

BAB III

TRANSMISSION CONTROL PROTOCOL & USER DATAGRAM PROTOKOL

3.1 USER DATAGRAM PROTOKOL

UDP menyediakan mekanisme dasar yang digunakan oleh program aplikasi untuk mengirim datagram ke program aplikasi lain. UDP menyediakan port protokol yang digunakan untuk membedakan satu program yang sedang dieksekusi dengan yang lain dalam satu mesin.

UDP menggunakan protokol dibawahnya (IP) untuk menyampaikan pesan dari satu mesin ke mesin lain dan menyediakan semantic pengiriman datagram yang tidak reliable, serta connectionless seperti IP. Disini tidak terdapat ACK untuk memastikan sampainya pesan di tujuan, data yang sampai tidak perlu terurut, dan tidak membutuhkan kontrol balik terhadap rate aliran informasi antar mesin.

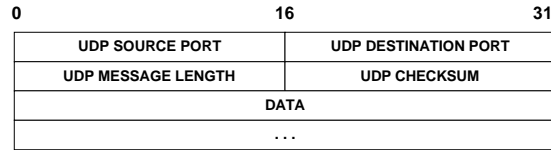
Suatu program aplikasi yang menggunakan UDP menerima tanggung jawab penuh untuk menangani masalah reliabilitas termasuk kehilangan pesan, duplikasi, delay, pengiriman yang tidak terurut dan putus koneksi.

Karena sifatnya yang connectionless dan unreliable, UDP digunakan oleh aplikasi-aplikasi yang secara periodic melakukan aktivitas tertentu (misalnya query routing tabel pada jaringan local), serta hilangnya satu data akan dapat di atasi pada query periode berikutnya dan melakukan pengiriman data ke jaringan lokal. Pendeknya jarak tempuh datagram akan mengurangi resiko kerusakan data.

Bersifat broadcasting atau multicasting. Pengiriman datagram ke banyak client sekaligus akan efisien jika prosesnya menggunakan metode connectionless

3.1.1 Format data UDP

Setiap pesan UDP disebut dengan user datagram. Pesan ini terdiri dari dua bagian yaitu header UDP dan data UDP, seperti pada gambar dibawah ini :



gambar Format datagram UDP

Field **SOURCE PORT** & **DESTINATION PORT** berisi 16 bit nomor port protocol UDP yang digunakan untuk men-demultiplex-kan datagram diantara proses-proses yang menunggu untuk menerimnya. Penggunaan **SOURCE PORT** adalah opsional. Ketika digunakan, field tersebut menspesifikasikan port kemana reply seharusnya dikirimkan.

LENGTH berisi jumlah oktet dalam datagram UDP termasuk di dalamnya header UDP & data user.

UDP CHECKSUM bersifat opsional dan tidak harus digunakan semua. Nilai 0 berarti checksum tidak harus dikomputasi.

3.1.2 UDP Pseudo-Header

Untuk mengkomputasi checksum UDP menyediakan /prepend pseudo-header ke datagram UDP, menambahkan sebuah oktet bit 0 ke pad datagramnya menjadi tepat kelipatan 16 bit dan mengkomputasi checksum dari seluruh objek yang ada. Pseudo-header tidak ikut ditransmisikan dalam datagram dan panjangnya pun tidak termasuk dalam length yang ada.

Untuk mengkomputasi sebuah checksum, pertama kali software akan menyimpan bit nol dalam field **CHECKSUM**, kemudian mengakumulasi 16 bit seluruh objek (pseudo-header, UDP header dan data user) ke dalam bentuk komplement satu.

Penggunaan pseudo header bertujuan untuk menjamin data sampai ke tujuan.

Pseudo-header ini terdiri dari 12 oktet data seperti pada gambar dibawah ini :

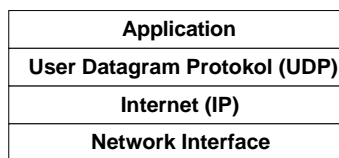
SOURCE IP ADDRESS		
DESTINATION IP ADDRESS		
ZERO	PROTO	UDP LENGTH

gambar format pseudo header

SOURCE ADDRESS & DESTINATION ADDRESS digunakan untuk menentukan alamat sumber & tujuan yang akan digunakan untuk pengiriman pesan UDP. PROTO berisi tipe protokol IP dimana code 17 adalah kode untuk UDP. UDP LENGTH berisi panjang datagram UDP tidak termasuk didalamnya panjang pseudo-headernya. Untuk verifikasi checksum, penerima harus mengekstrak field-field ini dari header IP, dan membentuknya dalam format pseudo-header dan kemudian mengkomputasi checksum.

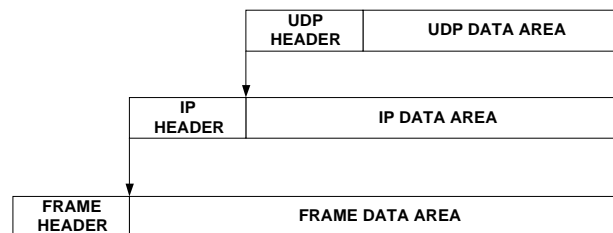
3.1.3 Layer-layer Protokol & Enkapsulasi UDP

UDP terletak di atas layer protokol Internet. Secara konseptual program aplikasi akan mengakses UDP, dimana dengan menggunakan IP untuk mengirim dan menerima datagram. Konsep layering UDP dapat digambarkan sebagai berikut :



gambar konsep layering protokol UDP

Pesan UDP yang terdiri dari data & header UDP akan dienkapsulasi dalam datagram IP agar bisa melalui jaringan Internet. Proses enkapsulasi dapat dilihat pada gambar berikut :



gambar enkapsulasi datagram UDP dalam datagram IP

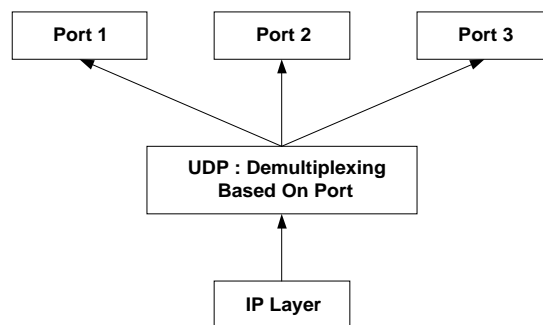
Enkapsulasi disini berarti proses penambahan atribut (bisa header maupun trailer) ke suatu data agar data tersebut dikenali dan dapat dilewatkan pada suatu protokol tertentu.

3.1.4 Multiplexing, Demultiplexing dan Ports

Sebuah software yang melalui layer-layer dari suatu hirarki protokol harus me-multiplex atau me-demultiplex-kan banyak objek pada layer berikutnya. UDP software menerima datagram UDP dari banyak program aplikasi dan melakukannya untuk transmisi protokol IP. Software UDP juga akan menerima kedatangan datagram UDP dari IP dan melakukannya ke program aplikasi yang sesuai.

Secara konseptual semua proses multiplexing dan demultiplexing antara software UDP dan program aplikasi terjadi melalui mekanisme port. Setiap program aplikasi harus bernegosiasi dengan sistem operasi untuk mendapatkan port protokol dan nomor port yang bersesuaian sebelum mengirim datagram UDP. Sekali sebuah port sudah dipakai oleh suatu program aplikasi, maka datagram dari program aplikasi mengirim melalui port akan memiliki nomor port tersebut dalam field SOURCE PORT.

Ketika memproses input, UDP akan menerima kedatangan datagram dari software IP dan men-demultiplex-kan berdasarkan port tujuan UDP seperti pada gambar berikut :



gambar contoh demultiplexing satu layer dia atas IP

cara termudah adalah membayangkan port UDP sebagai sebuah antrian. Di sebagian besar implementasi, ketika suatu program aplikasi bernegosiasi dengan suatu sistem operasi untuk penggunaan port, sistem operasi akan membentuk suatu antrian internal

yang dapat menangani kedatangan pesan. Aplikasi dapat menentukan atau mengubah ukuran antrian.

Ketika suatu datagram tiba, aplikasi akan mengecek untuk melihat apakah destination port-nya sesuai dengan salah satu port yang sedang digunakan. Jika ada yang sesuai maka UDP akan mengantri datagram tersebut pada port dimana program aplikasi dapat mengaksesnya. Jika antrian port penuh maka UDP akan membuang datagram tersebut.

3.1.5 Nomor-nomor Port UDP

Pemakaian port memiliki dua metode : *central authority & dynamic binding*.

Dengan metode *Central Authority*, semua pemakai harus menyetujui adanya pengaturan penggunaan port secara terpusat dan menyetujui juga adanya publikasi list seluruh nomor port yang digunakan. Semua software yang digunakan dibangun berdasarkan list yang ada. Metode ini juga disebut dengan *universal assignment* dan port yang digunakan disebut dengan *well-known port assignment*.

Dalam metode *Dynamic Binding* nomor port tidak dipublikasi secara global. Jika sebuah program membutuhkan port, maka *software* jaringan akan mengambil satu port & menomorinya. Untuk mengetahui nomor port komputer lain dilakukan dengan mengirim *request*, kemudian menunggu *reply* yang berisi nomor port yang digunakan.

3.2 LAYANAN RELIABLE STREAM TRANSPORT

3.2.1 Kebutuhan Layanan Pengiriman Stream

Layanan pengiriman data pada level dibawah level transport seringkali mengabaikan kehandalan suatu layanan. Kehilangan paket, kegagalan perangkat jaringan, ketidakterurutan data, duplikasi, delay yang besar merupakan sebab dibutuhkannya layanan transmisi data yang lebih reliable.

Karakteristik suatu layanan data dikatakan reliable adalah :

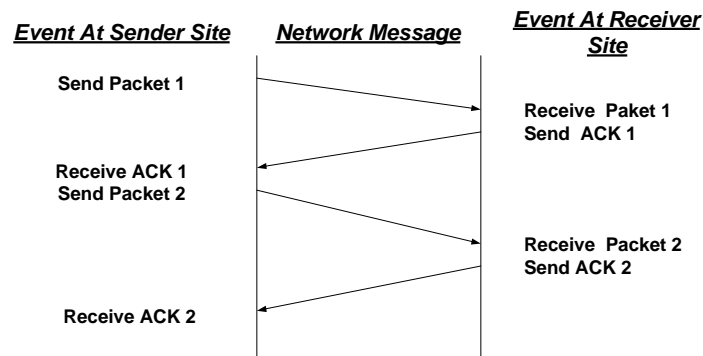
- *stream orientation*
untuk layanan data dalam kapasitas besar, data dianggap sebagai aliran (stream) bit yang dibagi-bagi menjadi oktet atau byte. aliran byte yang masuk ke mesin tujuan memiliki urutan yang sama pada waktu pengiriman.

- ***virtual circuit connection***
sebelum pengiriman data dilakukan diawali dengan komunikasi antara pengirim & tujuan mengenai pembentukan sebuah koneksi.
- ***buffered transfer***
program aplikasi mengirimkan aliran data ke virtual circuit yang terbentuk dengan cara mengirim oktet-oktet data ke *protocol software* secara berulang. Ketika mentransfer data setiap aplikasi dapat menggunakan beragam ukuran oktet. Protocol software akan mengirimkan data tersebut sesuai urutan aslinya. Protocol ini dapat dengan bebas membagi paket menjadi bagian-bagian independen untuk ditransferkan.
- ***Unstructured Stream***
- ***Full Duplex Connection***
Koneksi yang disediakan oleh layanan stream TCP/IP memungkinkan transfer secara bersamaan di dua arah (full duplex)

3.2.2 Reliabilitas

Salah satu metode untuk menerapkan layanan transfer reliable di atas layanan tidak reliable adalah dengan teknik ***Positive Acknowledgement with Retransmission***.

Metode yang dijalankan adalah metode pengiriman jawaban untuk pemberitahuan bahwa data yang dikirimkan telah diterima. Pengirim akan menyimpan 1 record dari setiap paket yang dikirimkannya & menunggu datangnya ack sebelum mengirim paket yang berikutnya. Ketika pengiriman dilakukan pengirim juga mengaktifkan suatu timer untuk menandai masa aktif suatu paket. Timer ini juga digunakan untuk batasan bilamana perlu dilakukan retransmisi.



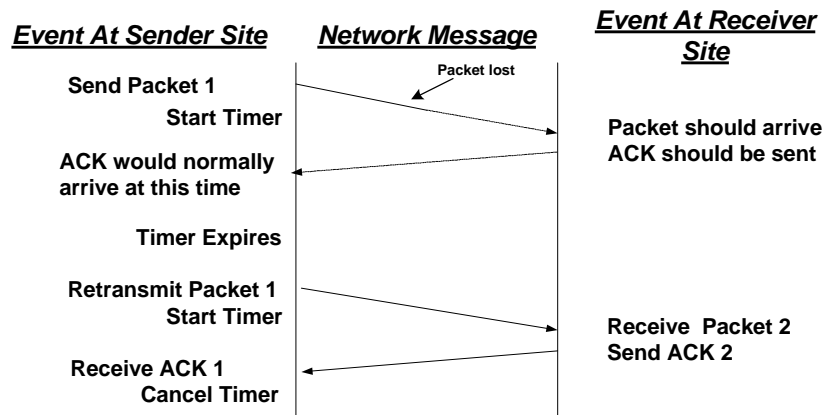
gambar Positive ACK with retransmission

Ket :

Sisi kanan & kiri menunjukkan jenis-jenis *event* yang terjadi baik di sisi pengirim (sender) maupun disisi penerima (receiver). Sedangkan garis diagonal menunjukkan transfer pesan melalui jaringan.

Dalam suatu kasus dimana suatu paket lost, maka dilakukan retransmisi pada saat timer expires (masih aktif paket habis). Mekanismenya dapat dilihat pada gambar dibawah ini

:



gambar mekanisme retransmisi

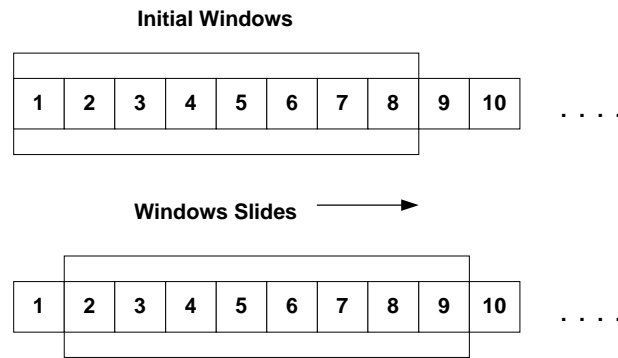
Masalah yang mungkin muncul akibat dari paket yang terlambat adalah adanya duplikasi data. Untuk menanganinya diperlukan perhatian pada aspek paket & ack-nya. Duplikasi dapat dideteksi oleh protocol dengan memeriksa sequence number yang terdapat pada tiap paket & meminta penerima untuk mengingat sequence number yang telah diterima.

Untuk menghindari kesalahan intepretasi antara paket tertunda atau ack yang terduplikasi, protocol ack positif akan mengirim kembali sequence number dalam paket ack.

3.2.3 Sliding Windows

Untuk meningkatkan efisiensi pengiriman sehingga jaringan tidak sering idle digunakan sliding windows.

Sliding windows adalah metode pengiriman banyak paket dalam satu ukuran window secara kontinyu tanpa harus menunggu kehadiran ack.



gambar Proses Sliding Window dengan 8 paket

Pada gambar di atas window yang dipakai berukuran delapan paket. Window akan bergeser sebanyak jumlah ack yang diterimanya yang menandakan boleh dikirimnya paket yang menjadi anggota dalam windows tersebut. Retransmisi juga akan dilakukan untuk paket-paket yang tidak dijawab dengan ack.

Performansi protocol sliding window bergantung pada ukuran window yang digunakan & kecepatan jaringan dalam menerima paket.

3.2.4 Transmission Control Protocol

TCP merupakan protokol layer transport yang dapat menyediakan layanan *connection-oriented, reliable & byte stream orientation* .

Protokol TCP menyediakan :

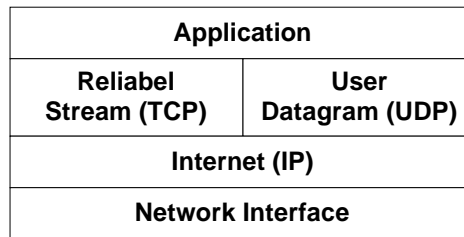
- spesifikasi format paket data & ack yang dipertukarkan antara 2 host untuk penyediaan layanan yang reliable
- spesifikasi metode yang digunakan oleh software TCP untuk pengenalan aplikasi tujuan pada suatu mesin
- metode komunikasi antar host jika terjadi duplikasi atau lost packet.
- metode komunikasi antar host untuk proses transfer data.

Pada dokumentasi protocol yang ada, hanya mendiskusikan mengenai operasi yang dapat disediakan oleh TCP, tetapi tidak menspesifikasikan prosedur program aplikasi yang

terlibat dalam suatu proses operasi. Alasan dari tidak ditentukannya interface program aplikasi adalah untuk mendukung fleksibilitas.

3.2.5 Port, Connection & End Point

Dalam skema dibawah ini protokol TCP terletak di atas protokol IP. Protokol ini mengijinkan adanya banyak program aplikasi pada suatu mesin untuk berkomunikasi secara bersamaan. Dimana dilakukan pula demultiplex trafik dari banyak aplikasi program tersebut.



gambar konsep layer UDP & TCP di atas IP

Untuk mengidentifikasi mesin tujuan digunakan nomor port. (berupa bilangan integer dalam skala kecil).

Protocol Software akan menempatkan setiap datagram yang baru datang sebagai antrian pada port-port yang ada sesuai dengan jenis aplikasinya.

Konsep identifikasi layanan yang dipegang oleh TCP adalah **connection abstraction**, yaitu identifikasi **virtual circuit connection**. Satu koneksi terdiri dari sepasang end point. End point adalah pasangan integer (host,port), dimana host menunjukkan alamat IP yang digunakan & port menunjukkan nomor port TCP dalam suatu host.

Contoh :

End point (128.10.2.3,25) :

- TCP port =25
- Yang berada di host dengan alamat IP 128.10.2.3

Koneksi antara host (18.26.0.36) di MIT ke host (128.10.2.3) di Purdue University :

(18.26.0.36,1069) dan (128.10.2.3,25)

secara bersamaan terdapat koneksi dari (123.9.0.32) ke host di Purdue :

(123.9.0.32,1184) dan (128.10.2.3,53)

koneksi dengan penggunaan end-point juga dapat dilakukan misal (128.2.254.139) di CMU ke Purdue :
(128.2.254.139,1184) and (128.0.2.3,53)

3.2.6 Pasive & Active Opens

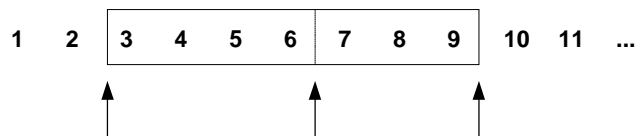
TCP adalah protocol *connection oriented* yang membutuhkan persetujuan antara 2 *end-points*. Satu *end-points* membentuk fungsi *passive open* dengan mengotrol operating system & mengindikasikan bahwa akan menerima koneksi yang akan terbentuk. Pada waktu itu OS akan menandai suatu port untuk suatu *end koneksi*.

Program aplikasi pada *end-system* yang lain (pasangannya) kemudian mengontak sistem operasi nya menggunakan *request active open* untuk membangun suatu koneksi. Sekali koneksi terbentuk, maka proses pengiriman data dapat berlangsung.

3.2.7 Segments, Streams & Sequence Number

Pada transmisi data level TCP, stream data dianggap sebagai urutan oktet-oktet atau byte-byte yang dibagi-bagi menjadi banyak *segment*. Biasanya setiap *segment* terdapat di dalam satu datagram IP.

Seperti telah disebutkan sebelumnya, mekanisme sliding window dapat digunakan untuk efisiensi transmisi & flow control. Mekanisme ini bekerja pada level oktet, bukan pada level segment atau paket. Oktet dari stream data di nomori secara terurut & pengirim menyimpan 3 pointer di setiap koneksi yang terbentuk.



Contoh Sliding Window dengan Oktet

Pointer pertama menandai bagian kiri sliding window yang memisahkan antara oktet yang sedang dikirim dan yang belum dikirim. Pointer ke dua manandai batas kanan yang merupakan tanda untuk oktet tertinggi dalam urutan yang dapat dikirim sebelum ack

datang. Pointer ketiga menandai batas dalam yang memisahkan antara oktet yang telah tengah dikirim dengan yang tengah belum dikirim.

Protocol software akan mengirim semua oktet tanpa delay sehingga batasan di window akan berpindah secara cepat dari kiri ke kanan.

3.2.8 Flow Control & Variabel Window Size

Penggunaan window dapat divariasikan ukurannya. Informasi mengenai ukuran oktet yang masih dapat diterima oleh *receiver buffer* disebut dengan *window advertisement* yang dikirim melalui paket ack.

Sumber dapat mengirim paket dalam ukuran window yang lebih besar sebagai respon terhadap peningkatan *window advertisement*, atau tidak mengirim lagi paket diluar batas window yang baru.

Mekanisme variasi ukuran window ini dapat digunakan untuk mengontrol aliran paket. Jika buffer penerima hampir/ sudah penuh maka dapat diminta pengurangan pengiriman atau bahkan penghentian pengiriman. Untuk jaringan internet yang memiliki kecepatan & kapasitas yang berbeda-beda, mekanisme flow control ini sangat diperlukan.

3.2.9 Format Segment TCP

Unit data yang ditransferkan antara dua software TCP di dua host yang berbeda disebut *segment*. Segment ini dipertukarkan untuk kepentingan *pembangunan koneksi, transfer data, pengiriman ack, pemberitahuan ukuran window & penutupan koneksi*.

SOURCE PORT			DESTINATION PORT		
SEQUENCE NUMBER					
ACKNOWLEDGEMENT NUMBER					
HLEN	RESERVED	CODE BITS	WINDOW		
CHECKSUM			URGENT POINTER		
OPTIONS (IF ANY)				PADDING	
DATA					
...					

Gambar Format Segment TCP

SOURCE PORT & DESTINATION PORT berisi nomor port TCP yang mengidentifikasi program aplikasi pada end koneksi. **SEQUENCE NUMBER**

mengidentifikasi posisi oktet data dalam segmen yang dikirimkan. **ACKNOWLEDGEMENT NUMBER** mengidentifikasi nomor oktet berikutnya yang dapat diterima oleh pengirim. **HLEN** menandakan panjang header segment yang dihitung pada kelipatan 32.

Beberapa segment memiliki isi informasi yang beragam seperti membawa ack, membawa data atau untuk membangun/menutup suatu koneksi. Software TCP menggunakan 6 bit pada field **CODE BITS** untuk menentukan tujuan & isi dari suatu segment. Reperesentasi dari penggunaannya dapat dilihat pada tabel dibawah ini :

Bit (Left to right)	Meaning if bit set to 1
URG	<i>Urgent pointer field is valid</i>
ACK	<i>Acknowledgement field is valid</i>
PSH	<i>This segment requests a push</i>
RST	<i>Reset the connection</i>
SYN	<i>Synchronize sequence numbers</i>
FIN	Sender has reached end of its byte stream

Gambar Field-field CODE BITS dalam header TCP

Pengirim akan menentukan ukuran data yang dapat diterima setiap pengiriman segment dengan mengisi field **WINDOW** dengan panjang 32 bit.

Tidak semua segment yang dipertukarkan antara dua sistem memiliki ukuran yang sama, namun demikian persetujuan antara ukuran maksimum segment harus ada. Permintaan ukuran segment maksimum ini dapat dilakukan dengan menggunakan field **OPTIONS**. Pemilihan maksimum segment dapat mempengaruhi performansi. Terlalu besar akan menyebabkan adanya fragmentasi, dan terlalu kecil menyebabkan utilitas jaringan rendah.