

TCP and Congestion Control (Day 1)

Yoshifumi Nishida
nishida@csl.sony.co.jp
Sony Computer Science Labs, Inc

1

Today's Lecture

- Part1: TCP concept
- Part2: TCP detailed mechanisms
- Part3: Tools for TCP

2

TCP concept

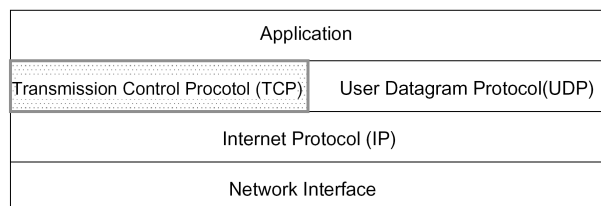
- What is TCP?
 - Introduction of TCP
- Concept of the TCP service
 - Byte Stream service with no structure
 - Full Duplex
 - Buffered Transfer
 - Connection Oriented
 - Reliable Service

3

What is TCP? (1)

- Transmission Control Protocol
 - A Protocol in transport layer
 - Provide communication from one application to another.
- Provides reliable data transfer service
 - UDP provides unreliable data transfer service.

Protocol Hierarchy



4

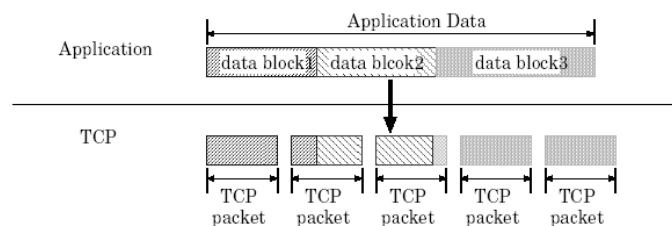
What is TCP? (2)

- TCP provides reliable data transfer service.
 - Why is reliable data transfer necessary?
 - Some applications need to send data to receivers as it is.
 - E-mail, File transfer..
- UDP provides unreliable data transfer service.
 - Why is unreliable data transfer necessary?
 - Some applications need to send data as quick as possible.
 - Video transfer..

5

Byte Stream Service

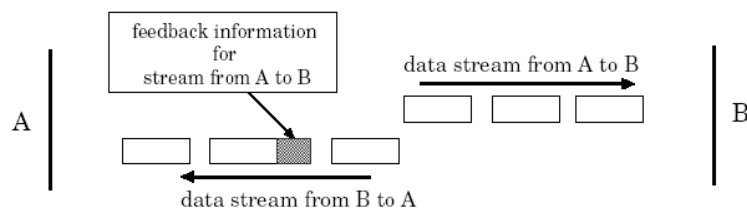
- Data handled by TCP has no structure.
 - TCP regards data as bit stream.
 - TCP splits data from application into multiple packets
 - The size of packets are arbitrarily determined by TCP
 - TCP guesses appropriate size of packets for each communication path.



6

Full Duplex Communication

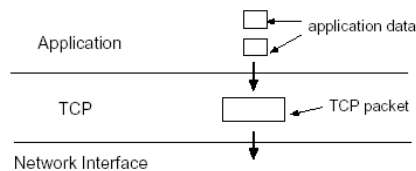
- TCP allows concurrent data transfer in both directions
 - Data can be transmitted while receiving
 - Use of "Piggyback"
 - Data packets can convey feedback information in the opposite direction



7

Buffered Data Transfer

- TCP may delay data transfer.
 - If TCP has not received enough data from the application
 - TCP tries to aggregate data as much as possible.

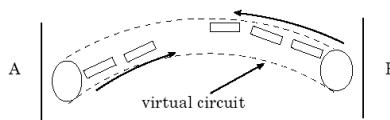


- If the receiver is slow
 - TCP tries not to overflow the receiver -> Flow Control
- If the network is congested
 - TCP tries to prevent congestion in networks -> Congestion Control
- Applications that use TCP do not know when data will be sent.

8

Connection Oriented

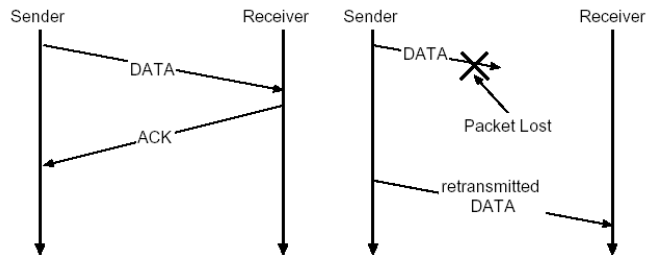
- Negotiation must take place between communicating nodes before data transfer.
 - Estimates the capability of each node.
 - How much data can be received at the same time?
 - Which optional functions are supported?
- Establishes virtual circuit between communicating nodes.
 - All that applications have to do is to hand data to TCP.



9

Reliable Service (1)

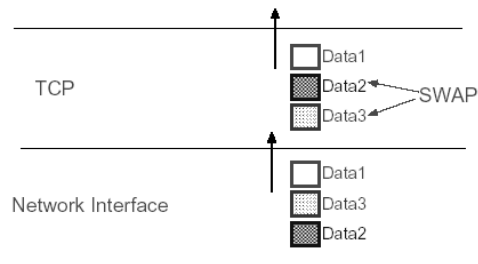
- Acknowledgment and Retransmission
 - TCP sends back "Acknowledgment packet" (ACK) as it receives data packets
 - Notifies the sender of the packet arrival
 - Sender retransmits a packet if ACK does not arrive within a certain period.



10

Reliable Service (2)

- Packet Re-ordering
 - TCP reassembles out of order packets



- Robust Checksum
 - Used for data integrity check
 - IP provides 16 bit checksum for IP header.
 - TCP provides 32 bit checksum for TCP header and TCP data.

11

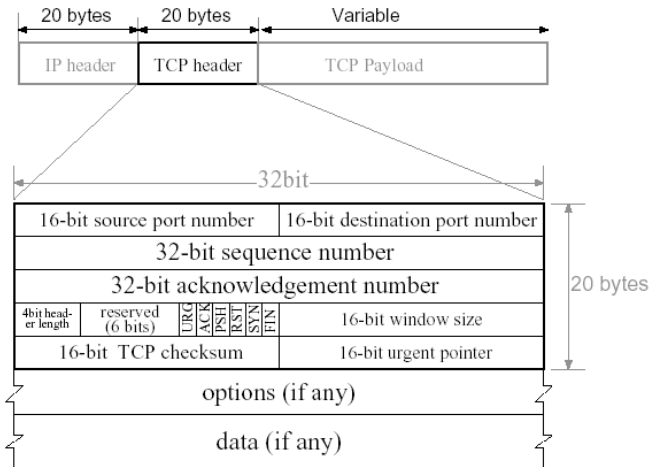
Detailed mechanisms of TCP

- TCP packet format
- Connection setup and termination
 - Mechanisms related to beginning and end of connection
- Data Transmission Control
 - Mechanisms related to data transfer

12

TCP Header Format (1)

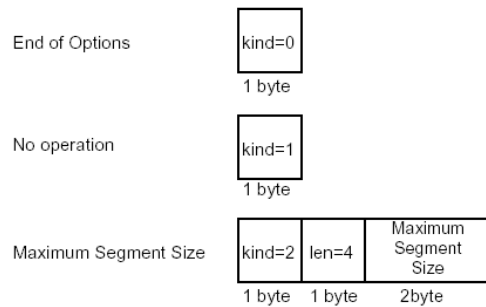
Example of TCP packet



13

TCP Header Format (2)

- TCP options:
 - Provide additional function for TCP.
- Option formats:
 - All options have "kind" field.
 - Options longer than 1 byte have "length" field.



14

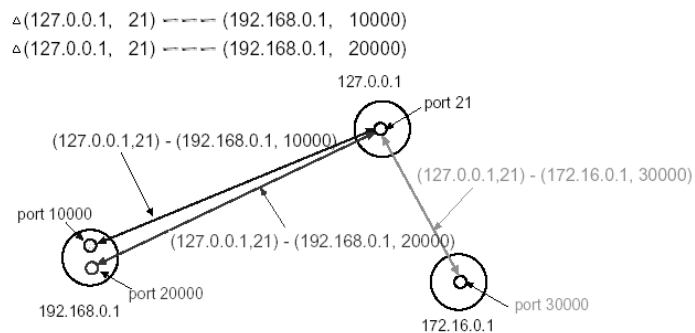
TCP Header Format (3)

- Header format has some limitations.
 - Sequence Number
 - From 0 to $(2^{32} - 1)$
 - Port Number
 - From 0 to $(2^{16} - 1)$
 - Window Size
 - From 0 to $(2^{16} - 1)$ bytes.
 - Header length
 - From 0 to 15 in 32bit words.
 - Max TCP header size is limited to 60 bytes
 - Since normal header has 20 bytes, TCP can have another 40 bytes for options.
 - Reserved bit
 - 6 bits are reserved for future extension

15

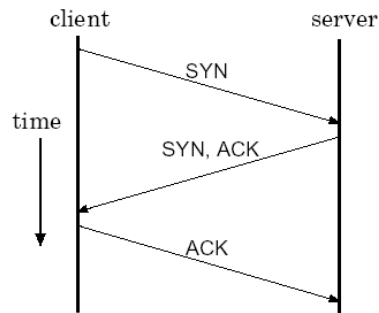
Connection Endpoint

- TCP uses IP address and Port number to identify the endpoint.
- Connection can be expressed by a pair of the endpoints.
 - Same port number can be shared by multiple connections on the same machine.
 - example of the connections:



Connection Establishment (1)

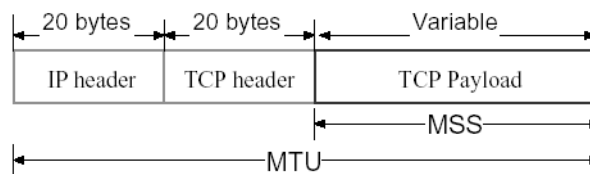
- 3way handshake
 - Exchange Synchronize Request (SYN) and Acknowledgment (ACK)
 - Receiver sends "Connection Request" and "Acknowledgment for Connection Request" simultaneously (using piggybacking)
 - A Connection is established by exchanging 3 packets



17

Connection Establishment (2)

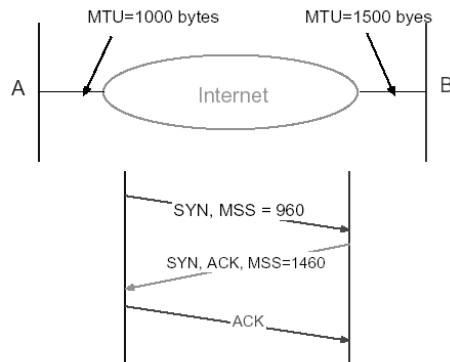
- During connection setup, both sides exchange "MSS" information with TCP option.
 - MSS: Maximum Segment Size
 - Largest payload size that TCP can send for this connection.
 - Usually, MSS is calculated by Maximum Transmission Unit (MTU) - 40 bytes.



18

Connection Establishment (3)

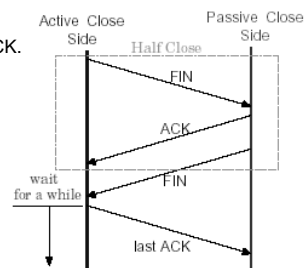
- An example of MSS negotiation
 - In this example, both sides use 960 bytes as MSS.



19

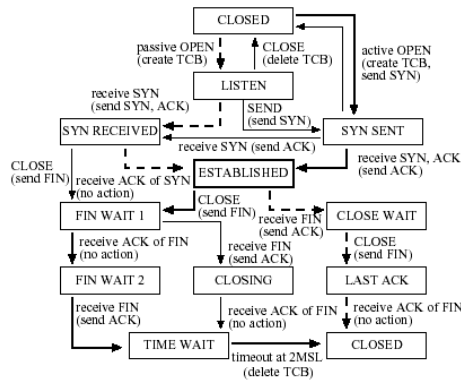
Connection Termination

- Half Close
 - A data flow can only be closed in one direction at a time.
 - So, a Connection is terminated by exchanging 4 packets
- 2 communicating nodes play a different role
 - active close side
 - Sends first FIN(indicate finish of the data flow).
 - Active close side waits for a while after it sends last ACK.
 - passive close side
 - Sends ACK for FIN and then sends second FIN.
 - Passive close side waits for last ACK.



TCP status and Operation (1)

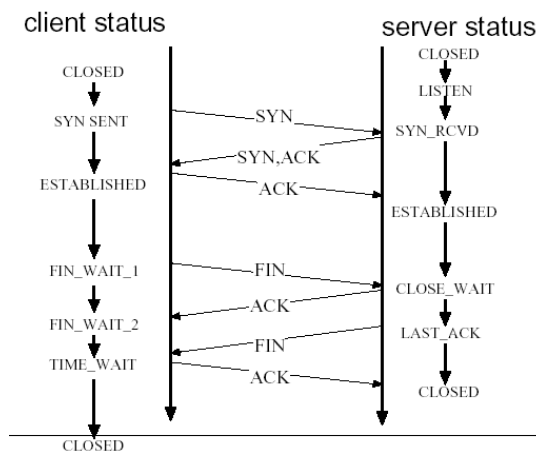
- TCP has 11 states
 - Operation of TCP can be explained in a finite machine model.



21

TCP status and Operation (2)

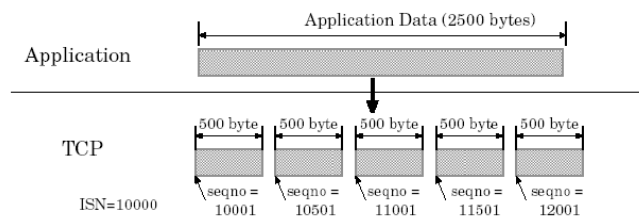
- Example of state transition
 - Typical case



22

Sequence Number (1)

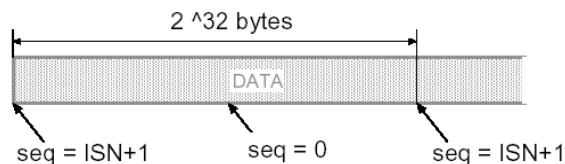
- Indicates the position of the data in the packets
 - Every byte is sequenced
 - Used for re-ordering packets and finding lost packets
- Consists of Initial Sequence Number (ISN) and position in the application data
 - ISN is randomly assigned for every TCP connection
 - For security reasons, ISN should not be easy to be guessed



23

Sequence Number (2)

- Max sequence number is $2^{32} - 1$.
 - If application data exceeds $2^{32} - 1$, the sequence number wraps back around to 0.

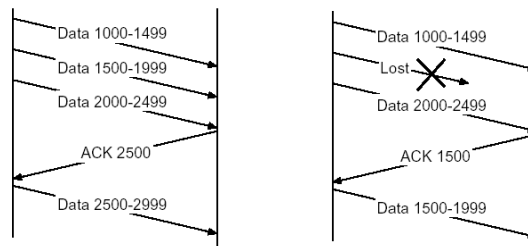


- Notice
 - SYN, and FIN packets also consume 1 sequence number, although they do not include any data.

24

Acknowledgment method

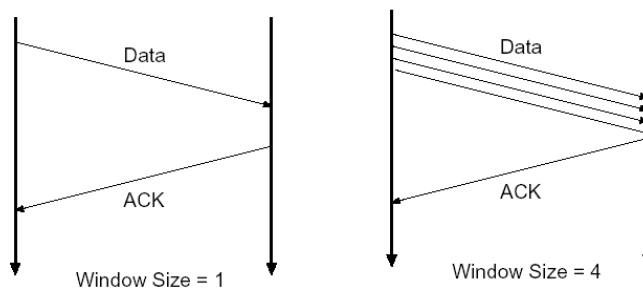
- Cumulative acknowledgment
 - An acknowledgment of sequence number X indicates that all bytes up to but not including X have been received.
 - An acknowledgment packet can acknowledge multiple packet arrivals.



25

Sliding Window (1)

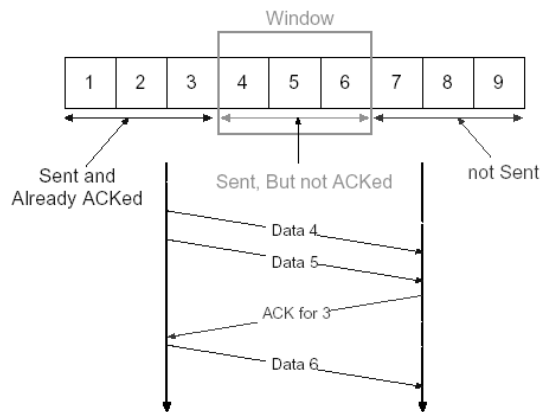
- TCP uses "Window Size" for data transmission.
 - Window Size:
 - The number of data packet which can be sent without waiting for ACKs



26

Sliding Window (2)

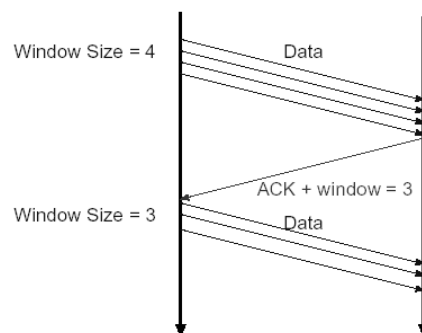
- Concept of the sliding window scheme



27

Sliding Window (3)

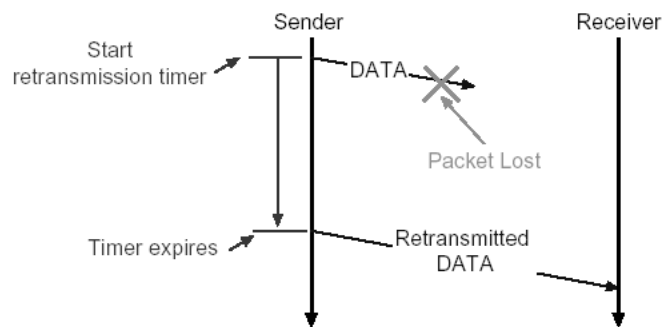
- Sliding Window Scheme can be used for flow-control
 - Flow-Control:
 - Receiver can advertise desirable window size to the sender
 - Avoid overflow situation



28

Packet Retransmission (1)

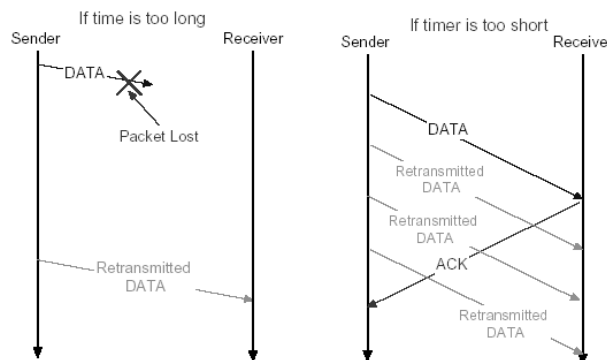
- TCP retransmits the lost packets
 - Guarantee high reliability
- How to find the lost packets?
 - Timeout of retransmit timer
 - Sender starts a retransmission timer after a transmitting packet.
 - If the timer expires before an ACK arrives, the packet will be retransmitted.



29

Packet Retransmission (2)

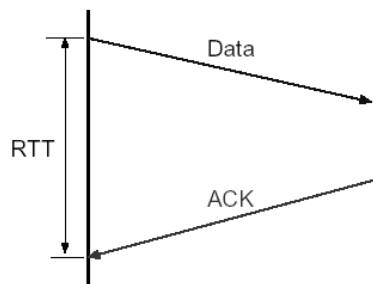
- But how do we set retransmit timer?
 - If timer is too long -> causes performance degradation
 - If timer is too short -> causes unnecessary retransmission



30

Packet Retransmission (3)

- How do we set retransmission timeout (RTO) value?
 - RTO is calculated from RTT(Round Trip Time)
 - Appropriate time for retransmission is very different from each communication path.
 - RTT:
 - The time received an ACK minus the time a data was sent



31

RTO calculation (1)

- TCP monitors the performance of each connection and computes appropriate RTO.
 - Adapt to various communication media (from analog modem to high-speed LAN..)
 - Adapt to network condition. (congestion..)
- Old algorithm for RTO
 - TCP uses "weighted Average" of measured RTTs as estimated RTT.
 - $SRTT = \alpha \times SRTT + (1 - \alpha) \times \text{measured RTT}$
 - $0 < \alpha < 1$ (usually set to 0.8 or 0.9)
 - TCP computes timeout from SRTT
 - $RTO = \beta \times SRTT$
 - $\beta > 1$ (usually set to 2)

32

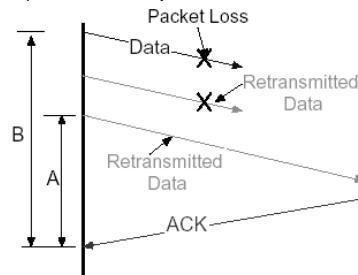
RTO calculation (2)

- Problem of the old calculation algorithm
 - RTT fluctuates widely in large scale networks.
 - Since old algorithm uses "Average", it can't keep up with fluctuation.
- New calculation algorithm
 - Err = measured RTT – SRTT
 - SRTT = SRTT + g * Err (g is usually set to 0.125)
 - D = D + h * (|Err| - D) (h is usually set to 0.25)
 - RTO = SRTT + 4 * D
 - Based on "Deviation"
 - "D" represents smoothed Mean Deviation
 - Mean Deviation is good approximation of the Standard Deviation
 - By using Mean Deviation, we can avoid computing square root.

33

Karn's algorithm (1)

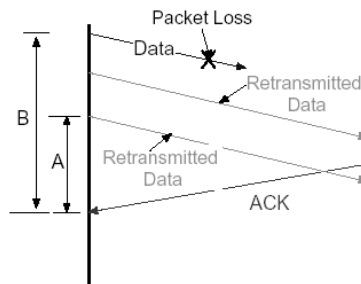
- Improve accuracy of the RTT measurement.
 - RTT measurement with packet loss
 - Duration A: use the most recent retransmission for RTT measurement.
 - Duration B: use the original transmission for RTT measurement.
 - Which duration is suitable for RTT measurement?
 - example 1:
 - We should use "duration A" as RTT in this case.
 - But, this assumption is not always correct.



34

Karn's algorithm (2)

- RTT measurement with packet loss
 - example 2:
 - We cannot use "duration A" as RTT in this case!
 - So, what should we do?



35

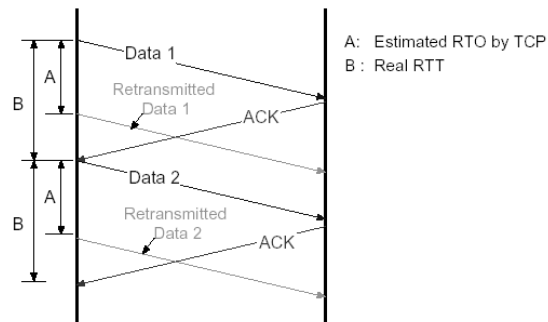
Karn's algorithm (3)

- Karn's algorithm
 - Rule 1: Ignore measured RTT for retransmitted packets.
 - Remove ambiguity from RTT measurements.
 - Rule 2: RTO should be doubled after retransmission.
 - This is called "Exponential Back-off"

36

Karn's algorithm (4)

- Why is Rule 2 necessary?
 - When RTO is smaller than Real RTT..
 - If only Rule 1 is applied, TCP will use A as RTO forever.
 - All packets will be retransmitted.
 - No RTT measurement happens.



37

Urgent Data Transmission (1)

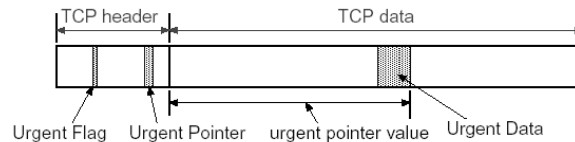
- TCP has simple mechanism to specify data priority.
 - High-priority data called "urgent data"
- What is urgent data used for?
 - When an application wants to abort data transfer, it uses the urgent data and tells peer to abort.
 - Interactive applications like rlogin, telnet use this.

38

Urgent Data Transmission (2)

- How to process urgent data?
 - Data Sender side:
 - Set the urgent flag and the urgent pointer in the TCP header.
 - Data Receiver side:
 - (When the Receiver receives the TCP packet with urgent data.)
 - Notify application that we have urgent data.
 - Stay in urgent mode until application finish to read urgent data.

TCP packet with urgent data



39

Urgent Data Transmission (3)

- Some limitation of the urgent data design.
 - TCP only knows the endpoint of the urgent data.
 - TCP does not know the start point of the urgent data or the length of the urgent data.
 - TCP only knows if application has finished reading the urgent data.
 - TCP does not know if application has started to read urgent data.
 - When urgent data is sent to the receiver consecutively...
 - The urgent pointer is overwritten whether application has finished reading the first data or not.
 - Receiver cannot maintain multiple urgent data at the same time.

40

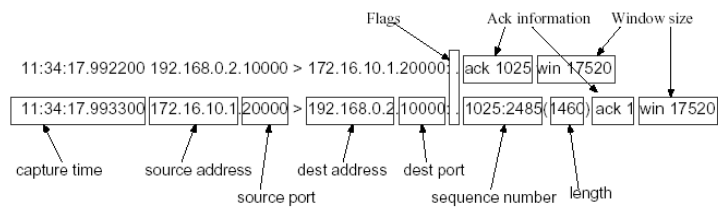
Tools for TCP

- tcpdump
- ethereal
- tcptrace
- tcpillust

41

tcpdump

- Most common packet capture tool.
– <http://www.tcpdump.org/>
- Example output for tcpdump



42

ethereal

- A more graphical tool compare to tcpdump.
 - <http://www.ethereal.com/>

43

tcptrace (1)

- Used to analyze output of various packet capture tools
 - elapse time for data transfer.
 - amount of transmitted and received bytes.
 - number of retransmitted packets.
 - round trip time variation..
- Support output of many tools.
 - tcpdump, snoop, etherpeek, HP Net Matrix
- Can generate graphical outputs.
 - needs xplot.

44

tcptrace (2)

- **Basic usage:**
% tcptrace dumpfile
- **Get graphical output**
% tcptrace -lr dumpfile

45

tcpillust

- Visualize tcp connection interaction
- <http://www.csl.sony.co.jp/person/nishida/tcpillust.html>

46

Today's Summary

- TCP provides reliability to IP communication.
 - Retransmits packets if packets are lost.
 - Delays packets transmission in response to the condition of the receivers or networks.
- TCP is designed for various situations.
 - Adapt to slow links or high-speed links or congested links.

47

Preview of the Next Lecture

- Part 1: TCP Issues and Solutions
- Part 2: Congestion Control
- Part 3: Simulating TCP

48