujian yang telah dilakukan serta melakukan analisis dari hasil pengujian tersebut, sehingga akan didapat perbandingan kinerja TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dan *Droptail* yang nantinya pada bab terakhir akan diibuat kesimpulan dari penelitian ini.

5.1 Pengujian

Berdasar rancangan dan implementasi simulasi yang telah dilakukan pada penelitian ini, maka pengujian perlu dilakukan untuk mendapatkan hasil yang diharapkan. Berikut merupakan hasil pengujian yang telah dilakukan.

5.1.1 Hasil Pengujian Varian TCP Dengan *Droptail*

Dari pengujian TCP Vegas dengan TCP New Reno menggunakan antrian *Droptail* yang telah dilakukan, rata-rata paket yang telah dikirimkan selama 300 detik pada topologi *Abilene* dalam simulasi ini yaitu untuk TCP Vegas 3.461.311 paket dan untuk TCP New Reno 3.564.456 paket. Untuk melihat perbandingan kinerja kedua TCP menggunakan antrian *Droptail*, maka dibutuhkan parameter hasil yaitu *packet delivery ratio, throughput, delay* dan *packet drop*. Berikut perbandingan kinerja dari setiap parameter hasil setelah dilakukan pengujian.

5.1.1.1 Hasil Pengujian *Packet Delivery Ratio*

Packet Delivery Ratio merupakan perbandingan jumlah paket yang diterima oleh *node* tujuan dan jumlah paket yang dikirimkan oleh *node* sumber dalam kurun waktu tertentu. Dalam penelitian ini, pengujian *packet delivery ratio* dilihat dari perbandingan jumlah paket yang diterima dan dikirim ketika TCP Vegas dan TCP New Reno mengirimkan paket data menggunakan antrian *Droptail* dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.1 Hasil Pengujian *Packet Delivery Ratio* Penambahan *Buffer*

Kapasitas *Buffer*	*Packet Delivery Ratio (%)*	
	TCP Vegas	TCP New Reno
20 paket	99.996	99.263
30 paket	99.997	99.413
40 paket	99.997	99.546
50 paket	99.997	99.649
60 paket	99.997	99.844

Tabel 5.1 menunjukkan nilai *packet delivery ratio* dari TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* yang dihasilkan dari lima kali pengujian yang tiap kali pengujiannya menggunakan kapasitas *buffer* yang berbeda. Pada pengujian ini akan dilakukan penambahan kapasitas *buffer* untuk tiap kali

pengujian yaitu 20, 30, 40, 50 dan 60 paket. Penambahan kapasitas *buffer* digunakan untuk menguji kinerja jaringan. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.1.2 Hasil Pengujian Hasil *Throughput*

Throughput merupakan banyaknya paket yang tiba atau diterima dalam interval waktu yang ditentukan. Dalam penelitian ini, pengujian *throughput* dilihat dari banyaknya paket yang diterima TCP tujuan dalam satuan waktu ketika TCP Vegas dan TCP New Reno mengirimkan paket data menggunakan antrian *Droptail* dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.2 Hasil Pengujian *Throughput* Penambahan *Buffer*

Kapasitas *Buffer*	*Throughput (kbps)*	
	TCP Vegas	TCP New Reno
20 paket	95354.133	99086.748
30 paket	95654.635	99943.298
40 paket	94552.551	100571.680
50 paket	93661.378	100721.284
60 paket	93661.378	103424.107

Tabel 5.2 menunjukkan nilai *throughput* dari TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* yang di hasilkan dari lima kali pengujian yang tiap kali pengujiannya menggunakan kapasitas *buffer* yang berbeda. Dalam penelitian ini di dapatkan minimum kapasitas *buffer* yaitu 20 paket. Pada pengujian ini akan dilakukan penambahan kapasitas *buffer* untuk tiap kali pengujian yaitu 20, 30, 40, 50 dan 60 paket. Penambahan kapasitas *buffer* digunakan untuk menguji kinerja jaringan. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.1.3 Hasil Pengujian *Delay*

Delay merupakan waktu yang dibutuhkan suatu paket saat dikirimkan dari *node* sumber ke *node* tujuan. Dalam penelitian ini, pengujian *delay* dilihat dari waktu yang dibutuhkan TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* saat mengirimkan paket data, dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.3 Hasil Pengujian *Delay* Penambahan *Buffer*

Kapasitas *Buffer*	*Delay (ms)*	
	TCP Vegas	TCP New Reno
20 paket	4.173	4.456
30 paket	4.572	5.326
40 paket	4.884	6.152
50 paket	4.929	7.025
60 paket	4.929	8.688

Tabel 5.3 menunjukkan nilai *delay* dari TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* yang di hasilkan dari lima kali pengujian yang tiap kali pengujiannya menggunakan kapasitas *buffer* yang berbeda. Dalam penelitian ini di dapatkan minimum kapasitas *buffer* yaitu 20 paket. Pada pengujian ini akan dilakukan penambahan kapasitas *buffer* untuk tiap kali pengujian yaitu 20, 30, 40, 50 dan 60 paket. Penambahan kapasitas *buffer* digunakan untuk menguji kinerja jaringan. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.1.4 Hasil Pengujian *Packet Drop*

Packet drop merupakan paket yang terbuang saat proses transmisi dari *node* sumber ke tujuan. Dalam penelitian ini, pengujian *packet drop* dilihat dari seberapa banyak paket yang terbuang ketika TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* mengirimkan paket data, dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.4 Hasil Pengujian *Packet Drop* Penambahan *Buffer*

Kapasitas *Buffer*	*Packet Drop* (%)	
	TCP Vegas	TCP New Reno
20 paket	0.002	0.734
30 paket	0.001	0.584
40 paket	0.000	0.450
50 paket	0.000	0.347
60 paket	0.000	0.151

Tabel 5.4 menunjukkan nilai *packet drop* dari TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* yang di hasilkan dari lima kali pengujian yang tiap kali pengujiannya menggunakan kapasitas *buffer* yang berbeda. Dalam penelitian ini di dapatkan minimum kapasitas *buffer* yaitu 20 paket. Pada pengujian ini akan dilakukan penambahan kapasitas *buffer* untuk tiap kali pengujian yaitu 20, 30, 40, 50 dan 60 paket. Penambahan kapasitas *buffer* digunakan untuk menguji kinerja jaringan. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran

paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.2 Hasil Pengujian Varian TCP Dengan *Random Early Detection*

Dari pengujian TCP Vegas dengan TCP New Reno menggunakan antrian *Random Early Detection* yang telah dilakukan, rata-rata paket yang telah dikirimkan selama 300 detik pada topologi *Abilene* dalam simulasi ini yaitu untuk TCP Vegas menggunakan penambahan *min thresh* yaitu 3.273.677 paket sedangkan penambahan *max thresh* yaitu 3.418.598 paket dan untuk TCP New Reno menggunakan penambahan *min thresh* 3,370.047 yaitu paket sedangkan penambahan *max thresh* yaitu 3.446.149 paket. Untuk melihat perbandingan kinerja kedua TCP menggunakan antrian *Random Early Detection*, maka dibutuhkan parameter hasil yaitu *packet delivery ratio, throughput, delay* dan *packet drop*. Berikut perbandingan kinerja dari setiap parameter hasil setelah dilakukan pengujian.

5.1.2.1 Hasil Pengujian *Packet Delivery Ratio*

Packet Delivery Ratio merupakan perbandingan jumlah paket yang diterima oleh *node* tujuan dan jumlah paket yang dikirimkan oleh *node* sumber dalam kurun waktu tertentu. Dalam penelitian ini, pengujian *packet delivery ratio* dilihat dari perbandingan jumlah paket yang diterima dan dikirim ketika TCP Vegas dan TCP New Reno mengirimkan paket data menggunakan antrian *Random Early Detection* dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.5 Hasil Pengujian *Packet Delivery Ratio* Penambahan *Min Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Packet Delivery Ratio (%)*	
			TCP Vegas	TCP New Reno
60 paket	20 paket	50 paket	99.996	99.674
	25 paket		99.997	99.688
	30 paket		99.997	99.685
	35 paket		99.997	99.681
	40 paket		99.997	99.656

Tabel 5.5 menunjukkan nilai *packet delivery ratio* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dengan *max thresh* 50 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

Tabel 5.6 Hasil Pengujian *Packet Delivery Ratio* Penambahan *Max Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Packet Delivery Ratio (%)*	
			TCP Vegas	TCP New Reno
60 paket	20 paket	30 paket	99.997	99.570
		35 paket	99.997	99.615
		40 paket	99.997	99.639
		45 paket	99.997	99.658
		50 paket	99.996	99.674

Tabel 5.6 menunjukkan nilai *packet delivery ratio* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh.* Pada pengujian ini akan dilakukan dengan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket dengan *min thresh* 20 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.2.2 Hasil Pengujian *Thoughput*

Throughput merupakan banyaknya paket yang tiba atau diterima dalam interval waktu yang ditentukan. Dalam penelitian ini, pengujian *throughput* dilihat dari banyaknya paket yang diterima TCP tujuan dalam satuan waktu ketika TCP Vegas dan TCP New Reno mengirimkan paket data menggunakan antrian *Random Early Detection* dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.7 Hasil Pengujian *Throughput* Penambahan *Min Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Throughput (kbps)*	
			TCP Vegas	TCP New Reno
60 paket	20 paket	50 paket	93066.480	97715.963
	25 paket		93706.947	97911.646
	30 paket		94404.364	97516.132
	35 paket		92496.451	96634.191
	40 paket		93661.378	98471.612

Tabel 5.7 menunjukkan nilai *throughput* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dengan *max thresh* 50 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan

setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

Tabel 5.8 Hasil Pengujian *Throughput* Penambahan *Max Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Throughput (kbps)*	
			TCP Vegas	TCP New Reno
60 paket	20 paket	30 paket	93592.601	96945.469
		35 paket	94409.096	97424.139
		40 paket	93627.144	97565.775
		45 paket	92353.098	97798.648
		50 paket	93066.480	97715.963

Tabel 5.8 menunjukkan nilai *throughput* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan dengan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket dengan *min thresh* 20 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.2.3 Hasil Pengujian *Delay*

Delay merupakan waktu yang dibutuhkan suatu paket saat dikirimkan dari *node* sumber ke *node* tujuan. Dalam penelitian ini, pengujian *delay* dilihat dari waktu yang dibutuhkan TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* saat mengirimkan paket data, dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.9 Hasil Pengujian *Delay* Penambahan *Min Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Delay (ms)*	
			TCP Vegas	TCP New Reno
60 paket	20 paket	50 paket	4.311	6.309
	25 paket		4.496	6.577
	30 paket		4.677	6.782
	35 paket		4.970	7.034
	40 paket		4.929	7.167

Tabel 5.9 menunjukkan nilai *delay* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dengan *max thresh* 50 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik,

dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

Tabel 5.10 Hasil Pengujian *Delay* Penambahan *Max Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Delay (ms)*	
			TCP Vegas	TCP New Reno
60 paket	20 paket	30 paket	4.285	5.617
		35 paket	4.291	5.861
		40 paket	4.327	6.051
		45 paket	4.343	6.104
		50 paket	4.311	6.309

Tabel 5.10 menunjukkan nilai *delay* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan dengan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket dengan *min thresh* 20 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

5.1.2.4 Hasil Pengujian *Packet Drop*

Packet drop merupakan paket yang terbuang saat proses transmisi dari *node* sumber ke tujuan. Dalam penelitian ini, pengujian *packet drop* dilihat dari seberapa banyak paket yang terbuang ketika TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* mengirimkan paket, dalam keadaan jaringan yang mengalami peningkatan *traffic* sehingga menyebabkan *congestion.*

Tabel 5.11 Hasil Pengujian *Packet Drop* Penambahan *Min Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Packet Drop* (%)	
			TCP Vegas	TCP New Reno
60 paket	20 paket	50 paket	0.001	0.322
	25 paket		0.000	0.308
	30 paket		0.000	0.311
	35 paket		0.000	0.314
	40 paket		0.000	0.340

Tabel 5.11 menunjukkan nilai *packet drop* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan penambahan *min thresh* yaitu 20 sesuai

dengan minimum *buffer*, 25, 30, 35 dan 40 paket dengan *max thresh* 50 paket. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms.

Tabel 5.12 Hasil Pengujian *Packet Drop* Penambahan *Max Thresh*

Kapasitas *Buffer*	Nilai *min thresh*	Nilai *max thresh*	*Packet Drop* (%)	
			TCP Vegas	TCP New Reno
60 paket	20 paket	30 paket	0.001	0.428
		35 paket	0.001	0.382
		40 paket	0.001	0.357
		45 paket	0.001	0.338
		50 paket	0.001	0.322

Tabel 5.12 menunjukkan nilai *packet drop* dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* yang di hasilkan dari lima kali pengujian. *Random Early Detection* memiliki dua parameter *min thresh* dan *max thresh*. Pada pengujian ini akan dilakukan dengan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket dengan *min thresh* 20 sesuai dengan minimum *buffer*. Untuk kapasitas *buffer* akan tetap yaitu 60 paket. Dan lama waktu simulasi untuk pengujian yaitu 300 detik, dengan ukuran paket yang digunakan yaitu 1024 *bytes*, *bandwidth* yang digunakan setiap *link* untuk menghubungkan semua *node* dalam topologi *Abilene* yaitu 10 Mbps dengan *delay* 2 ms

5.2 Analisis Hasil

Dari pengujian yang telah selesai dilakukan pada sub bab sebelumnya, didapat hasil pengujian yang ditampilkan yaitu *packet delivery ratio, throughput, delay* dan *packet drop*. Dari hasil pengujian itu, akan ditampilkan dalam bentuk grafik untuk mempermudah dalam menganalisa perbandingan kinerja dari TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dan *Droptail*.

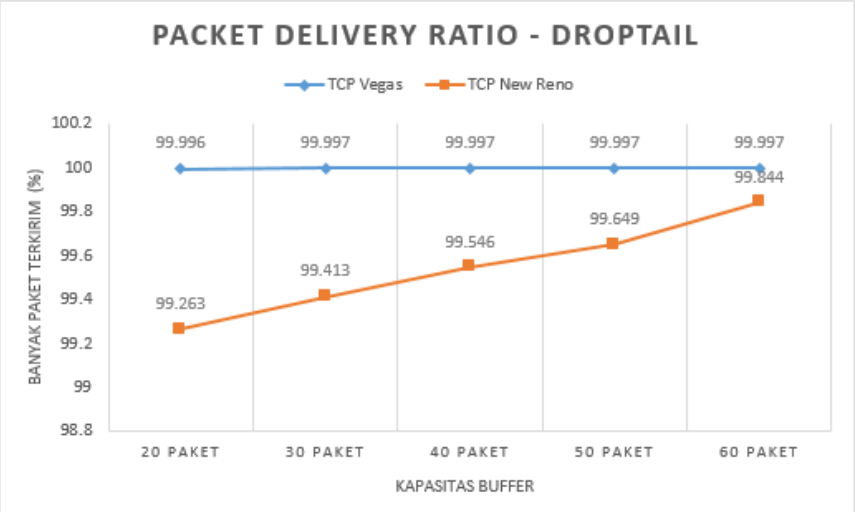
5.2.1 Analisis Hasil Varian TCP Dengan Droptail

5.2.1.1 Analisis Hasil *Packet Delivery Ratio*

Perbandingan *packet delivery ratio* antara TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* dengan penambahan kapasitas *buffer* yaitu mulai dari 20, 30, 40, 50 dan 60 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.1 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung datar, dimulai dari nilai 99.996 persen dengan kapasitas *buffer* 20 paket, kemudian grafik naik ke nilai 99.997 persen ketika kapasitas *buffer* 30, 40, 50 dan 60 paket.

- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 99.263 persen dengan kapasitas *buffer* 20 paket, kemudian grafik naik ke angka 99.413 persen, 99.546 persen, 99.649 persen, 99.844 persen dengan kapasitas *buffer* yaitu 30, 40, 50 dan 60 paket.
- Dari dua hasil tersebut, penambahan kapasitas *buffer* mempengaruhi peningkatan kinerja di sisi *packet delivery ratio* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah kapasitas *buffer*, semakin besar pula paket data yang dapat di layani.
- Meskipun saat menggunakan antrian *Droptail* kedua TCP mengalami kenaikan*,* namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Droptail* akan menampung paket data yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh, paket data yang datang akan di *drop* sehingga mengakibatkan *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *packet delivery ratio* TCP New Reno ketika menggunakan penambahan kapasitas *buffer* pada antrian *Droptail* yaitu karena semakin bertambah kapasitas *buffer*, semakin bertambah pula data yang ditampung untuk dilayani. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Droptail* akan menampung paket yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* yang berakibat pada turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Droptail* sehingga ukuran RTT tidak lagi besar. Alasan tingginya *packet delivery ratio* TCP Vegas ketika penambahan kapasitas *buffer* pada antrian *Droptail* yaitu karena meskipun kapasitas *buffer* bertambah, TCP Vegas mampu mengontrol pengiriman paket data secara stabil dengan melihat perubaha RTT, hal tersebut *membuat packet delivery ratio* TCP Vegas lebih unggul daripada TCP New Reno saat menggunakan antrian *Droptail*.



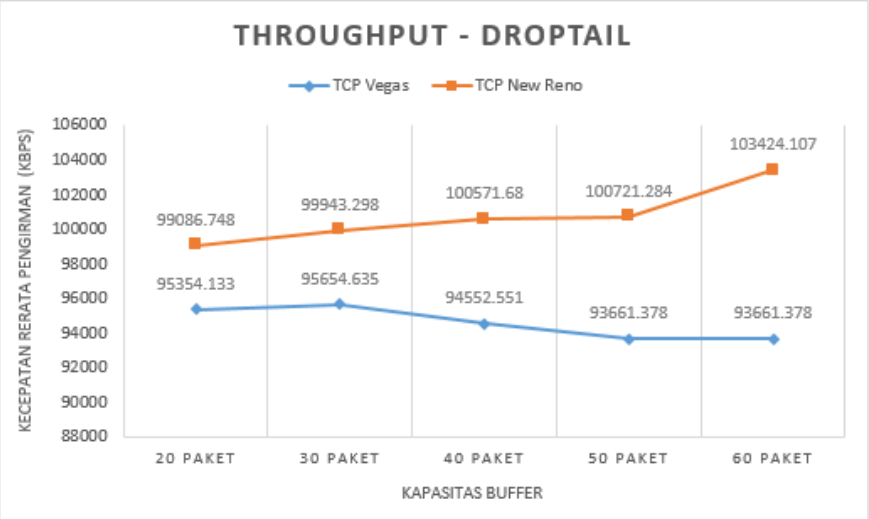
Gambar 5.1 Grafik Perbandingan *Packet Delivery Ratio* Penambahan *Buffer*

5.2.1.2 Analisis Hasil *Throughput*

Perbandingan *throughput* antara TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* dengan penambahan kapasitas *buffer* yaitu mulai dari 20, 30, 40, 50 dan 60 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.2 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung turun, dimulai dari nilai 95354.133 kbps dengan kapasitas *buffer* 20 paket, kemudian grafik naik ke angka 95654.635 kbps persen ketika kapasitas *buffer* 30 paket. Dan selanjutnya grafik mengalami penurunan ke angka 94552.551 kbps, 93661.378 kbps, 93661.378 kbps dengan kapasitas *buffer* yaitu 40, 50 dan 60 paket.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 99086.748 kbps dengan kapasitas *buffer* 20 paket, kemudian grafik naik ke nilai 99943.298 kbps, 100571.680 kbps, 100721.284 kbps, 103424.107 kbps dengan kapasitas *buffer* yaitu 30, 40, 50 dan 60 paket.
- Dari dua hasil tersebut, penambahan kapasitas *buffer* mempengaruhi peningkatan kinerja di sisi *throughput* pada TCP New Reno karena semakin besar kapasitas *buffer,* semakin banyak pula data yang dilayani, sedangkan untuk TCP Vegas meskipun sempat mengalami peningkatan ketika kapasitas *buffer* 30 paket, lalu mengalami penurunan.
- Hasil tersebut dapat dilihat bahwa saat menggunakan antrian *Droptail* TCP New Reno lebih baik dalam sisi *throughput* dibandingkan TCP Vegas. Karena TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Droptail* akan menampung paket yang datang hingga kapasitas *buffer*

penuh. Jika kapasitas *buffer* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Droptail* sehingga ukuran RTT tidak lagi besar. Alasan turunnya *throughput* TCP Vegas ketika penambahan kapasitas *buffer* pada antrian *Droptail* yaitu karena dengan penambahan kapasitas *buffer* maka data yang ditampung akan semakin banyak, hal tersebut akan membuat nilai RTT menjadi besar, ketika nilai RTT besar TCP Vegas menganggap jaringan mengalami *congestion* sehingga akan terjadi penurunan pengiriman data yang menyebabkan kinerja *throughput* menurun seiring bertambahnya kapasitas *buffer* pada natrian *Droptail.* Sedangkan TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Droptail* akan menampung paket data yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh, paket data yang datang akan di *drop* sehingga mengakibatkan *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman akan kembali dari awal sehingga kepadatan antrian akan berkurang Alasan meningkatnya *throughput* TCP New Reno ketika menggunakan penambahan kapasitas *buffer* pada antrian *Droptail* yaitu karena semakin bertambah kapasitas *buffer*, semakin bertambah pula data yang ditampung untuk dilayani. Dan meskipun akhirnya terjadi *packet drop* namun akan dilanjutkan ke fase *fast recovery* untuk mempertahankan *throughput* tetap tinggi. Hal tersebut membuat *throughput* TCP New Reno lebih baik daripada TCP Vegas saat menggunakan antrian *Droptail*.



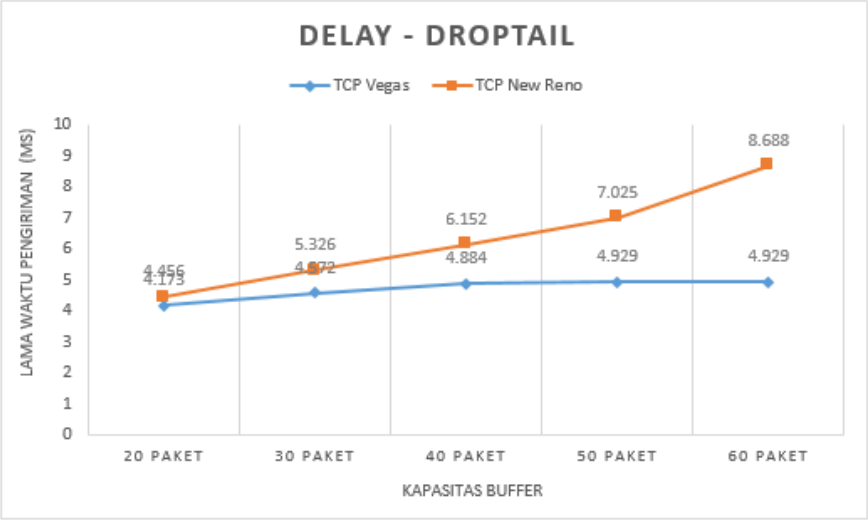
Gambar 5.2 Grafik Perbandingan *Throughput* Penambahan *Buffer*

5.2.1.3 Analisis Hasil *Delay*

Perbandingan *delay* antara TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* dengan penambahan kapasitas *buffer* yaitu mulai dari 20, 30, 40, 50 dan 60 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.3 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung naik, dimulai dari nilai 4.173 ms, 4.572 ms, 4.884 ms, 4.929 ms dan 4.929 ms dengan kapasitas *buffer* yaitu 20, 30, 40, 50 dan 60 persen.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 4.456 ms, 5.326 ms, 6.152 ms, 7.025 ms dan 8.688 ms dengan kapasitas *buffer* yaitu 20, 30, 40, 50 dan 60 paket.
- Dari dua hasil tersebut, penambahan kapasitas *buffer* mempengaruhi peningkatan di sisi *delay* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah kapasitas *buffer*, semakin besar pula paket data yang dapat di layani dan semakin lama pula waktu pelayanan untuk paket data.
- Meskipun saat menggunakan antrian *Droptail* kedua TCP mengalami kenaikan*,* namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Droptail* akan menampung paket data yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh, paket data yang datang akan di *drop* sehingga mengakibatkan *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman paket data akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *delay* TCP New Reno ketika menggunakan penambahan kapasitas *buffer* pada antrian *Droptail* yaitu karena semakin bertambah kapasitas *buffer*, semakin bertambah pula data yang ditampung untuk dilayani dan hal tersebut menyebabkan waktu untuk melayani paket data menjadi bertambah lama. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Droptail* akan menampung paket yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* untuk menyesuaikan pengirman paket data yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri

antrian *Droptail* sehingga ukuran RTT tidak lagi besar dan menyesuaikan ukuran kapasitas *buffer*. Alasan meningkatnya *delay* TCP Vegas ketika penambahan kapasitas *buffer* pada antrian *Droptail* yaitu karena dengan bertambahnya kapasitas *buffer*, paket data yang ditampung akan menjadi semakin banyak, hal tersebut membuat nilai RTT menjadi besar yang membuat TCP Vegas mengganggap jaringan mengalami *congestion,* ketika *congestion* terjadi pengiriman paket data akan menurun sehingga banyak data yang dapat di tampung pada *buffer* tidak terlalu banyak dan hal tersebut membuat waktu tunggu paket untuk dilayani semakin kecil. Hal tersebut membuat *delay* TCP Vegas lebih rendah daripada TCP New Reno saat menggunakan antrian *Droptail*.



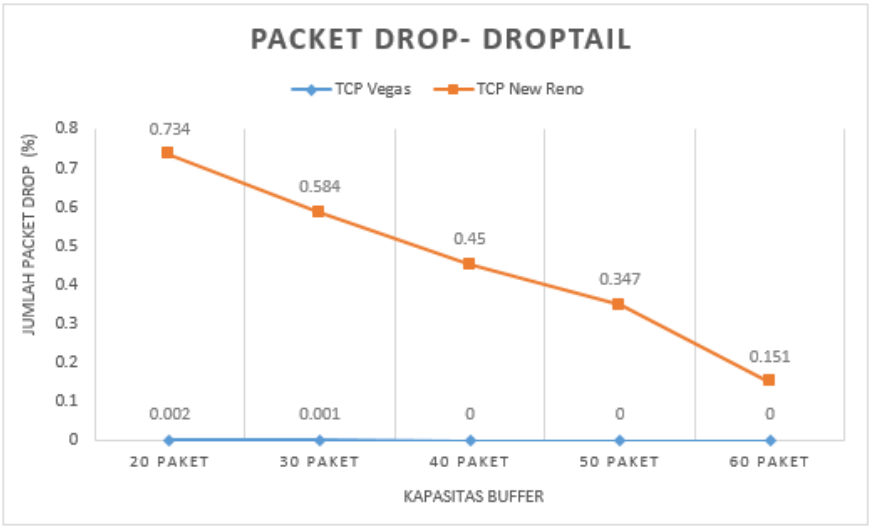
Gambar 5.3 Grafik Perbandingan *Delay* Penambahan *Buffer*

5.2.1.4 Analisis Hasil *Packet Drop*

Perbandingan *packet drop* antara TCP Vegas dan TCP New Reno menggunakan antrian *Droptail* dengan penambahan kapasitas *buffer* yaitu mulai dari 20, 30, 40, 50 dan 60 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.4 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung turun, dimulai dari nilai 0.002 persen dengan kapasitas *buffer* 20 paket, kemudian grafik turun ke nilai 0.001 persen ketika kapasitas *buffer* 30 paket. Dan selanjutnya grafik akan turun dan tetap konstan di nilai 0 persen ketika kapasitas *buffer* naik menjadi 40, 50 dan 60 persen.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 0.734 persen dengan kapasitas *buffer* 20 paket, kemudian grafik turun ke nilai 0.584 persen ketika kapasitas *buffer* 30 paket, selanjutnya grafik akan terus turun ke nilai 0.450 persen, 0.347 persen, 0.151 persen dengan kapasitas *buffer* yaitu 40, 50 dan 60 paket.

- Dari dua hasil tersebut, penambahan kapasitas *buffer* mempengaruhi penurunan di sisi *packet drop* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah kapasitas *buffer,* semakin besar paket data yang dapat di layani dan semakin sedikit pula paket yang di *drop*.
- Meskipun saat menggunakan antrian *Droptail* kedua TCP mengalami penurunan*,* namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Droptail* akan menampung paket data yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh, paket data yang datang akan di *drop* sehingga mengakibatkan *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman paket data akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan turunnya *packet drop* TCP New Reno ketika menggunakan penambahan kapasitas *buffer* pada antrian *Droptail* yaitu dengan kapasitas *buffer* yang kecil, ketika paket data yang datang lebih banyak dari pada kapasitas *buffer* mengakibatkan *packet drop* akan sering terjadi*.* Dengan penambahan kapasitas *buffer*, paket data yang dapat dilayani akan bertambah dan *packet drop* akan menjadi turun*.* Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Droptail* akan menampung paket yang datang hingga kapasitas *buffer* penuh. Jika kapasitas *buffer* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* untuk menyesuaikan pengiriman paket data yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Droptail* sehingga ukuran RTT tidak lagi besar dan menyesuaikan ukuran kapasitas *buffer*. Alasan turunnya *packet drop* TCP Vegas ketika penambahan kapasitas *buffer* pada antrian *Droptail* yaitu dengan bertambahnya kapasitas *buffer,* paket data yang ditampung semakin banyak menyebabkan RTT menjadi besar dan TCP Vegas menganggap jaringan terjadi *congestion*, sehingga data yang dikirim akan dikurangi dan menyebabkan data yang dapat ditampung pada *buffer* tidak terlalu banyak dan hal tersebut membuat *packet drop* menjadi semakin berkurang. Hal tersebut membuat *packet drop* TCP Vegas lebih rendah daripada TCP New Reno saat menggunakan antrian *Droptail*.



Gambar 5.4 Grafik Perbandingan *Packet Drop* Penambahan *Buffer*

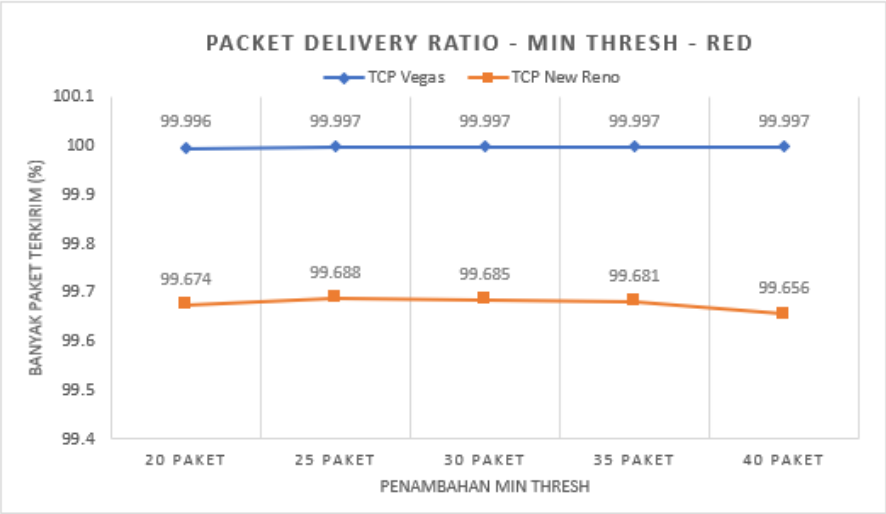
5.2.2 Analisis Hasil Varian TCP Dengan *Random Early Detection*

5.2.2.1 Analisis Hasil *Packet Delivery Ratio*

Perbandingan *packet delivery ratio* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dan *max thresh* yaitu 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.5 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung naik, dimulai dari nilai 99.996 persen dengan *min thresh* 20 paket, kemudian grafik naik secara konstan ke nilai 99.997 persen ketika *min thresh* 25, 30, 35 dan 40 paket.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung turun, dimulai dari nilai 99.674 persen dengan *min thresh* 20 paket, kemudian grafik naik ke nilai 99.688 persen ketika *min thresh* 25 paket, kemudian grafik akan turun ke nilai 99.685 persen, 99.681 persen dan 99.656 persen ketika *min thresh* yaitu 30, 35 dan 40 paket.
- Dari dua hasil tersebut, penambahan *min thresh* mempengaruhi peningkatan di sisi *packet delivery ratio* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah *min thresh*, semakin besar pula paket data yang dapat di layani.
- Meskipun saat menggunakan antrian *Random Early Detection* kedua TCP mengalami kenaikan, namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random*

Early Detection akan menampung paket data yang datang hingga batas *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop*, sehingga ketika terjadi *packet drop*, *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *packet delivery ratio* TCP New Reno ketika menggunakan penambahan *min thresh* pada antrian *Random Early Detection* yaitu karena semakin bertambah *min thresh*, semakin bertambah pula data yang ditampung untuk dilayani. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* yang berakibat pada turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar. Alasan tingginya *packet delivery ratio* TCP Vegas ketika penambahan *min thresh* pada antrian *Random Early Detection* yaitu karena meskipun *min thresh* bertambah, TCP Vegas mampu mengontrol pengiriman paket data secara stabil dengan melihat perubahan RTT, hal tersebut *membuat packet delivery ratio* TCP Vegas lebih unggul daripada TCP New Reno saat menggunakan antrian *Random Early Detection*.



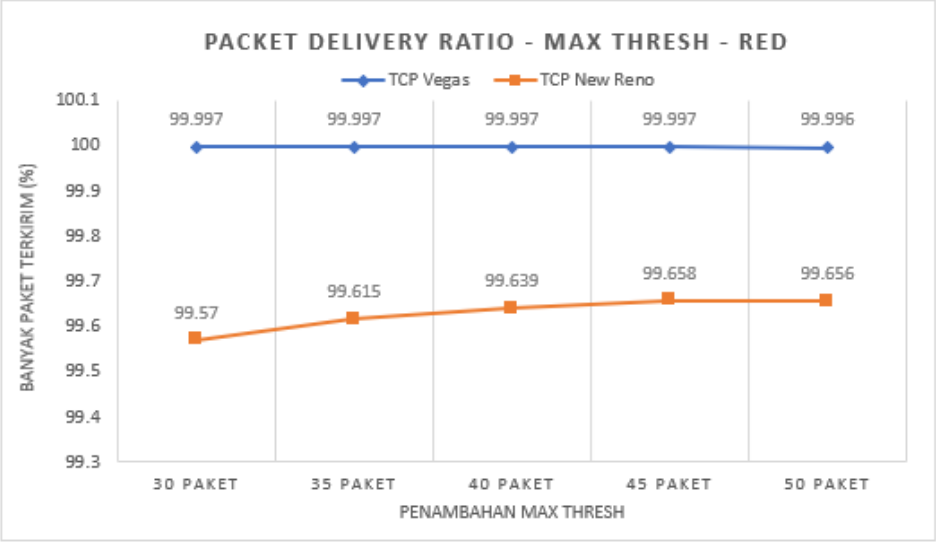
Gambar 5.5 Grafik Perbandingan *Packet Delivery Ratio* Penambahan *Min Thresh*

Perbandingan *packet delivery ratio* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan *min thresh* yaitu 20 paket dan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.6 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan pergerakan grafik yang konstan yaitu pada nilai 99.997 persen ketika *max thresh* 30, 35, 40 dan 45 paket, kemudian grafik turun ke nilai 99.996 persen ketika *max thresh* naik menjadi 50 paket.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 99.570 persen dengan *max thresh* 30 paket, kemudian grafik naik lagi ke nilai 99.615 persen ketika *max thresh* 35 paket, kemudian grafik naik terus ke nilai 99.639 persen, 99.658 persen dan 99.674 persen dengan *max thresh* yaitu 40, 45 dan 50 paket.

- Dari dua hasil tersebut, penambahan *max thresh* mempengaruhi peningkatan di sisi *packet delivery ratio* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah *max thresh,* semakin besar paket data yang akan ditandai atau di *drop* secara *random* dan kemungkinan paket *drop* bisa di minimalisir.
- Meskipun saat menggunakan antrian *Random Early Detection* kedua TCP mengalami kenaikan*,* namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga batas *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop*, sehingga ketika terjadi *packet drop*, *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *packet delivery ratio* TCP New Reno ketika menggunakan penambahan *max thresh* pada antrian *Random Early Detection* yaitu karena semakin bertambah *max thresh*, semakin bertambah pula data yang ditampung untuk dilayani dan di *drop* secara *random* serta semakin sedikit paket data yang di *drop* secara langsung. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,*

sehingga akan mengurangi *congestion window* yang berakibat pada turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar. Alasan tingginya *packet delivery ratio* TCP Vegas ketika penambahan *max thresh* pada antrian *Random Early Detection* yaitu karena meskipun *max thresh* bertambah, TCP Vegas mampu mengontrol pengiriman paket data secara stabil dengan melihat perubahan RTT, hal tersebut *membuat packet delivery ratio* TCP Vegas lebih unggul daripada TCP New Reno saat menggunakan antrian *Random Early Detection*.



Gambar 5.6 Grafik Perbandingan *Packet Delivery Ratio* Penambahan *Max Thresh*

5.2.2.2 Analisis Hasil *Throughput*

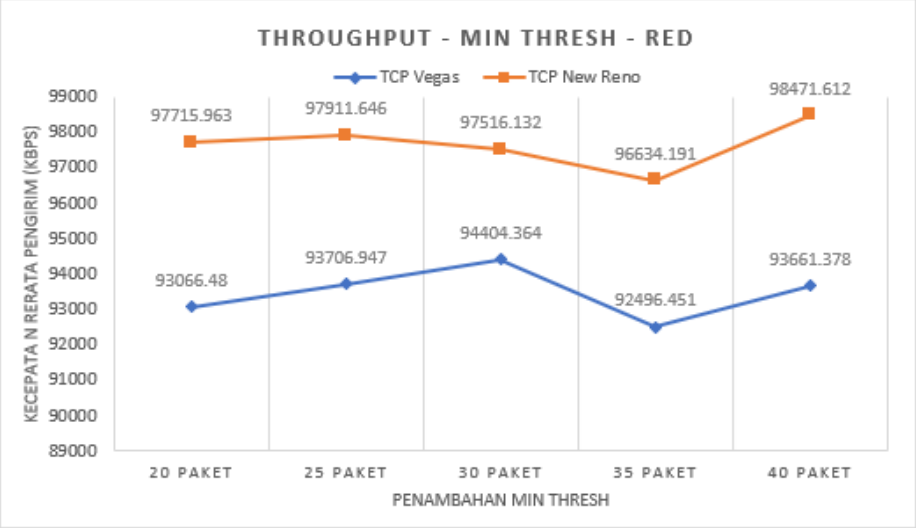
Perbandingan *throughput* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dan *max thresh* yaitu 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.7 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung naik, dimulai dari nilai 93066.480 kbps dengan *min thresh* 20 paket, kemudian grafik naik ke nilai 93706.947 kbps ketika *min thresh* 25 paket, kemudian grafik naik ke nilai 94404.364 kbps ketika *min thresh* 30 paket, kemudian grafik turun ke nilai 92496.451 kbps ketika *min thresh* 35 paket, kemudian grafik naik kembali ke nilai 93661.378 kbps ketika *min thresh* 40 persen.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang fluktuatif dimulai dari angka 97715.963 kbps dengan *min thresh* 20 paket, kemudian grafik naik ke nilai 97911.646 kbps ketika *min thresh* 25 paket, kemudian grafik turun ke angka

97516.132 kbps ketika *min thresh* 30 paket, kemudian grafik turun ke nilai 96634.191 paket ketika *min thresh* 35 paket*,* kemudian grafik akan naik kembali ke nilai 98471.612 kbps dengan *min thresh* yaitu 40 paket.

- Dari dua hasil tersebut, penambahan *min thresh* mempengaruhi peningkatan di sisi *throughput* pada TCP New Reno karena semakin besar *min thresh* semakin banyak pula paket data yang dilayani, sedangkan untuk TCP Vegas meskipun sempat mengalami peningkatan ketika *min thresh* 30 paket, namun setelahnya mengalami penurunan.
- Hasil tersebut dapat dilihat bahwa saat menggunakan antrian *Random Early Detection* TCP New Reno lebih baik dalam sisi *throughput* dibandingkan TCP Vegas. Karena TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar. Alasan naik turunnya *throughput* TCP Vegas ketika penambahan *min thresh* pada antrian *Random Early Detection* yaitu karena dengan penambahan *min thresh* maka data yang ditampung akan semakin banyak, hal tersebut akan membuat nilai RTT menjadi besar, ketika nilai RTT besar TCP Vegas menganggap jaringan mengalami *congestion* sehingga akan terjadi penurunan pengiriman data yang menyebabkan kinerja *throughput* menurun seiring bertambahnya *min thresh* pada atrian *Random Early Detection.* Sedangkan TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop*, sehingga ketika terjadi *packet drop*, *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *throughput* TCP New Reno ketika menggunakan penambahan *min thresh* pada antrian *Random Early Detection* yaitu karena semakin bertambah *min thresh*, semakin bertambah pula data yang ditampung untuk dilayani. Dan meskipun akhirnya terjadi *packet drop* akan dilanjutkan ke fase *fast recovery* untuk mempertahankan *throughput* tetap tinggi. Hal tersebut membuat

throughput TCP New Reno lebih baik daripada TCP Vegas saat menggunakan antrian *Random Early Detection*.

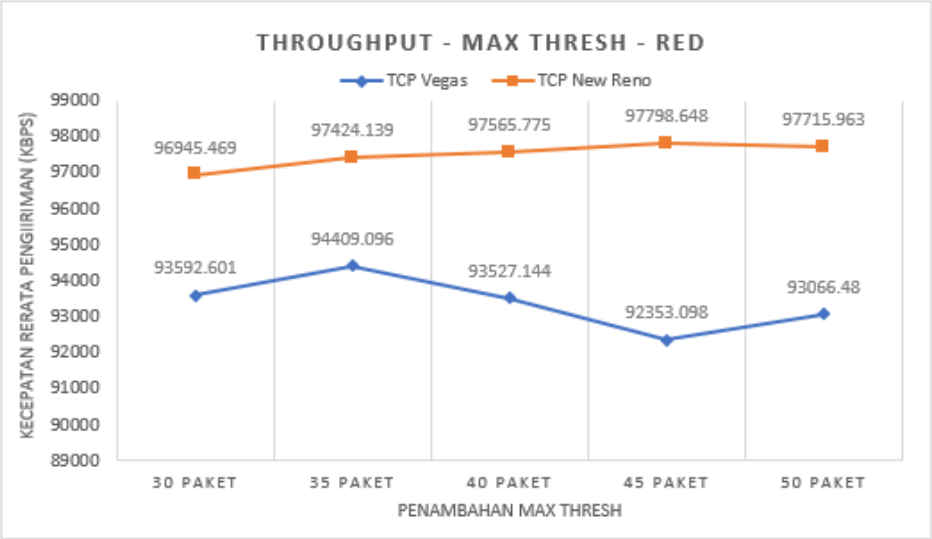


Gambar 5.7 Grafik Perbandingan *Throughput* Penambahan *Min Thresh*

Perbandingan *throughput* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan penambahan *min thresh* yaitu 20 paket dan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.8 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang fluktuatif dimulai dari nilai 93592.601 kbps dengan *max thresh* 30 paket, kemudian grafik naik ke nilai 94409.096 kbps ketika *max thresh* 35 paket, kemudian grafik turun ke angka 93627.144 kbps dengan *max thresh* 40 paket, kemudian grafik turun lagi ke nilai 92353.098 kbps ketika *max thresh* 45 paket, kemudian grafik naik ke nilai 93066.480 kbps ketika *max thresh* naik menjadi 50 paket.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 96945.469 kbps, 97424.139 kbps, 97565.775 kbps dan 97798.648 kbps dengan *max thresh* 30, 35, 40 dan 45 paket, kemudian grafik akan turun ke nilai 97715.963 kbps dengan *max thresh* yaitu 50 paket.
- Dari dua hasil tersebut, penambahan *max thresh* mempengaruhi peningkatan di sisi *throughput* pada TCP New Reno karena semakin besar *max thresh* semakin banyak pula paket data yang dilayani, sedangkan untuk TCP Vegas meskipun sempat mengalami peningkatan ketika *max thresh* 30 paket, namun setelahnya mengalami penurunan.
- Hasil tersebut dapat dilihat bahwa saat menggunakan antrian *Random Early Detection* TCP New Reno lebih baik dalam sisi *throughput* dibandingkan TCP Vegas. Karena TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data

secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar. Alasan naik turunnya *throughput* TCP Vegas ketika penambahan *max thresh* pada antrian *Random Early Detection* yaitu karena dengan penambahan *max thresh* maka data yang ditampung akan tetap dan data yang di *drop* secara *random* semakin bertambah serta semakin sedikit paket data yang di *drop* secara langsung, hal tersebut akan membuat nilai RTT menjadi besar, ketika nilai RTT besar TCP Vegas menganggap jaringan mengalami *congestion* sehingga akan terjadi penurunan pengiriman data yang menyebabkan kinerja *throughput* menurun seiring bertambahnya *max thresh* pada atrian *Random Early Detection.* Sedangkan TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop*, sehingga ketika terjadi *packet drop*, *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *throughput* TCP New Reno ketika menggunakan penambahan *max thresh* pada antrian *Random Early Detection* yaitu karena semakin bertambah *max thresh*, semakin bertambah pula data yang ditampung untuk dilayani dan di *drop* secara *random* serta semakin sedikit paket data yang di *drop* secara langsung. Dan meskipun akhirnya terjadi *packet drop* akan dilanjutkan ke fase *fast recovery* untuk mempertahankan *throughput* tetap tinggi. Hal tersebut membuat *throughput* TCP New Reno lebih baik daripada TCP Vegas saat menggunakan antrian *Random Early Detection*.



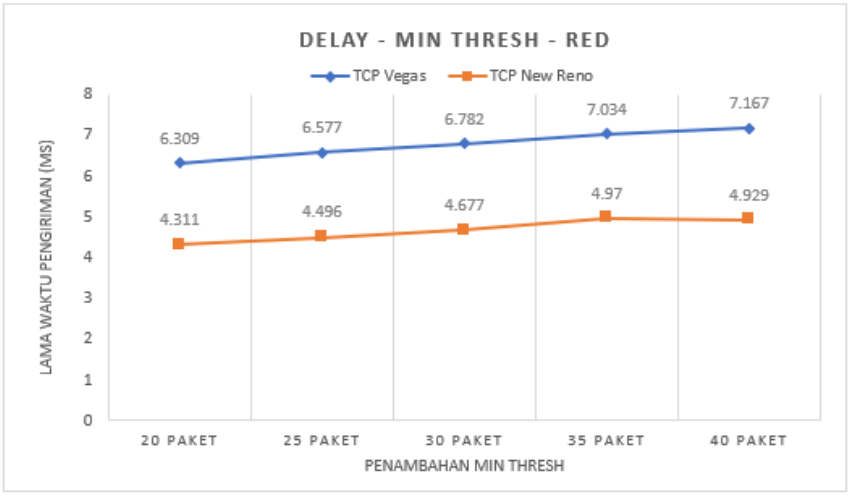
Gambar 5.8 Grafik Perbandingan *Throughput* Penambahan *Max Thresh*

5.2.2.3 Analisis Hasil *Delay*

Perbandingan *delay* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dan parameter *max thresh* yaitu 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.9 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung naik, dimulai dari nilai 4.311 ms, 4.496 ms, 4.677 ms dan 4.970 ms dengan *min thresh* 20 paket, 25 paket, 30 paket dan 35 paket. Terakhir grafik akan turun ke nilai 4.929 ms ketika *min thresh* 40 persen.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 6.309 ms, 6.577 ms, 6.782 ms, 7.034 ms dan 7.167 ms dengan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket.
- Dari dua hasil tersebut, penambahan *min thresh* mempengaruhi peningkatan di sisi *delay* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah *min thresh*, semakin besar pula paket data yang dapat di layani dan semakin lama pula waktu pelayanan untuk paket data.
- Meskipun saat menggunakan antrian *Random Early Detection* kedua TCP mengalami kenaikan, namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop*

secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop* sehingga ketika terjadi *packet drop*, *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman paket data akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *delay* TCP New Reno ketika menggunakan penambahan *min thresh* pada antrian *Random Early Detection* yaitu karena semakin bertambah *min thresh*, semakin bertambah pula data yang ditampung untuk dilayani dan hal tersebut menyebabkan waktu untuk melayani paket data menjadi bertambah lama. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* untuk menyesuaikan pengiriman paket data yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar dan menyesuaikan ukuran kapasitas *buffer*. Alasan meningkatnya *delay* TCP Vegas ketika penambahan *min thresh* pada antrian *Random Early Detection* yaitu karena dengan bertambahnya *min thresh*, paket data yang ditampung akan menjadi semakin banyak, hal tersebut membuat nilai RTT menjadi besar yang membuat TCP Vegas mengganggap jaringan mengalami *congestion,* ketika *congestion* terjadi pengiriman paket data akan menurun sehingga banyak paket data yang dapat di tampung tidak terlalu banyak dan hal tersebut membuat waktu tunggu paket untuk dilayani semakin kecil. Hal tersebut membuat *delay* TCP Vegas lebih rendah dari TCP New Reno saat menggunakan antrian *Random Early Detection*.

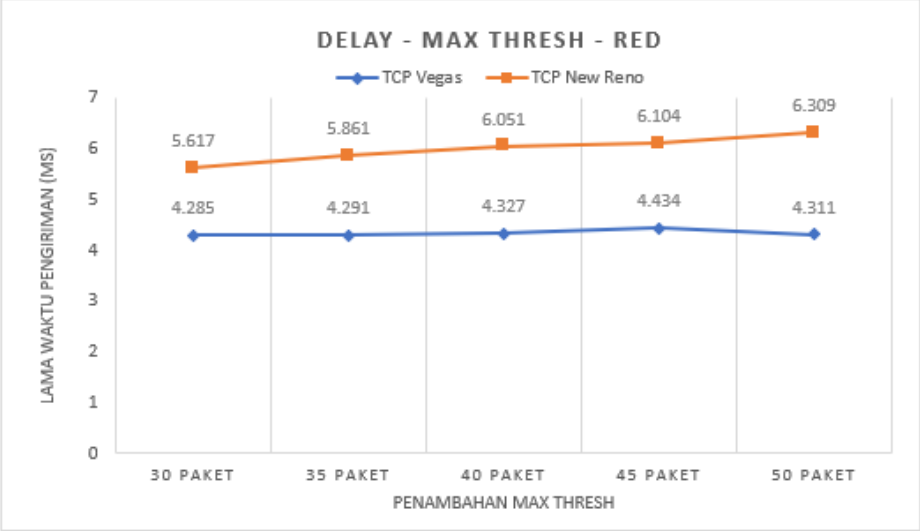


Gambar 5.9 Grafik Perbandingan *Delay* Penambahan *Min Thresh*

Perbandingan *delay* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan *min thresh* yaitu 20 paket dan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.10 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang cenderung naik, dimulai dari nilai 4.285 ms, 4.291 ms, 4.327 ms dan 4.343 ms dengan *max thresh* 30, 35, 40 dan 45 paket, terakhir grafik turun ke 4.311 ms ketika *max thresh* naik menjadi 50 paket.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung naik, dimulai dari nilai 5.617 ms, 5.861 ms, 6.051 ms, 6.104 ms dan 6.309 ms dengan *max thresh* 30, 35, 40, 45 dan 50 paket.
- Dari dua hasil tersebut, penambahan *max thresh* mempengaruhi peningkatan di sisi *delay* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah *max thresh*, semakin besar paket data yang akan ditandai atau di *drop* secara *random* dan kemungkinan paket *drop* bisa di minimalisir.
- Meskipun saat menggunakan antrian *Random Early Detection* kedua TCP mengalami kenaikan*,* namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop* sehingga ketika terjadi *packet drop*, *congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman paket data akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan meningkatnya *delay* TCP New Reno ketika menggunakan penambahan *max thresh* pada antrian *Random Early Detection* yaitu karena semakin bertambah *max thresh*, semakin bertambah pula data yang ditampung untuk dilayani dan di *drop* secara *random* serta semakin sedikit paket data yang di *drop* secara langsung hal tersebut menyebabkan waktu untuk melayani paket data menjadi bertambah lama. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT

menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* untuk menyesuaikan pengiriman paket data yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar. Alasan meningkatnya *delay* TCP Vegas ketika penambahan *max thresh* pada antrian *Random Early Detection* yaitu karena dengan bertambahnya *max thresh*, paket data yang ditampung akan tetap dan paket data yang ditampung untuk di *drop* secara *random* akan bertambah, hal tersebut membuat nilai RTT menjadi besar yang membuat TCP Vegas mengganggap jaringan mengalami *congestion,* ketika *congestion* terjadi pengiriman paket data akan menurun sehingga banyak paket data yang dapat di tampung tidak terlalu banyak dan hal tersebut membuat waktu tunggu paket untuk dilayani semakin kecil. Hal tersebut membuat *delay* TCP Vegas lebih rendah dari TCP New Reno saat menggunakan antrian *Random Early Detection*.



Gambar 5.10 Grafik Perbandingan *Delay* Penambahan *Max Thresh*

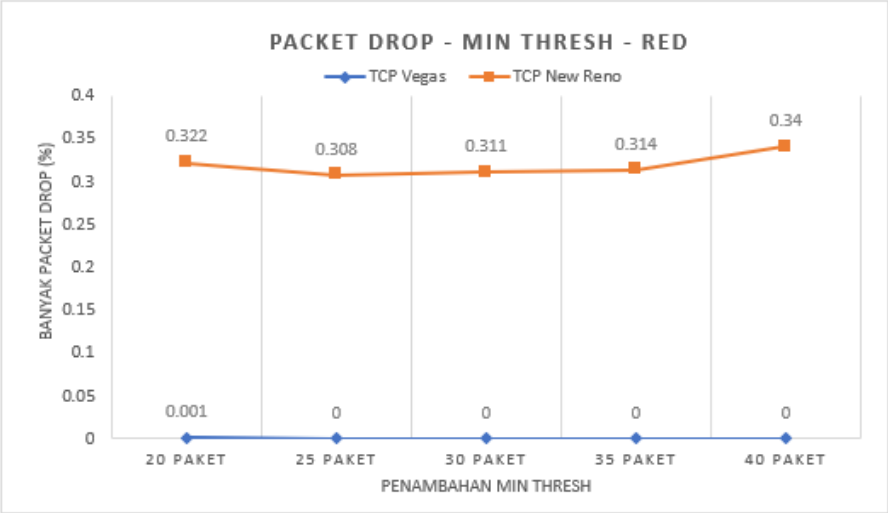
5.2.2.4 Analisis Hasil *Packet Drop*

Perbandingan *packet drop* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan penambahan *min thresh* yaitu 20, 25, 30, 35 dan 40 paket dan *max thresh* yaitu 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.11 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang konstan yang dimulai dari nilai 0.001 persen dengan *min thresh* 20 paket, kemudian grafik turun ke nilai 0 persen dengan *min thresh* 25, 30, 35 dan 40 paket.

- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* grafik yang cenderung naik, dimulai dari nilai 0.322 persen dengan *min thresh* 20 paket, kemudian grafik turun ke nilai 0.308 persen ketika *min thresh* 25 paket, selanjutnya grafik akan naik ke nilai 0.311 persen, 0.314 persen dan 0.340 persen dengan *min thresh* yaitu 30, 35 dan 40 paket.
- Dari dua hasil tersebut, penambahan *min thresh* mempengaruhi penurunan di sisi *packet drop* pada TCP Vegas maupun TCP New Reno. Hal tersebut karena semakin bertambah *min thresh*, semakin besar paket data yang dapat di layani dan semakin sedikit pula *packet* yang di *drop.*
- Meskipun saat menggunakan antrian *Random Early Detection* kedua TCP mengalami penurunan*,* namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop* sehingga ketika terjadi *packet drop, congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman paket data akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan turunnya *packet drop* TCP New Reno ketika menggunakan penambahan *min thresh* pada antrian *Random Early Detection* yaitu dengan *min thresh* yang kecil, ketika paket data yang datang lebih banyak dari pada batas *min thresh* mengakibatkan *packet drop* akan sering terjadi*.* Dengan penambahan *min thresh*, paket data yang dapat dilayani akan bertambah dan *packet drop* akan menjadi turun*.* Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* untuk menyesuaikan pengiriman paket data yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar dan menyesuaikan ukuran *min thresh*. Alasan turunnya *packet drop* TCP Vegas ketika penambahan *min thresh* pada antrian *Random Early Detection* yaitu dengan bertambahnya *min thresh,* paket data yang ditampung semakin banyak menyebabkan RTT menjadi besar dan TCP Vegas menganggap jaringan terjadi *congestion*, sehingga paket data yang dikirim akan dikurangi dan menyebabkan data yang dapat ditampung *min*

thresh tidak terlalu banyak dan hal tersebut membuat *packet drop* menjadi semakin berkurang. Hal tersebut membuat *packet drop* TCP Vegas lebih rendah daripada TCP New Reno saat menggunakan antrian *Random Early Detection*.

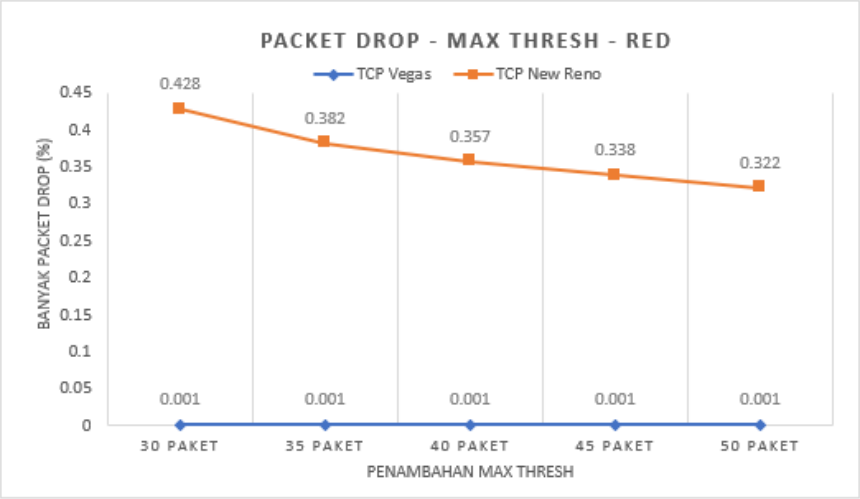


Gambar 5.11 Grafik Perbandingan *Packet Drop* Penambahan *Min Thresh*

Perbandingan *packet drop* antara TCP Vegas dan TCP New Reno menggunakan antrian *Random Early Detection* dengan kapasitas *buffer* yaitu 60 paket dengan *min thresh* yaitu 20 paket dan penambahan *max thresh* yaitu 30, 35, 40, 45 dan 50 paket ditampilkan dengan grafik yang dapat dilihat pada Gambar 5.12 dengan penjelasan sebagai berikut:

- TCP Vegas yang ditunjukan pada grafik dengan simbol garis berwarna biru menunjukkan grafik yang konstan dengan nilai 0.001 persen dengan *max thresh* 30, 35, 40, 45 dan 50 paket.
- TCP New Reno yang ditunjukkan pada grafik dengan simbol garis berwarna *orange* menunjukkan grafik yang cenderung turun, dimulai dari nilai 0.428 persen ketika *max thresh 30* paket, kemudian grafik turun ke nilai 0.382 persen, 0.357 persen ketika *max thresh* 35 dan 40 paket, kemudian grafik naik ke nilai 0.388 persen ketika *max thresh* 45 paket, kemudian grafik akan turun lagi ke angka 0.322 persen dengan *max thresh* yaitu 50 paket.
- Dari dua hasil tersebut, penambahan *max thresh* mempengaruhi penurunan di sisi *packet drop* pada TCP New Reno, namun pada TCP Vegas *packet drop* berjalan konstan pada angka yang tetap. Hal tersebut karena semakin bertambah *max thresh*, semakin besar paket data yang akan ditandai atau di *drop* secara *random* dan kemungkinan paket *drop* bisa di minimalisir.
- Meskipun saat menggunakan antrian *Random Early Detection* kedua TCP mengalami penurunan, namun TCP Vegas lebih unggul dari TCP New Reno. Karena TCP New Reno dalam mengirimkan paket data tidak memperhatikan kondisi jaringan dengan mengirimkan paket data secara cepat sehingga *congestion window* meningkat secara eksponensial. Namun

ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme antrian *Random Early Detection* akan menampung paket data yang datang hingga *min thresh* penuh. Jika *min thresh* penuh, paket data yang datang akan di *drop* secara *random* dan jika paket data yang datang melebihi batas *max thresh* maka akan langsung di *drop* sehingga ketika terjadi *packet drop, congestion window* pada TCP New Reno akan berkurang setengah dan kecepatan pengiriman paket data akan kembali dari awal sehingga kepadatan antrian akan berkurang. Alasan turunnya *packet drop* TCP New Reno ketika menggunakan penambahan *max thresh* pada antrian *Random Early Detection* yaitu dengan *max thresh* yang kecil, ketika paket data yang datang lebih banyak dari pada batas *max thresh* mengakibatkan *packet drop* secara *ranndom* akan sering terjadi. Dengan penambahan *max thresh*, paket data yang dapat dilayani dan di *drop* secara *random* akan bertambah dan *packet drop* akan menjadi turun. Sedangkan TCP Vegas dalam mengirimkan paket data akan melihat kondisi jaringan dengan menghitung varian RTT dari paket yang dikirimkan. Namun ketika semua sumber mengirimkan paket data secara bersamaan akan terjadi lonjakan *traffic* data, dimana mekanisme manajemen antrian *Random Early Detection* akan menampung paket yang datang hingga *min thresh* penuh. Jika *min thresh* penuh menyebabkan RTT menjadi besar. Jika RTT besar, TCP Vegas menganggap jaringan mengalami *congestion,* sehingga akan mengurangi *congestion window* untuk menyesuaikan pengiriman paket data yang berakibat turunnya pengiriman paket data untuk menghindari *packet drop*. Setelah itu TCP Vegas akan menstabilkan ukuran *congestion window* supaya paket data yang dikirim ke tujuan tidak lagi membanjiri antrian *Random Early Detection* sehingga ukuran RTT tidak lagi besar. Alasan turunnya *packet drop* TCP Vegas ketika penambahan *max thresh* pada antrian *Random Early Detection* yaitu dengan bertambahnya *max thresh,* paket data yang ditampung dan di *drop* secara *random* semakin banyak menyebabkan RTT menjadi besar dan TCP Vegas menganggap jaringan terjadi *congestion*, sehingga paket data yang dikirim akan dikurangi dan menyebabkan data yang dapat ditampung *max thresh* tidak terlalu banyak dan hal tersebut membuat *packet drop* menjadi sangat berkurang. Hal tersebut membuat *packet drop* TCP Vegas lebih rendah daripada TCP New Reno saat menggunakan antrian *Random Early Detection* .



Gambar 5.12 Grafik Perbandingan *Packet Drop* Penambahan *Max Thresh*