BAB II
DASAR TEORI

2.1. KAJIAN PUSTAKA

Pada tahun 2021, Heru Setiawan, Denar Regata Akbi, dan Diah Risqiwati melakukan penelitian berjudul "Analisis Kinerja di TCP *New* Reno Dan TCP Cubic Terhadap Fase *Slow Start* pada Jaringan *Mobile Ad Hoc* Menggunakan Protokol AODV". Tujuan dari penelitian ini adalah untuk memantau bagaimana kinerja TCP Reno dan TCP Cubic ketika diimplementasikan dalam topologi MANET dengan menggunakan *routing* protokol AODV. Pengujian dilakukan dengan memvariasikan jumlah *node* dan ukuran *buffer* paket menggunakan simulator *Network Simulator*-3. Parameter yang diuji meliputi *congestion window*, sshtresh, *delay*, dan *throughput*. Hasil penelitian menunjukkan bahwa kinerja kedua varian TCP selama fase *slow start* memiliki perbedaan. TCP *New* Reno menunjukkan stabilitas yang lebih konsisten dalam setiap pengujian dengan rata-rata ssthreshold sebesar 20 segmen. Dengan demikian, nilai *throughput* dan *delay* pada TCP *New* Reno juga relatif lebih baik dibandingkan dengan varian TCP Cubic yang digunakan sebagai pembanding. [1].

Pada tahun 2020, Salman, Gusti Amri Ginting, dan Dyah Reni Wahyuningrum melakukan penelitian berjudul "Analisis Unjuk Kerja TCP *Window Size* 64k menggunakan Algoritma TCP New Reno pada Jaringan *Wired* dan *Wireless*". Tujuan dari penelitian ini adalah untuk membandingkan performa layanan FTP pada jaringan *wired* dan nirkabel dengan menggunakan TCP *Window size* sebesar 64KB dan algoritma TCP *New* Reno. Simulasi dilakukan menggunakan simulator Riverbed *Modeler* versi 17.5. Pengujian difokuskan pada perbandingan *Quality of Service* (QoS), termasuk *throughput, delay, jitter*, dan *packet loss* pada beberapa skenario yang berbeda. Skenario yang diuji mencakup jaringan dengan 4, 8, 16, dan 32 klien, baik pada jaringan wired maupun nirkabel 802.11n dengan implementasi algoritma TCP *New* Reno. Hasil penelitian menunjukkan bahwa pada jaringan *wireless* 802.11n dengan 32 *client* dan transfer data sebesar 75% dari total *bandwidth*, skenario tersebut menunjukkan performa terbaik pada parameter *throughput*. Pada jaringan dengan 4 *client* dan transfer data sebesar 25% dari total

bandwidth, skenario tersebut menunjukkan performa terbaik pada parameter *delay*. Sedangkan pada jaringan dengan 4 *client* dan transfer data sebesar 75% dari total *bandwidth*, skenario tersebut menunjukkan performa terbaik pada parameter *jitter*. Pada jaringan *Wired* Pengiriman datanya lebih lambat, dimana nilai *avarage jitter* terbaik dengan $0{,}3 \times 10^{-9}$ ms. Pada jaringan *Wireless* lebih sering terjadi paket hilang dimana nilai *avarage throughput* yaitu 13,0506 kb/sec, dan nilai *delay* yaitu 0,253 ms [4].

Pada tahun 2016, Medi Taruk dan Hario Jati Setyadi melakukan penelitian yang berjudul "Analisis Mekanisme Penanganan Kemacetan (*Congestion Control*) pada Algoritma Varian Protokol TCP". Penelitian ini berfokus pada modifikasi pada fase *slow-start* dan *congestion avoidance* untuk mengatasi kemacetan dalam jaringan. Dua perbaikan yang dipelajari adalah *larger initial window* dan *smooth start*, yang kemudian dibandingkan dengan algoritma *smooth congestion avoidance* dalam fase *congestion avoidance* untuk mengurangi kehilangan paket. Penelitian ini menggunakan aplikasi simulasi dengan menggunakan FTP (*File Transport Protocol*) sebagai input data dan generator trafik CBR (*Constant Bit Rate*) untuk menghasilkan lalu lintas jaringan. Dengan dua sumber input ini, yaitu FTP dan CBR, penelitian ini dapat menguji kinerja berbagai varian protokol TCP, termasuk TCP Reno, TCP Tahoe, TCP Vegas, dan TCP SACK. Hasil penelitian menunjukkan bahwa ketika hanya satu varian TCP yang bekerja pada satu waktu, baik itu TCP Reno, TCP Tahoe, TCP Vegas, atau TCP SACK, besar *throughput* yang dihasilkan hampir sama dan tidak ada perbedaan yang signifikan. Ketika mengimplementasikan varian-varian TCP tersebut dalam kondisi QoS yang setara (*fairness*), hasil menunjukkan bahwa semua varian TCP yang diuji memiliki throughput yang tidak berbeda secara signifikan, Walaupun terdapat sedikit perbedaan dalam jumlah data yang diterima oleh setiap *node* yang mewakili varian TCP yang digunakan, varian TCP Vegas dan TCP SACK menunjukkan *throughput* rata-rata yang cukup tinggi, sementara TCP Reno memiliki *throughput* rata-rata yang paling rendah, dan TCP Tahoe berada di antara keduanya, terutama ketika terjadi kehilangan paket karena kelebihan kapasitas jaringan. [3].

Pada tahun 2018, Aria T. H., Sabriansyah R. A., dan Adhitya B. melakukan penelitian berjudul "Analisa Kinerja Algoritma TCP *Congestion Control* Cubic,

Reno, Vegas dan Wesrtwood+". Penelitian ini bertujuan untuk membandingkan kinerja empat varian TCP, yaitu TCP Cubic, Reno, Vegas, dan Westwood+. Pengujian dilaksanakan dengan menganalisis tiga parameter, yaitu cwnd (*congestion window size*), ssthreshold (*slow start threshold*), dan *throughput* dari empat jenis algoritma TCP *Congestion Control* tersebut. Penelitian ini menggunakan dua perangkat, yaitu *sender* dan *receiver*, yang terhubung menggunakan kabel ethernet dengan koneksi ber*bandwidth* 100Mbit/s. Data TCP dikumpulkan pada *sender* menggunakan algoritma TCP Probe dan hasil yang di dapat ditampilkan dalam bentuk grafik menggunakan Gnuplot. Hasil penelitian menyatakan bahwa TCP Westwood+ merupakan varian TCP yang terbaik di antara ketiga TCP lainnya dalam semua lingkungan *Service Level Agreement* (SLA) service provider dengan metode *Bandwidth Estimation* yang digunakan. Dalam pengujian ini, TCP Westwood+ menunjukkan performa yang lebih baik dalam mengatur *congestion window size*, *slow start threshold*, dan *throughput* dibandingkan dengan TCP Cubic, Reno, dan Vegas. [9].

Dapat dilihat pada Tabel 2.1 merupakan kajian Pustaka dari penelitian ini yang menjadi acuan.

Tabel 2.1 Kajian Pustaka

Judul	Topik Penelitian	Metode	Hasil
(H. Setiawan, D. R. Akbi, D. Risqiwati 2021) Analisis Kinerja Di TCP *New* Reno Dan TCP Cubic Terhadap Fase *Slow Start* Pada Jaringan Mobile Ad Hoc Menggunakan Protokol AODV [1]	Membandingan kinerja TCP *New* Reno dan TCP Cubic terhadap *fase slow start*, dan diuji dengan menambahkan *node* dan mengubah ukuran buffer paket	Menggunakan simulator NS-3 dengan topologi jaringan *MANET*. Parameter yang dianalisis yaitu CWND, ssthresd, *throughput* dan *delay*	TCP *New* Reno menunjukkan stabilitas yang relatif baik dalam setiap pengujian dengan nilai rata-rata ssthreshold sebesar 20 segmen, yang mengakibatkan peningkatan nilai *throughput* dan *delay* yang lebih baik.

Judul	Topik Penelitian	Metode	Hasil
(Salman, J. G. A. Ginting, R. D. Wahyuningrum 2020) Analisis Unjuk Kerja TCP *Window Size* 64k menggunakan Algoritma TCP *New* Reno pada Jaringan Wired dan *Wireless* [4]	Penenlitian ini membandingkan performa jaringan *wired* dan *wireless* pada layanan FTP menggunakan TCP Window size 64kb.	Penelitian ini menggunakan simulator Riverbed *Modeler* versi 17.5. Algoritma yang digunakan yang TCP *New* Reno, parameter yang di analisis adalah QoS.	jaringan *Wired* Pengiriman datanya lebih lambat, pada jaringan *Wireless* lebih sering terjadi paket hilang dimana nilai *avarage throughput* yaitu 13,0506 kb/sec, dan nilai *delay* yaitu 0,253 ms
(M. R. G. Asgar 2018) Analisis Perbandingan Unjuk Kerja TCP *New* Reno dan Westwood+ untuk mereduksi kongesti pada jaringan WLAN [10]	Penelitian ini membandingkan kinerja dari algoritma TCP *New* Reno dengan TCP Westwood+ delam mereduksi kongesti pada jaringan *Wireless Local Area* (WLAN).	Penelitian ini menggunakan *Network Simulator* 3, penulis menggunakan algoritma TCP *New* Reno pada jaringan WLAN.	Hasil kinerja *Throughput* dan *Packet delivery ratio* TCP *New* Reno lebih baik dibanding TCP Westwood+.
(Ria T. H., Sabriansyah R. A., Adhitya B. 2018) Analisa Kinerja Algoritma TCP *Congestion Control* Cubic, Reno, Vegas dan Wesrtwood+ [9]	Penelitian ini membandingkan kinerja empat varian TCP *congestion control*, yaitu TCP Cubic, Reno, Vegas dan Westwood+.	Pengujian yang dilakukan adalah menganalisa congestion window, ssthreshold dan *throughput* empat jenis algoritma TCP *Congestion Control*.	Hasil penelitian ini menyimpulkan bahwa TCP Westwood+ memiliki performa yang lebih baik daripada ketiga varian TCP lainnya dalam semua kondisi *Service Level Agreement* (SLA).

2.2. DASAR TEORI

Dasar teori adalah landasan yang diajdikan acuan dalam penulisan laporan penelitian ini. Secara garis besar dasar teori meliputi jaringan *Wireless*, Protokol TCP, Prinsip kerja TCP, *Congestion Control*, TCP *New* Reno, TCP Tahoe, *Software* Simulasi Riverbed, Parameter *Congestion Control* dan kajian studi terdahulu yang dijadikan sebagai acuan dan penetapan analisis dalam penelitian yang akan dilakukan.

2.2.1. Jaringan Nirkabel (*Wireless*)

Jaringan nirkabel, atau sering disebut juga jaringan *wireless*, merujuk pada teknologi yang menghubungkan dua atau lebih perangkat secara tanpa kabel dan memungkinkan pertukaran data antar perangkat. Dalam penggunaannya, jaringan nirkabel tidak membutuhkan pengelolaan kabel yang kompleks dan jaringan kabel. Sebagai pengganti kabel, teknologi ini memanfaatkan hubungan elektromagnetik. Selain digunakan untuk komunikasi, jaringan nirkabel juga dikenal dengan istilah komunikasi tanpa kabel (*Wireless Communication*) yang memungkinkan transfer informasi secara jarak jauh, contohnya telepon seluler, jaringan nirkabel, dan satelit. [10].

2.2.2. *Wireless* LAN (WLAN)

Wireless LAN atau WLAN adalah kumpulan komputer yang terhubung satu sama lain membentuk jaringan komputer, namun jalur lalu lintas datanya menggunakan gelombang elektromagnetik sebagai media transmisi. Perbedaan utama antara *wireless* dan LAN adalah pada jenis media transmisi yang digunakan. *Wireless* menggunakan gelombang elektromagnetik sebagai media transmisi, sementara LAN menggunakan kabel sebagai media transmisi [4].

Dengan media transmisi gelombang elektromagnetik, makan WLAN memberikan kemudahan bagi *user* dalan dalam implementasinya, antara lain:

1. Instalasi dan konfigurasi tidak perlu menggunakan kabel atau konektor, sehingga memudahkan prosesnya.
2. Dengan menggunakan teknologi ini, pengguna dapa terhubung ke internet dari mana saja yang memiliki sinyal WLAN.

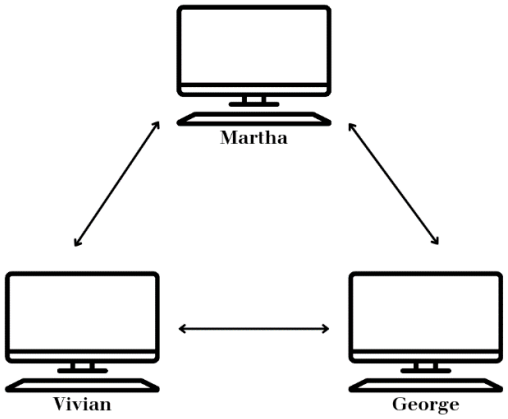
3. WLAN memungkinkan pengguna untuk terhubung ke internet dengan perangkat mobile seperti laptop, tablet, dan *smartphone* sehingga dapat terhubung darimana saja.
4. WLAN dapat menyediakan kecepatan akses yang tinggi ke internet, tergantung pada standar yang digunakan dan kualitas sinyal [10].

Menurut standar IEEE teknologi WLAN memiliki tiga model topologi utama, yaitu:

1. *Independent Basic Service Set* (IBSS)

   Independent Basic Service Set (IBSS) adalah jenis topologi yang biasanya dikenal dengan *ad-hoc network* yang dimana merupakan sebuah set perangkat yang dapat terhubung ke jaringan WLAN tanpa menggunakan satu *Access Point* (AP). *Ad-hoc network* memungkinkan pengguna untuk terhubung ke internet atau jaringan lokal tanpa bergantung pada koneksi kabel. *Ad-hoc network* juga dapat terdiri dari satu atau lebih *Virtual Local Area Network* (VLAN) yang memungkinkan pembagian akses jaringan menurut kebutuhan.

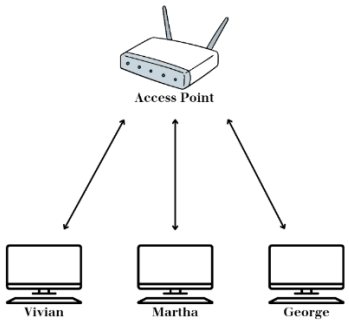
   Jaringan *ad hoc* banyak digunakan karena tidak memerlukan penggunaan kabel dalam proses transmisi data. Selain itu, biaya yang diperlukan relatif rendah karena hanya memerlukan perangkat *wireless* pada laptop. Jaringan *ad hoc* juga mudah digunakan untuk berbagi file dan melakukan konfigurasi. Namun, terdapat batasan dalam jumlah jaringan yang dapat terhubung, dan proses perpindahan data memerlukan waktu yang cukup lama. [11]. Pada Gambar 2.1 merupakan topologi WLAN model *Independent Basic Service Set* (IBSS).



Gambar 2.1 Topologi WLAN Model IBSS.

2. *Basic Service Set* (BSS)

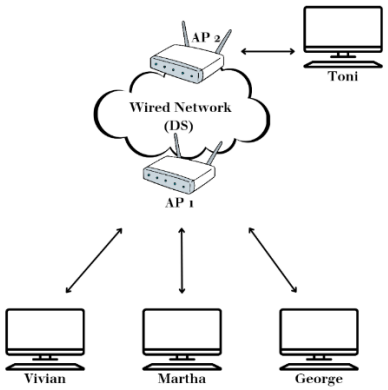
Basic Service Set (BSS) merupakan sebuat set perangkat yang terhubung ke jaringan WLAN melalui satu *Access Point* (AP). Setiap klien yang ingin terhubung dengan klien lainnya harus terhubung dengan *access point* terlebih dahulu [11]. Pada Gambar 2.2 merupakan topologi WLAN model *Basic Service Set* (BSS).



Gambar 2.2 Topologi WLAN Model BSS.

3. *Extended Service Set* (ESS)

Extended Service Set (ESS) adalah sekumpulan perangkat yang terhubung ke jaringan WLAN melalui beberapa *Access Point* (AP). Secara sederhana, jaringan ESS terdiri dari beberapa jaringan BSS dengan tujuan untuk memberikan cakupan area yang lebih luas. Dalam topologi ini, jaringan BSS dan ESS dapat dikombinasikan dengan jaringan kabel. Jenis koneksi ini dikenal sebagai infrastruktur, di mana perangkat *wireless client* dapat terhubung dan berkomunikasi dengan klien lain pada jaringan kabel [11]. Pada Gambar 2.3 merupakan topologi WLAN model *Extended Service Set* (ESS).



Gambar 2.3 Topologi WLAN Model ESS [11]

2.2.3. Spesifikasi Standar *Wireless* LAN (WLAN)

IEEE 802.11 memiliki banyak standar, tetapi yang paling sering digunakan pada titik akses WLAN adalah IEEE 802.11a, 802.11b, dan 802.11g, juga dikenal sebagai 802.11a/b/g. Standar 802.11b beroperasi pada frekuensi 2,4 GHz dan memiliki kecepatan transfer data hingga 11 Mbps. Standar 802.11a beroperasi pada frekuensi 5 GHz dan memiliki kecepatan transfer data hingga 54 Mbps. Selain itu, 802.11g juga memiliki data rate 54 Mbps namun beroperasi pada frekuensi 2,4 GHz. Pertumbuhan komunikasi nirkabel saat ini didorong oleh ketersediaan perangkat berbasis radio umum, seperti *walkie talkie*, telepon tanpa kabel, telepon seluler, dan radio lainnya. Kebutuhan akan periferal komputer yang mudah dipasang dan dapat terhubung ke jaringan yang telah menyetujui pengembangan teknologi nirkabel untuk jaringan komputer. *Wireless Area Network* (WAN) memiliki dua jenis metode, yaitu *ad-hoc* dan infrastruktur. Meskipun jaringan LAN nirkabel menyerupai jaringan kabel, perbedaannya dapat dilihat pada penggunaan kanal frekuensi yang serupa dan SSID (*Service Set Identifier*) yang digunakan untuk membedakan dan meningkatkan jaringan nirkabel.[12].

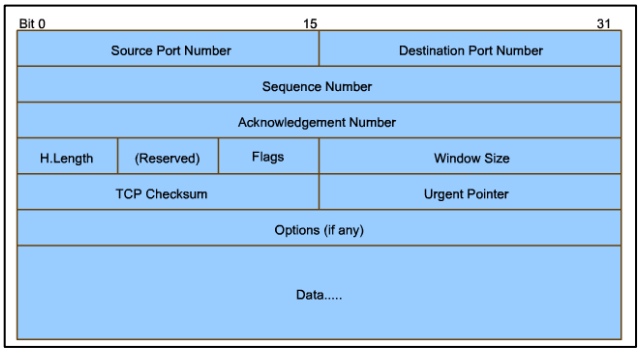
Satu-satunya standar saat ini untuk IEEE 802.11, yang sebelumnya mencakup 802.11a, 802.11b, dan 802.11g, adalah IEEE 802.11n. Setiap standar memiliki sistem keamanan, kecepatan, dan teknologi modulasi yang berbeda. Standar 802.11n diperkenalkan pada tahun 2009 dan secara teoritis memiliki kecepatan transfer data hingga 600 Mbps, meskipun menurut pengujian WiFi *Alliance*, kecepatannya hanya mencapai 450 Mbps. Standar saat ini memungkinkan pengiriman konten dengan baik melalui layanan yang membutuhkan *bandwidth* dalam jumlah besar, seperti *video streaming High Definition* (HD) dan *Voice over Internet Protocol* (VoIP).

Ada beberapa keunggulan yang ditawarkan oleh standar 802.11n. Pertama, penggunaan *dual frequency*, yang memungkinkan penggunaan frekuensi 2,4 GHz atau 5 GHz. Ini mendukung perangkat yang masih menggunakan standar sebelumnya dengan frekuensi 2,4 GHz. Keunggulan kedua adalah teknologi modulasi MIMO (*Multiple Input Multiple Output*), yang menggunakan dua atau lebih antena untuk menangkap sinyal radio selama *transfer* data, meningkatkan kualitas sinyal dan *throughput* data dengan memecah *stream* data secara spatial.

Standar 802.11n memiliki kecepatan maksimum dan jangkauan sinyal yang lebih baik, serta lebih tahan terhadap gangguan seperti sinyal interferensi. Namun, kelemahannya termasuk biaya yang lebih tinggi dibandingkan standar sebelumnya. Penggunaan lebih dari satu sinyal dapat mudah terganggu jika berdekatan dengan jaringan berbasis 802.11b/g [13].

2.2.4. *Transmission Control Protocol* (TCP)

Transmission Control Protocol (TCP) adalah standar komunikasi data yang digunakan untuk pertukaran data antara komputer. Protokol ini beroperasi pada layer keempat OSI, yaitu lapisan transport layer. TCP adalah protokol jaringan yang terbuka dan bebas terhadap mekanisme *transport* pada jaringan fisik, sehingga dapat digunakan di berbagai lingkungan. TCP menyediakan pengiriman paket data yang andal, sehingga paket yang diterima oleh penerima TCP tidak diubah, tidak digandakan, dan dibuang dengan cara yang benar. Algoritma kontrol TCP merupakan faktor penting dalam produktivitas dan volume data yang dikirimkan melalui jaringan. Pertimbangkan aplikasi yang menggunakan protokol TCP, seperti HTTP, FTP, Email, dan lainnya. Protokol TCP memastikan bahwa data yang dibagi antara aplikasi pada komputer yang berbeda dapat digabungkan dan dibuka dengan aman, memungkinkan transmisi data yang efisien dan aman melalui jaringan [14]. Pada Gambar 2.4 merupakan anatomi sebuah segmen yang terdapat pada protokol TCP.



Gambar 2.4 *Segment* pada TCP [15]

Source port dan *destination port* Tujuan mencakup nomor *port* dari aplikasi pengirim dan penerima. Nomor urut berfungsi untuk mendeteksi data yang hilang dan mengembalikan data yang hilang atau kedaluwarsa. *Offset* adalah bilangan bulat 32-bit di *header* TCP, sedangkan nomor Pengakuan mendefinisikan soket

TCP yang diantisipasi untuk digunakan nanti. *Offset* ini mengidentifikasi titik di mana data dibagi. *Reserved* memiliki ukuran bit 6 bit, dan *Code* bit memiliki fungsi kontrol yang digunakan untuk memulai dan mengakhiri sesi. *Window* berisi ukuran *window* dari pengirim yang akan dikirimkan dalam format oktet. *Checksum* menggunakan CRC (*Cyclic Redundancy Check*) karena TCP tidak memberlakukan lapisan di bawah *checksum*. CRC memeriksa header file dan data untuk memastikan integritasnya. Jika Penunjuk Mendesak disertakan dalam kode bit-set, ia memiliki kemampuan untuk menganggap bidang yang ditentukan. Opsi memiliki panjang data 0 atau nilai 32-bit. Namun, jika ada opsi yang mencegah *field option* menjadi nilai 32-bit, bit 0 harus disetel ke nol untuk menampilkan data dengan ukuran 32-bit yang lebih besar dari rata-rata. Data adalah data aktual yang diambil dari formulir pengajuan aplikasi. TCP memiliki berbagai bidang *header* yang digunakannya untuk mengatur komunikasi dan transfer data antara aplikasi yang mengirim dan menerima pesan di seluruh jaringan. [14]. Beberapa varian TCP (*Transmission Control Protocol*) *Congestion Control* yang dapat ditemukan dalam TCP yaitu:

1. TCP Tahoe: TCP Tahoe adalah varian pertama dari TCP yang memiliki algoritma kontrol kemacetan bawaan. Varian ini menggunakan *Slow Start*, AIMD (*Additive Increase Multiplicative Decrease*), dan *Fast* Retransmit untuk menghindari kemacetan. Ketika TCP Tahoe menerima tiga ACK duplikat, varian ini akan memicu *Fast* Retransmit untuk mengirim ulang paket yang hilang
2. TCP Reno: TCP Reno adalah varian TCP yang merupakan pengembangan dari TCP Tahoe. TCP Reno memiliki beberapa peningkatan pada algoritma kontrol kemacetan, seperti *fast* Retransmit dan *fast Recovery*, yang memungkinkan TCP Reno untuk merespons kemacetan jaringan dengan lebih cepat dan efisien. TCP Reno juga lebih cepat daripada TCP Tahoe di jaringan yang cepat
3. TCP Vegas: TCP Vegas adalah varian TCP yang menggunakan pendekatan yang berbeda dalam mengatasi kemacetan. Varian ini menggunakan pendekatan yang lebih proaktif daripada TCP Tahoe dan TCP Reno. TCP Vegas menggunakan RTT (*Round Trip Time*) untuk menghitung kecepatan

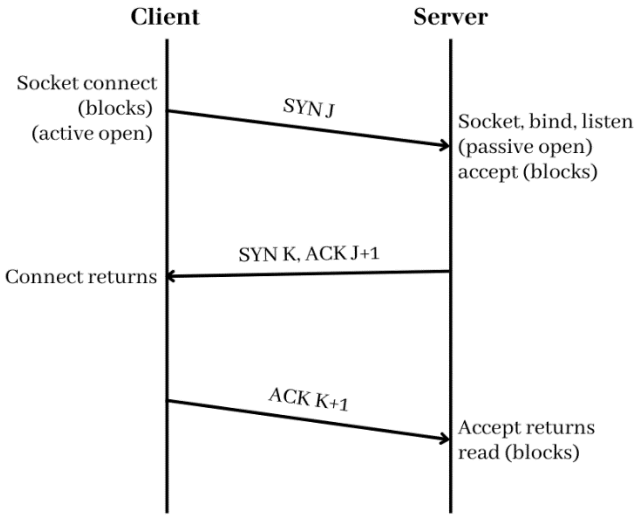
jaringan dan menghindari kemacetan dengan mengurangi laju pengiriman data sebelum kemacetan terjadi

4. TCP New Reno: TCP New Reno adalah pengembangan dari TCP Reno yang memperbaiki beberapa masalah pada TCP Reno. Varian ini memungkinkan TCP untuk mengirimkan beberapa paket setelah menerima tiga ACK duplikat, sehingga dapat meningkatkan *throughput* jaringan
5. TCP Cubic: TCP Cubic adalah varian TCP yang dirancang untuk meningkatkan throughput jaringan pada jaringan yang sangat besar dan sangat lambat. Varian ini menggunakan pendekatan yang berbeda dalam menghitung laju pengiriman data dan menghindari kemacetan
6. TCP Westwood+: TCP Westwood+ adalah varian TCP yang dirancang untuk meningkatkan *throughput* pada jaringan nirkabel. Varian ini menggunakan pendekatan yang adaptif dalam menghitung laju pengiriman data dan menghindari kemacetan [16].

2.2.4.1. Prinsip Kerja TCP

Saat melakukan tugas, *protocol* TCP memiliki beberapa prinsip kerja yaitu:

1. *Connection Oriented*



Gambar 2.5 Komunikasi *Client-Server (Three Way Handshake)*

Gambar 2.5 merupakan komunikasi *client-server* (*Three Way Handshake*) yang dimana Sebelum data ditransmisikan, TCP akan membuat koneksi antara *host* yang mengirim data dan *host* yang menerimanya untuk memastikan bahwa data dikirim dengan benar. Ini memungkinkan setiap *host*

untuk menangani transmisi data dengan benar. Data dapat diambil setelah koneksi gagal. Saat proses pengiriman data sedang berjalan, TCP juga melakukan perawatan pada koneksi tersebut dengan mengirimkan ACK ke setiap segmen yang masih ada pada *host* target. ACK ini berfungsi sebagai konfirmasi bahwa target *host* telah menulis segmen tersebut. Jika ada *acknowledgment* yang tidak sampai ke *host* target, TCP dapat melakukan transmisi ulang dengan mengirimkan *acknowledgment* yang tidak sampai ke *host* target. Segera setelah transmisi data selesai, TCP akan menutup koneksi. Hal ini dilakukan agar pengirim *host* dan *host* memahami bahwa proses transfer data telah selesai dan koneksi dapat dimatikan dengan aman.[17].

2. *Reliable Transmission*

   Data yang dikirimkan melalui TCP diberikan nomor urut unik pada setiap *byte* data untuk memastikan bahwa data bisa disusun kembali dengan benar setelah diterima. Hal ini penting karena data bisa mengalami perpecahan, kehilangan, atau tiba di perangkat tujuan dalam urutan yang tidak sesuai selama proses transmisi. Ketika data diterima, paket data yang palsu akan diabaikan, dan jika paket datang dalam urutan yang tidak tepat, maka akan di urutkan ulang agar bisa disusun kembali dengan benar.

3. *Error Detection*

   Apabila terjadi *error*, pada saat ada paket yg hilang disaat proses transmisi berlangsung, maka paket dapat dikirimkan kembali sesuai dengan paket yang hilang. Agar menjamin semua intergasi setiap segmen TCP, TCP menerapkan perhitungan TCP *Checksum*.

4. *Flow Control*

   Flow Control berfungsi untuk mengontrol kecepatan pengiriman data antara dua *node*, sehingga mencegah pengiriman data yang berlebihan kepada penerima yang lebih lambat. Ini sangat penting ketika dua perangkat memiliki kecepatan komunikasi yang berbeda. Misalnya, kemampuan PC dan *smartphone* biasanya berbeda saat data ditransfer di antara keduanya. *Smartphone* mungkin dapat mentransfer data dari PC lebih cepat. Oleh karena itu, TCP akan mengubah alikuot data sehingga ponsel cerdas tidak terus-

menerus berkonflik atau macet dengan data yang masuk, memungkinkan transmisi yang lebih lama.

5. *Segment size control*

   Segment Size Control berperan dalam menentukan *Maximum Segment Size* (MSS), yang merupakan ukuran maksimum data yang dapat dikirimkan dalam satu segmen TCP untuk mencegah terjadinya fragmentasi IP. Informasi mengenai MSS memberikan batas ukuran data maksimum yang dapat ditransmisikan sebagai segmen tunggal oleh TCP. Untuk memastikan kinerja yang optimal, disarankan untuk menetapkan MSS dengan ukuran yang kecil agar menghindari penahapan IP. Penahapan IP bisa menyebabkan paket-paket data terpisah dan pengiriman ulang yang berlebihan, sehingga dapat menyebabkan paket-paket hilang dan mengurangi efisiensi jaringan. Oleh karena itu, penggunaan MSS yang tepat dapat membantu mencegah masalah penahapan dan memastikan pengiriman data yang lebih lancar dan efisien. [15].

6. *Congestion Contol*

   Congestion Control merupakan mekanisme yang mengontrol masuknya paket data ke dalam jaringan, memungkinkan penggunaan yang lebih baik dari infrastruktur jaringan bersama dan menghindari keruntuhan kongestif. Salah satu mekanisme yang dilakukan adalah mensetting aliran data yang masuk ke jaringan [18].

2.2.4.2. **TCP *Port Number***

Dalam *Transmission Control Protocol* (TCP), sebuah *port* berjumlah 16-bit bilangan yang di konversi, dengan rentang dari 0 hingga 65535. Nomor *port* adalah cara untuk mengidentifikasi proses spesifik ke mana data akan dikirim setelah tiba di *host*. Nomor port TCP dibagi menjadi tiga yaitu:

1. *Port* terkenal (0-1023): Ini adalah *port* yang dicadangkan untuk tujuan tertentu dan biasanya digunakan oleh proses tingkat sistem atau oleh program yang memerlukan akses khusus. Contohnya termasuk *port* 80 (HTTP), *port* 21 (FTP), dan port 22 (SSH).

2. *Port* terdaftar (1024-49151): Ini adalah *port* yang terdaftar di Internet *Assigned Numbers Authority* (IANA) untuk tujuan tertentu. *Port* ini biasanya digunakan oleh program yang menyediakan layanan ke sistem lain.
3. *Port* dinamis atau pribadi (49152-65535): Ini adalah *port* yang tidak terdaftar secara resmi di IANA dan tersedia untuk digunakan oleh program apa pun yang memerlukan *port* [19].

2.2.4.3. TCP *Flag*

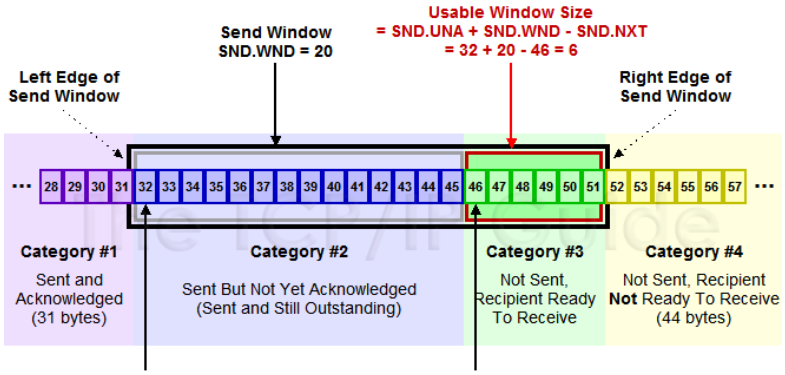
TCP Flag Merupakan bit kontrol yang menunjukkan status koneksi yang berbeda dan informasi tentang bagaimana sebuah paket harus ditangani. Dalam sebuah paket terdapat dua bagian yaitu *header* dan *payload*. Pada paket *header* terdapat beberapa komponen salah satunya komponen *flags*, terdiri dari enam yaitu:

1. URG (*Urgent*), mengidentifikasi data yang masuk sebagai data yang genting. Segmen yang masuk tersebut tidak harus menunggu sampai segmen sebelumnya dikonsumsi oleh pihak penerima tetapi dikirim langsung dan segera diproses.
2. ACK (*Acknowledgment*) menandakan bahwa bidang *acknowledgment* berisi item yang diinginkan berikutnya dalam koneksi. Flag ini selalu berubah, terutama di segmen pertama sepanjang proses pembuatan koneksi TCP.
3. PSH (*Push*), untuk memastikan bahwa data diberi prioritas dan diproses di ujung pengiriman atau penerimaan
4. RST (*Reset Connection*), digunakan ketika segmen tiba yang tidak dimaksudkan untuk koneksi saat ini.
5. SYN (*Synchronize*), menunjukkan kalau segmen TCP yang berkaitan mengandung *Intial Seguence Number* (ISN).
6. FIN (*Finish*), menunjukkan bahwa *sender* segmen TCP telah habis dalam mengirimkan paket dalam suatu koneksi TCP [4].

2.2.4.4. TCP *Window Size*

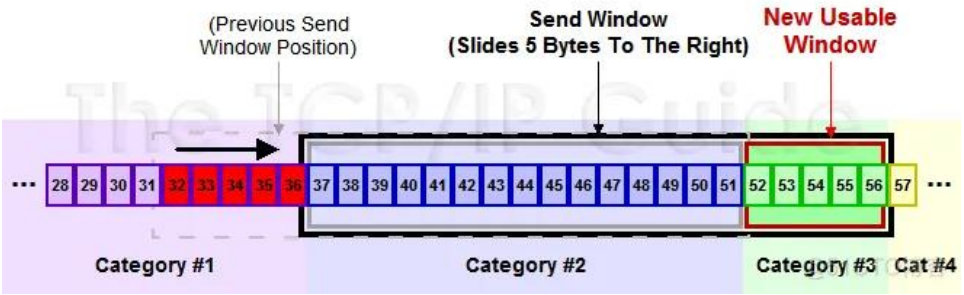
Window Size merupakan salah satu bagian dari *header* TCP yang menetapkan seberapa banyak dan seberapa cepat data dapat dikirim oleh pengirim tanpa harus menunggu konfirmasi dari penerima. TCP akan memilih kecepatan transmisi data yang paling optimal yang sesuai dengan kemampuan jaringan dan

perangkat yang terlibat, dengan tujuan mengurangi kebutuhan untuk melakukan pengiriman ulang data (retransmisi) [4].



Gambar 2.6 Struktur TCP *Window* [4].

Gambar 2.6 merupakan struktur TCP *Window*, *window* mentapkan seberapa banyak *byte* yang bisa dikirim oleh pengirim, mencakup jumlah *byte* dari paket yang telah dikirim tetapi belum dikonfirmasi (kategori 2) dan paket yang siap untuk dikirim namun belum terkirim (kategori 3). Setelah pengirim menerima konfirmasi, *byte* yang termasuk dalam kategori 2 akan dipindahkan ke kategori 3 dan 4. Proses pemindahan ini disebut sebagai *sliding* dan berfungsi sebagai mekanisme pengontrol aliran data. Dengan *sliding window*, pengirim dapat mengatur aliran data dengan efisien, memastikan bahwa jumlah data yang dikirim sesuai dengan kapasitas jaringan dan penerima sehingga menghindari kelebihan pengiriman atau kehilangan data. [4].



Gambar 2.7 Proses pergeseran pada TCP *Window* [4].

Gambar 2.7 merupakan proses pergeseran pada TCP *Window*, ukuran jendela (*window size*) mengatur jumlah data yang dapat dikirim dalam satu waktu. Dengan menggunakan ukuran jendela, setiap segmen data harus dikonfirmasi sebelum segmen berikutnya dikirimkan, sehingga memungkinkan penggunaan *bandwidth* yang lebih efisien oleh setiap perangkat *host* yang terlibat dalam

komunikasi. Penting untuk dicatat bahwa ukuran jendela TCP adalah variabel selama koneksi berlangsung. Setiap konfirmasi yang dikirim mencakup ukuran jendela, yang menunjukkan berapa banyak jumlah data yang dapat diterima oleh penerima. Namun, paket yang diterima oleh penerima tidak selalu langsung diteruskan ke penerima. Mereka pertama-tama disimpan dalam *buffer*. Terkadang, penerima harus mengendalikan aliran paket yang diterimanya, misalnya jika penerima menerima paket dengan ukuran jendela 140, tetapi sebenarnya hanya dapat meneruskan 40-byte data ke aplikasi. Dalam kasus ini, sisanya sebanyak 100-byte data harus tetap disimpan dalam *buffer* sampai aplikasi siap untuk memprosesnya. Salah satu alasan tambahan adalah bahwa komunikasi tidak selalu terjadi secara langsung antara dua perangkat saja. Misalnya, dalam kasus transmisi dari klien ke server, server tersebut juga bisa melayani komunikasi dengan banyak perangkat lainnya. Karena itu, *buffer* server dapat terisi dengan data saat datangnya paket dari berbagai sumber yang berbeda. Jika situasi ini berlanjut, akibatnya data dapat gagal diterima atau bahkan terbuang. Untuk mengatasi masalah ini, TCP mempunyai kemampuan untuk mengatur kontrol kemacetan (*congestion control*) dengan ukuran jendela yang sama dengan ukuran jendela penerima. Namun, ukuran jendela akan dibagi menjadi setengahnya jika terjadi kehilangan paket atau terjadi kemacetan pada jaringan. Pendekatan ini memungkinkan ukuran jendela untuk ditingkatkan sesuai kebutuhan untuk mengontrol ruang *buffer* dan pemrosesan data. Dengan menggunakan ukuran jendela yang lebih besar, TCP dapat memproses lebih banyak data sekaligus [4].

2.2.5. Kongesti

Kongesti terjadi ketika penggunaan kapasitas jaringan melebihi batas yang tersedia, sehingga *buffer* pada router menjadi terlalu banyak beban. Kemacetan pada jaringan nirkabel sering terjadi ketika adanya kepadatan pengiriman data dan kurangnya kapasitas *bandwidth* sehingga menyebabkan komunikasi data terputus pada saat pengiriman data. Ketika *buffer* telah mencapai kapasitas maksimalnya, paket yang tiba selama periode kongesti akan diabaikan, menyebabkan penurunan jumlah data yang diterima dan juga menyebabkan delay yang tidak dapat diprediksi. Untuk mengatasi masalah kongesti tersebut, digunakan mekanisme berikut in [20]:

1. *Congestion avoidance*

Congestion avoidance adalah sebuah cara untuk mencegah terjadinya kongesti. Dalam TCP, pencegahan kemacetan menggunakan kehilangan paket sebagai indikator adanya kongesti, sementara ada juga yang menggunakan perhitungan waktu respons (RTT) sebagai indikator terjadinya kongesti.

2. *Congestion Control*

   Congestion control merupakan cara yang digunakan apabila kemacetan telah terjadi. *Congestion control* diterapkan melalui dua sisi yaitu:

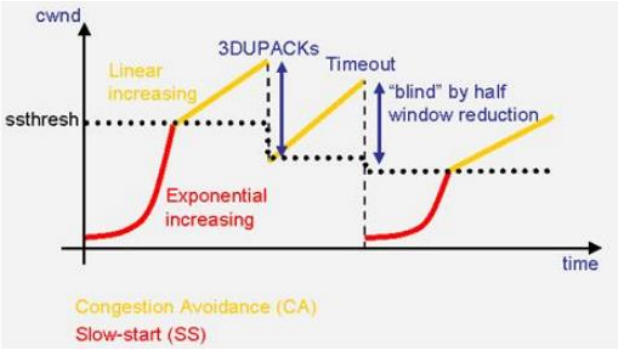
   a. Pengirim mengontrol jumlah data yang dikirimkan kepada penerima dengan menyesuaikan jumlah data yang dikirimkan sesuai dengan kondisi jaringan. Hal ini dapat dilakukan dengan mengimplementasikan algoritma kontrol kemacetan seperti *Additive Increase/Multiplicative Decrease* (AIMD) atau *Random Early Detection* (RED).
   b. Penerima mengontrol jumlah data yang diterimanya dengan mengirimkan *window advertisement* kepada pengirim, yang menunjukkan berapa banyak *byte* yang dapat diterima penerima pada saat ini. Hal ini dapat membantu mengontrol aliran data dan mencegah terjadinya kemacetan pada jaringan [20].

2.2.6. TCP *New Reno*

TCP *New Reno* adalah peningkatan dari TCP Reno, tetapi perbedaan utamanya terletak pada *fase fast recovery*. TCP *New Reno* dapat menangani beberapa kesalahan paket atau paket yang jatuh. Dalam fase ini, TCP *New Reno* membuat sedikit pembaruan atau perbaikan. Saat TCP Reno dan TCP New Reno mengalami *single packet drop*, kedua TCP tersebut akan tetap berada dalam fase transmisi ulang dan tidak masuk ke fase pemulihan. Tetapi jika TCP menemukan beberapa paket yang salah atau dua kesalahan *multi*-paket, TCP Reno akan memasuki fase awal yang lambat, tetapi TCP *New* Reno akan terus beroperasi dalam fase *fast recovery*. TCP New Reno adalah algoritme yang dirilis dari TCP Reno.

Mirip dengan TCP *New* Reno, yang akan melanjutkan fase *fast recovery* karena dapat menangani penurunan *multi*-paket, TCP Reno akan gagal dan berpindah dari fase awal yang lambat jika terjadi penurunan multi-paket paket. TCP

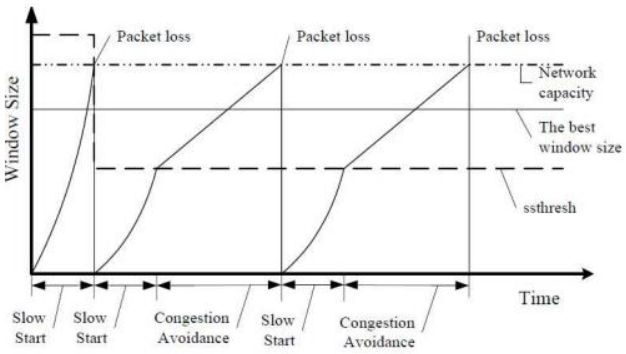
New Reno hanya dapat mengelola dua paket tetes atau kesalahan setiap operasi multi-paket *drop* atau kesalahan. Jika ada lebih dari 2 paket yang dijatuhkan, itu akan mengakibatkan batas waktu dan pindah ke fase berikutnya [4]. Pada gambar 2.8 merupakan kontrol kongesti pada TCP *New* Reno.



Gambar 2.8 Kontrol Kongesti pada TCP *New Reno*

2.2.7. TCP *Tahoe*

TCP *Tahoe* adalah implementasi pertama dari *Transmission Control Protocol* (TCP) dan termasuk mekanisme seperti *Slow Start*, *congestion avoidance*, dan *Fast* Retransmit. Dalam *Slow Start*, sistem dimulai dengan *window size* awal yang kecil dan secara bertahap meningkatkannya. Jika terjadi kehilangan paket, *window size* akan dikurangi dan sistem memasuki mode *Congestion Avoidance*, di mana ukuran jendela meningkat secara linear. *Fast* Retransmit memungkinkan untuk retransmisi paket yang hilang sebagai respons dari menerima tiga duplikat pengakuan, tanpa menunggu *timeout* untuk berakhir. Hal ini menyebabkan penggunaan yang lebih efisien dari saluran dan peningkatan tingkat koneksi, tapi tidak menggunakan mekanisme *fast recovery* [21]. Pada gambar 2.9 merupakan *fase slow start* dan *congestion avoidance* pada TCP Tahoe.



Gambar 2.9 *Congestion Window* TCP *Tahoe* [22].

Algoritma Tahoe merupakan sebuah protokol TCP yang banyak digunakan untuk pengaturan ukuran jendela (*window size*) untuk memastikan komunikasi *end-to-end*. Karakteristik dari TCP Tahoe dijelaskan dalam Gambar 2.9 yang beroperasi dengan cara sebagai berikutt:

- Setelah *fast retransmission* TCP mengatur *window size* 1 dan *sstresh* menjadi *window size*/2.
- TCP memasuki fase *slow start*.
- Ketika *window size* mencapai *ssthreshold*, TCP men-*trigger congestion avoidance* [22].

a. *Slow start*

Slow-start adalah tahap awal dalam protokol TCP yang bertujuan untuk mengetahui kapasitas jaringan yang tersedia. Pada tahap ini, TCP akan mengirimkan satu paket data dan menunggu konfirmasi (ACK) dari penerima. Jumlah paket yang dikirimkan akan terus meningkat dengan tingkat pertumbuhan yang eksponensial, misalnya dari satu paket, dua paket, empat paket, dan seterusnya.

Peningkatan jumlah paket secara eksponensial ini akan berhenti ketika terdeteksi bahwa kenaikan tersebut telah menyebabkan adanya kehilangan paket atau telah mencapai titik batas tertentu yang disebut "ssthreshold" (*Slow-Start Threshold*). Setelah mencapai titik ssthreshold, kenaikan jumlah paket yang dikirimkan akan beralih dari eksponensial menjadi linier. Dalam tahap ini, pertumbuhan jumlah paket akan lebih stabil dan berlangsung dengan tingkat pertumbuhan yang lebih terkontrol [10].

b. *Congestion Avoidance*

Congestion Avoidance merupakan tahap di mana TCP berusaha menghindari terjadinya kongesti pada jaringan. Pada tahap ini, ukuran *window* kongesti (CWND) akan meningkat secara linier dengan pertambahan sebesar 1 untuk setiap ACK yang diterima dari penerima. Namun, jika terjadi 3 ACK yang merupakan duplikasi, maka nilai ssthreshold akan dikurangi menjadi setengah dari nilai CWND saat itu, dan nilai CWND sendiri akan diturunkan ke nilai ssthreshold tersebut. Dengan demikian, ukuran jendela akan

dikendalikan dengan lebih hati-hati untuk menghindari terjadinya kongesti pada jaringan [10].

c. *Fast Retransmit*

Pada tahap ini, terjadi retransmit pada paket-paket yang hilang. Jika TCP penerima menerima tiga ACK yang merupakan duplikasi, maka akan dilakukan retransmisi terhadap paket yang diduga hilang. *Fast* retransmit adalah sebuah perbaikan pada protokol TCP yang bertujuan untuk mengurangi waktu penundaan oleh pengirim sebelum melakukan retransmisi terhadap segmen data yang hilang. Pengirim TCP akan menggunakan *timer* untuk memantau segmen tertentu dalam jangka waktu tertentu. Jika *timer* ini berakhir sebelum adanya konfirmasi (ACK) dari penerima, pengirim akan mengasumsikan bahwa segmen tersebut akan hilang dalam jaringan, dan akan melakukan retransmisi untuk segmen yang hilang tersebut. Tujuan dari tahap retransmisi ini adalah untuk memberikan respon yang cepat jika terjadi kehilangan paket, sehingga data dapat dikirim ulang dengan segera untuk memastikan kehandalan dan kecepatan komunikasi [10].

Duplikat *acknowledgement* (duplikat ACK) adalah dasar dari mekanisme pengiriman ulang cepat berikut: Setelah menerima paket (misalnya, paket dengan nomor urutan 1), penerima akan mengirimkan pemberitahuan (ACK) dengan menambahkan nomor 1 ke alamat paket. nomor urut, yaitu urutan 2. Hal ini menunjukkan bahwa penerima sudah menerima paket dengan nomor "Urutan 1" dan mengharapkan paket dengan nomor "Urutan 2" nantinya. Asumsi selanjutnya adalah paket di bawah (nomor urutan 2) telah rusak. Meskipun demikian, penerima tetap mendapatkan paket dengan nomor 3 dan 4 di dalamnya. Setelah menerima paket yang bertuliskan resi nomor 3, penerima melanjutkan pengiriman surat pemberitahuan, tetapi hanya menyebutkan resi nomor 2, bukan resi nomor 3. Mulai lagi ketika penerima menerima paket dengan nomor referensi "Urutan 5", penerima tetap mengirimkan pemberitahuan sambil menyebutkan "Urutan 2". Karena pengirim menerima tiga ACK dengan nomor "Urutan 2", pengirim menyimpulkan bahwa paket dengan nomor tersebut telah kedaluwarsa. Sebagai tanggapan, pengirim akan mengirim ulang paket untuk memastikan bahwa data

yang dikirim akurat. Karena mekanisme pengiriman ulang yang cepat, pengirim dapat mendeteksi paket yang tertinggal dengan cepat berdasarkan tiga ACK yang diterima dari penerima, memungkinkan pengiriman ulang terjadi tanpa batas waktu. [10].

d. *Fast Recovery*

Pada fase *fast recovery*, ketika terjadi tiga duplikasi ACK setelah melakukan *fast* retransmission, TCP tidak akan kembali ke fase *slow start* seperti biasanya. Sebaliknya, TCP langsung memasuki fase *congestion avoidance*. Dalam fase *fast recovery*, *throughput* tetap dijaga tinggi meskipun terjadi kecil dan sedang kongesti pada jaringan [1]. Perbedaan utama antara TCP Reno dan TCP New Reno terletak pada fase *fast recovery*. Jika ada dua paket yang hilang atau mengalami kesalahan (multi paket *drop*) dalam satu paket, maka TCP Reno akan mengalami kegagalan dan memulai kembali dari fase *slow start*. Sementara itu, TCP New Reno memiliki kemampuan untuk menangani multi paket *drop* dan tetap bertahan dalam fase *fast recovery*. Dengan demikian, TCP New Reno dapat lebih efisien dalam mengatasi situasi dengan multi paket *drop* dan menghindari memulai dari fase *slow start* [10].

Dengan menggunakan mekanisme fast retransmit saja, ketika terdeteksi kongesti (*packet loss*), ukuran *congestion window* akan diturunkan menjadi satu. Namun, karena pengiriman data masih dapat menghasilkan *acknowledgment*, jika terjadi *timeout* (tidak ada ACK yang diterima dalam batas waktu tertentu), TCP kembali ke tahap *slow start*. Namun, ketika terjadi 3 duplikasi ACK, TCP akan memasuki fase *fast recovery* tanpa harus kembali ke tahap *slow start* seperti yang terlihat dalam grafik pada Gambar 2.11. Prosedur *fast recovery* saat mendeteksi kehilangan paket melalui tiga duplikasi ACK dalam TCP New Reno berjalan sebagai berikut:

a. *Ssthreshold* = CWND/2
b. CWND = *ssthreshold*
c. Melakukan *fast retransmit*
d. Melakukan *fast recovery*
e. Lalu memasuki fase *congestion avoidance*

Berikut ini adalah prosedur ketika terjadi RTO pada paket:

a. *Ssthreshold* = CWND/2

b. CWND = 1

c. Melakukan *fast retransmit*

d. Lalu memasuki fase *slow start*

3 duplikasi ACK juga digunakan sebagai indikator adanya kongesti melalui deteksi kehilangan paket. Terkadang, TCP Reno dapat mengalami *timeout* karena terjadinya *multiple loss* pada paket yang dikirimkan. Pengaruh jumlah dari *multiple loss* yang terjadi terhadap terjadinya *timeout* dapat dijelaskan sebagai berikut:

1) Ketika ada dua *packet loss*, *timeout* terjadi kadang-kadang.
2) Ketika ada tiga *packet loss*, *timeout* biasanya terjadi.
3) Ketika ada empat *packet loss*, *timeout* dipastikan terjadi [10].

2.2.8. *Software* Simulasi RIVERBED *Modeler*

RIVERBED Modeler adalah perangkat lunak yang dapat digunakan untuk seluruh tahap penelitian, mulai dari desain model, simulasi, pengumpulan data, hingga analisis data. RIVERBED Modeler memberikan lingkungan pengembangan yang lengkap untuk mendukung pembuatan model jaringan komunikasi dan sistem terdistribusi. Melalui simulasi menggunakan metode kejadian diskrit, baik perilaku maupun kinerja dari model jaringan dapat dianalisis. Perangkat lunak ini menawarkan aplikasi *Graphical User Interface* (GUI) yang memudahkan untuk mengkonfigurasi skenario dan membangun model jaringan. Selain itu, hasil simulasi dapat diperoleh untuk berbagai skenario yang dapat menjadi dasar bagi suatu jaringan perencana berdasarkan paket. Sederhananya, RIVERBED Modeller membantu peneliti dan operator jaringan dalam memahami dan memprediksi bagaimana jaringan dan sistem dasarnya berperilaku dalam berbagai situasi [23]. Pada Gambar 2.11 ini merupakan tampilan depan *software* riverbed *modeler*.

Gambar 2.11 *Software* RIVERBED *Modeler*

RIVERBED *Modeler* adalah perangkat lunak simulasi jaringan yang memiliki berbagai kelebihan. Salah satunya adalah kemampuannya untuk mendesain jaringan berdasarkan perangkat yang ada di pasaran, protokol, layanan, dan teknologi terkini dalam industri telekomunikasi. Perangkat lunak ini dirancang oleh OPNET *Technologies* Inc. RIVERBED Modeler dapat memberikan percepatan dalam penelitian dan pengembangan jaringan (R&D *network*), mengurangi waktu yang dibutuhkan untuk menghadirkan produk ke pasar (*time-to-market*), serta meningkatkan kualitas produk yang dihasilkan. Dengan fitur-fitur yang dimilikinya, perangkat lunak ini memungkinkan para pengguna untuk melakukan simulasi dan analisis dengan akurasi tinggi, sehingga dapat membantu dalam pengambilan keputusan yang lebih baik dan efisien terkait dengan perencanaan, pengembangan, dan optimisasi jaringan.

Salah satu cara yang praktis untuk merencanakan jaringan berbasis paket adalah dengan melakukan simulasi jaringan dengan kondisi yang mirip dengan situasi yang ada. Dengan menggunakan beberapa model simulasi, kita dapat memprediksi kebutuhan jaringan seperti lebar pita (*bandwidth*), kebutuhan untuk mengendalikan kongesti (*congestion control*), jenis perangkat yang sesuai, dan hal lainnya. Hasil simulasi ini membantu kita untuk merencanakan jaringan berbasis protokol *Internet Protocol* (IP) dengan lebih efektif. RIVERBED *Modeler* memungkinkan simulasi jaringan berbasis paket, termasuk simulasi dengan protokol *Internet Protocol* (IP), *Asyncronous Transfer Mode* (ATM), *Frame Relay*, atau TDM. Berbagai jenis layanan juga dapat disimulasikan, seperti layanan internet (WEB), VolP, *transfer file, video conference, streaming video*, dan lainnya, sesuai dengan kebutuhan dari pengguna simulasi. Hasil dari simulasi menggunakan perangkat lunak RIVERBED meliputi berbagai variabel kinerja jaringan seperti *delay, throughput, bit error rate*, dan lain-lain. Hasil simulasi ini dapat digunakan untuk membuat beberapa skenario yang berbeda, sehingga menjadi dasar yang kuat dalam perencanaan jaringan berbasis paket [23].

2.2.9. *File Transfer Protocol* (FTP)

File Transfer Protocol (FTP) adalah protokol jaringan standar yang digunakan untuk mengirim file dari satu komputer ke komputer lain melalui jaringan berbasis TCP, seperti Internet. FTP menggunakan arsitektur server-klien

dan mendukung dua jenis koneksi, yaitu koneksi kontrol dan data, antara klien dan server. Dengan menggunakan protokol yang jelas yang biasanya terdiri dari nama pengguna dan kata rahasia, pengguna dapat mengidentifikasi dirinya sendiri. Namun, FTP juga memungkinkan koneksi anonim jika server dikonfigurasi untuk mengizinkannya. Agar transmisi data lebih aman dan melindungi informasi seperti nama pengguna dan kata sandi, serta mengenkripsi isi *file* yang di*transfer*, FTP sering kali dilengkapi dengan keamanan SSL/TLS, yang dikenal sebagai FTPS. Dengan demikian, FTPS dapat menyediakan lapisan keamanan tambahan melalui enkripsi data yang dikirimkan melalui protokol ini.

FTP (*File Transfer Protocol*) berfungsi sebagai sarana untuk mentransfer file dan data melalui jaringan apa pun yang menggunakan koneksi TCP. Saat digunakan, FTP sering merekomendasikan perangkat lunak berbasis *Open Source* untuk memastikan tingkat stabilitas yang tinggi dan kemudahan penggunaan yang lebih besar dalam mendeteksi infeksi virus dan *malware*. FTP menjadi pilihan yang tepat sebagai metode protokol untuk menyimpan dan men*transfer* file atau data dengan cepat, baik saat proses upload maupun *download* antara komputer server dan klien, tanpa perlu menggunakan *flashdisk* sebagai perantara untuk mengambil data dari komputer server. Dengan demikian, FTP memungkinkan pertukaran data yang efisien dan praktis di antara perangkat dalam jaringan [24].

2.2.10. *Congestion Window*

Congestion Window (CWND) adalah suatu parameter yang digunakan oleh protokol TCP untuk mengatur jumlah data yang dapat dikirimkan oleh pengirim sebelum menerima konfirmasi (ACK) dari penerima. Fungsi dari CongWin ini adalah untuk menghindari terjadinya kelebihan muatan (*congestion*) dalam jaringan yang dapat menyebabkan penurunan kinerja jaringan dan kehilangan paket data. Dengan mengendalikan ukuran CWND, TCP dapat mengatur laju pengiriman data agar sesuai dengan kondisi jaringan dan mencegah terjadinya kongesti yang berpotensi merugikan kinerja komunikasi.

Ketika pengirim mengirimkan paket data, *Congestion Window* (CWND) akan menentukan jumlah data yang dapat dikirim berdasarkan kapasitas jaringan yang tersedia. Setiap kali pengirim menerima konfirmasi (ACK) dari penerima bahwa paket data telah diterima, CWND akan ditingkatkan untuk memungkinkan

pengiriman lebih banyak data. Namun, jika terjadi kelebihan muatan atau kongesti dalam jaringan, pengirim akan menerima sinyal dari jaringan dan CWND akan dikurangi untuk mengurangi jumlah data yang dikirimkan. Dengan mengendalikan jumlah data yang dikirimkan, CWND memastikan bahwa jaringan tidak mengalami kongesti yang berlebihan, sehingga pengiriman data dapat dilakukan dengan efisien dan handal [22].

2.2.11. *Throughput*

Throughput, yaitu kecepatan transfer data efektif, yang diukur dalam bps. *Throughput* merupakan jumlah total kedatangan paket yang sukses yang dapat diamati pada *destination* selama interval waktu tertentu dibagi oleh durasi interval waktu tersebut [6].

Rumus untuk menhitung *throughput* adalah:

$$Throughput = \frac{jumlah\ data\ yang\ dikirim}{waktu\ pengiriman\ data} \quad (3.1)$$

Tabel 2.2 Klasifikasi kualitas *Throughput*

Kualitas Degradasi	***Throughput***
Sangat bagus	100%
Bagus	75%
Sedang	50%
Buruk	< 25%

Tabel 2.2 merupakan klasifikasi kualitas *throughput*, dimana kategori degradasi sangat bagus jika nilai *throughput* 100%, kategori degradasi bagus jika nilai *throughput* 75%, kategori degradasi sedang jika nilai *throughput* 50%, dan kategori degradasi buruk jika nilai throuhgput < 25% [6].

2.2.12. *Round Trip Time* (RTT)

RTT (*Round Trip Time*) adalah waktu yang diperlukan oleh suatu paket data untuk melakukan perjalanan dari pengirim ke penerima dan kembali lagi ke pengirim. RTT merupakan ukuran yang digunakan untuk mengukur waktu respons atau latensi yang terjadi pada jaringan komputer. Proses pengukuran RTT dilakukan dengan cara pengirim mengirimkan sebuah paket data ke penerima dan kemudian menunggu balasan atau respons dari penerima. Ketika paket data diterima oleh penerima, penerima akan mengirimkan pesan balasan (*acknowledgment)* ke

pengirim yang menunjukkan bahwa paket data telah diterima dengan baik. Waktu yang diperlukan untuk pengiriman paket data dan penerimaan pesan balasan ini akan menjadi RTT. RTT sangat penting dalam mengukur kecepatan dan kualitas respons jaringan, karena mempengaruhi kinerja dan efisiensi komunikasi dalam jaringan.

RTT merupakan waktu yang diukur dalam satuan waktu seperti milidetik (ms) atau mikrodetik (μs), tergantung pada tingkat kecepatan jaringan yang digunakan. Nilai RTT yang kecil menandakan jaringan yang responsif atau cepat, sedangkan nilai RTT yang besar menunjukkan jaringan yang kurang responsif atau lambat. Pengukuran RTT ini memiliki peran penting dalam mengukur kinerja jaringan dan dapat mempengaruhi kualitas layanan serta *throughput* pada jaringan tersebut. Contohnya, dalam protokol jaringan seperti TCP (*Transmission Control Protocol*), RTT dapat mempengaruhi kinerja protokol tersebut. Semakin lama waktu respons jaringan (RTT), semakin lambat juga pengiriman data pada protokol TCP, yang pada gilirannya dapat berdampak pada kualitas layanan atau *throughput* pada jaringan tersebut. Dengan demikian, pemahaman tentang RTT sangat penting untuk meningkatkan efisiensi dan kualitas komunikasi dalam jaringan [25].

2.2.13. *Delay*

Delay merupakan total waktu yang dibutuhkan oleh suatu paket data dari pengirim hingga sampai ke penerima melalui jaringan. Proses *delay* dari pengirim ke penerima umumnya terdiri dari tiga komponen, yaitu *hardware latency*, *delay* akses, dan *delay* transmisi. Diantara ketiganya, *delay* transmisi adalah jenis *delay* yang paling sering terjadi bagi lalu lintas data yang melewati jaringan [6].

Rumus untuk menghitung *delay* adalah:

$$Rata - rata\ delay = \frac{Waktu\ penerimaan\ paket - waktu\ pengiriman\ paket}{total\ data\ yang\ diterima} \quad (3.2)$$

Tabel 2.3 Klasifikasi kualitas *Delay*

Kualitas Degradasi	*Delay*
Sangat bagus	< 150 ms
Bagus	150 s/d 300 ms
Sedang	300 s/d 450 ms
Buruk	> 450 ms

Tabel 2.4 merupakan klasifikasi kualitas *Delay*, dimana kategori degradasi sangat bagus jika nilai *delay* <150 ms, kategori degradasi bagus jika nilai *delay* 150 s/d 300 ms, kategori degradasi sedang jika nilai *delay* 300 s/d 450 ms dan kategori degradasi buruk jika nilai *delay* 450 ms [6].

2.2.14. *Mobile Ad-Hoc Network (MANET)*

Jenis jaringan *Ad-hoc* yang paling umum adalah jaringan *Ad-hoc* seluler (MANET), yang terdiri dari sekelompok perangkat bergerak yang dapat berkomunikasi satu sama lain melalui koneksi nirkabel. Setiap *node* dalam jaringan ini, juga dikenal sebagai perangkat, dilengkapi dengan antena omnidirectional. Setiap *node* dalam MANET memiliki kemampuan routing yang mirip dengan fungsi router pada umumnya, yaitu mengenkripsi dan memperkuat saluran komunikasi antar *node*. Selain itu, *node* dalam jaringan dapat beroperasi dalam keadaan bebas tanpa perlu terlibat dengan infrastruktur tetap jaringan.

Dua *node* lemah yang berada dalam jangkauan transmisi yang seragam dapat berkomunikasi secara diam-diam dalam jaringan *mobile ad hoc* (MANET). Namun, jika tidak ada koneksi yang berkesinambungan antara dua *node* yang bersangkutan, maka *node* yang terletak di antara keduanya dapat bertindak sebagai *relay* atau perantara untuk mendorong paket data agar komunikasi tetap berjalan. Setiap *node* dalam jaringan harus hadir dan berpartisipasi aktif dalam proses transfer paket data untuk mengirimkan data dari sumber ke tujuan yang dituju. *File* yang perlu dihapus dikemas menjadi beberapa paket data yang lebih kecil, dan setiap paket dihapus menggunakan jalur jaringan yang berbeda. Setiap paket data di dalamnya dienkripsi dan dihubungkan secara aman di *node* target untuk menghasilkan file yang otentik dan komprehensif. Akibatnya, setiap *node* MANET bekerja dengan rajin untuk mengirim dan memantau paket data untuk memastikan komunikasi yang sukses di seluruh *node* dalam jaringan.

Ada tiga jenis protokol routing yang dapat digunakan dalam jaringan *ad-hoc*: hybrid, reaktif, dan proaktif. Tujuan dari protokol proaktif, juga dikenal sebagai protokol berbasis tabel, adalah untuk terus memperbarui topologi seluruh jaringan. Dengan memanfaatkan informasi topologi ini, protokol proaktif dapat dengan mudah mengidentifikasi lokasi terbaik untuk mengirim paket data pada saat dibutuhkan. Selain itu, protokol reaktif atau disebut juga *on-demand* akan

melakukan komunikasi berdasarkan permintaan dari *mobile node*. Pemrosesan paket data hanya terjadi ketika sebuah *node* memintanya, membuat biaya untuk memindahkan paket ke rute yang lebih tinggi daripada di bawah protokol proaktif yang terus menerus mengirimkan paket untuk memberi tahu *node* tentang perubahan rute. Routing *hybrid* adalah kombinasi protokol proaktif dan reaktif yang mencari solusi terbaik berdasarkan persyaratan dan kondisi jaringan yang terus berubah [2].