

# Transport Layer Concepts and Design Issues



**Dr. Rahul Banerjee**

Computer Science and Information Systems Group

Birla Institute of Technology and Science

Pilani - 333 031, INDIA

E-mail: [rahul@bits-pilani.ac.in](mailto:rahul@bits-pilani.ac.in)

Home Page: <http://www.bits-pilani.ac.in/~rahul/>

# Interaction Goals:



- Understanding of the **Transport Layer**
- A look at **various Design Issues**
- A comparative study of various **Transport Layer Protocols**
- **Crash Recovery Schemes**
- An Understanding of **Current Industry Practices** and **Evolving Trends**

# Transport Layer: What is it?



- Transport Layer is a layer of the Network Architecture that is primarily concerned with getting TPDUs from the upper layer (usually Application Layer) and delivering it to the same layer at the intended destination node (through the underlying Network Layer). The reverse is also the responsibility of this layer.

# Transport Layer Responsibilities



- It primarily deals with:
  - Accepting APDU from the Application Layer through the SAP
  - Processing these APDU
  - Deciding transport connection requirements (for further transmitting this DU after encapsulating it within a TPDU)
  - Passing this packet through the SAP to the lower layer (NL)

# TL Responsibilities ...



- It also deals with ...
  - Accepting TPDU from the lower layer through the SAP
  - Processing the TPDU
  - Removing the encapsulation and passing the APDU through the SAP to the upper layer (Application Layer)
  - Providing support for connection-oriented / connectionless services as the case may be (depending upon the protocol stack and need)
  - Provide diagnostic support for network monitoring, configuration, management and trouble-shooting at the Transport Layer or higher layer.

# Related Terms:



- Transport
- Transport Service
- Transport Service Primitives
- Transport Service Provider
- Transport Service User
- Transport Service Access Point (TSAP)
- Transport Entity
- Transport Connections
- TSAP versus NSAP

# Generic Transport Service Primitives: A Possible Set



- Create / Identify and Assign / Bind
- Listen / Wait
- Accept
- Connect
- Send / Transmit
- Receive
- Disconnect / Close / Terminate

# Generic Transport Service Primitives: A Less Flexible Sub-set



- Listen
- Connect
- Send
- Receive
- Disconnect

# The TCP Header Structure



Source Port Number (16-bit)	Destination Port Number (16-bit)
Sequence Number (32-bit)	
Acknowledgement Number (32-bit)	
HLEN + Reserved + Code	Sliding Window
TCP Checksum (16-bit)	Urgent Pointer (16-bit)
Options (if present) + Padding (if needed)	
Payload Data	

# About the TCP & UDP Ports



- **TCP and UDP Ports are 16-bit numbers.**
- **They are of three types:**
  - **Well-known Ports (0-1023: Controlled by the IANA),**
  - **Registered Ports (1024-49159) and**
  - **Ephemeral / Dynamic Ports (49152-65535). (RFC 1700 shows a list suggested initially)**
- **FTP over TCP uses 21 whereas TFTP over UDP uses 69 for instance.**
- **X-Windows Server uses 6000-6063 Registered Ports.**
- **For BSD, the scheme is as under:**
  - **WKP: 1-1023**
  - **Reserved Ports: 1024-5000**
  - **Severs: 5001-65535 (no privilege)**
- **For Solaris, these ranges vary again!**

# Inside the TCP



- **TCP stands for Transmission Control Protocol.** (RFC 793 by John Postel)
- It offers a **Connection-oriented Transport Service.**
- **Assumes the underlying IP-subnet as unreliable and therefore takes care of reliability, flow control and reordering of data units as per requirement.**
- **Sends data to the IP Layer in MSS-sized blocks or smaller, after prefixing a TCP header to each such segment.**
- **Default value of the MSS is 536, in case the peer at the other end does not specify a smaller value to be used with it.**
- **MSS is normally of lesser than or equal to the size of the MTU (for IPv4: 40, for IPv6: 60 ).**

# Inside the TCP (Contd.)



- **TCP requires that a TCP Client establishes a Full-Duplex connection with a TCP Server (before any real data could be exchanged between them). After the data exchange is over, this connection has to be explicitly Terminated.**
- **As the TCP provides a reliable service, it expects an ACK to be received for the data transmitted by an TCP-entity (Client or Server).**
- **If the ACK does not arrive within a Time-out period, it retransmits the data and waits for a longer period of time to receive an ACK.**
- **Even if after a certain number of such attempts the data cannot be successfully transmitted, it gives up further attempts and informs the Application Layer. (Intermediate failures are not reported to the Application, however!)**

# Inside the TCP (Contd.)



- The maximum period for such retransmission-attempts and associated wait-periods for a single data unit, put together, may be anywhere between 4 Minutes to 10 Minutes, depending upon the TCP implementation and Stack Configuration.
- Round-Trip Time (RTT) is automatically, dynamically, computed between a Client-Server pair by a routine internal to the TCP implementation.
- RTTs are always more for WANs than for LANs.
- TCP recognizes byte-boundaries and is thus a byte-stream-oriented protocol.
- As it provides each of its Segments a Serial Number, reordering, rejection of duplicate segments etc. becomes possible.
- It uses Sliding Window Protocol for the purpose of data transmission / reception / flow-control.

# The 3-Way Handshake in TCP



- **TCP requires a Three-Way Handshake for the Connection-Establishment.**
- This is called so since a minimum of three data-units need to be exchanged between a TCP Client and a TCP Server for establishing a TCP connection.
- These packets may be **SYN-I-Seq-No (C-to-S)**, **SYN-Ini-Seq-No (S-to-C)** on which **ACK-I+1-Seq-No** piggybacks (S-to-C) and lastly, **ACK-Ini+1-Seq-No (C-to-S)**.
- SYN stands for Synchronize segment. It takes just 1-byte of Sequence Number Space.
- In a similar way, Connection-Termination takes four data-units. It takes place using **FIN (Final Segment)** and associated **ACK**. Both sides send one FIN and one ACK to each-other, in this case.
- SYN contains TCP options of **MSS**, **Sliding Window Scaling** (Left-Shifting by 0 to 14 bits allows window-sizes of 64K to 1 GB) [RFC 1323 by Jacobson et al], **Timestamp** (for High-speed connections) etc.

# The TCP/IP Access Points



- In a TCP/IP network, services offered by any layer can only be used through parameter / data passing through the various Service Access Points (SAPs) located on Layer-boundaries.
- TSAPs and NSAPs are two major examples of SAPs located at the AL-TCP/UDP Layer-boundary and TCP/UDP-IP Layer-boundary respectively.
  - A typical example of an NSAP is an IP address.
  - A typical example of a TSAP is an IP address + a 16-bit integer called as Port Number.
- Port Numbers permit unique identification of several simultaneous processes using TCP / UDP.
- IETF's RFC 1700 lists select IANA-suggested Port Number Assignments.

# Transport Service Primitives: The Berkeley Sockets Set for the TCP



- Socket
- Bind
- Listen
- Accept
- Connect
- Send
- Receive
- Close

# The Transport Service Access Point (TSAP) & the Network Service Access Point (NSAP)



- In an IP network / internetwork, the NSAP refers to the IP Address of the node.
- In a TCP / UDP over IP stack, the TSAP refers to the pair: {IP Address, Local Port Number} *It is at the TSAP, at which a peer process listens / contacts.*
- *Both TSAPs and NSAPs could be one or more per node / host.*

# QoS Considerations in the TL As Used During the Option Negotiation Process



- Required Priority of Service
- Ceiling for Residual Error Ratio
- Maximum Acceptable Connection Establishment & Transit Delays
- Minimum Acceptable Throughput
- Maximum Acceptable Probability of Connection-Establishment-Failure
- Maximum Acceptable TL-initiated Abnormal Termination of Connection
- Security and Protection Specifications

# The UDP Header Structure



Source Port Number (16-bit)	Destination Port Number (16-bit)
Message Length (16-bit)	UDP Checksum (16-bit: Optional)
Payload Data	

# About the TCP & UDP Ports



- **TCP and UDP Ports are 16-bit numbers.**
- **They are of three types:**
  - **Well-known Ports (0-1023: Controlled by the IANA),**
  - **Registered Ports (1024-49159) and**
  - **Ephemeral / Dynamic Ports (49152-65535). (RFC 1700 shows a list suggested initially)**
- **FTP over TCP uses 21 whereas TFTP over UDP uses 69 for instance.**
- **X-Windows Server uses 6000-6063 Registered Ports.**
- **For BSD, the scheme is as under:**
  - **WKP: 1-1023**
  - **Reserved Ports: 1024-5000**
  - **Severs: 5001-65535 (no privilege)**
- **For Solaris, these ranges vary again!**

# Inside the Transport Layer Protocols: The RTP Perspective



# Inside the Transport Layer Protocols: The RTCP Perspective



# Inside the Transport Layer Protocols: The RTSP Perspective



# Inside the Transport Layer Protocols: The STCP Perspective



# The SIP Perspective



# Types of Crashes



- Classification One:
  - Host Crashes
    - Client Crashes
    - Server Crashes
  - Subnet-Device Crashes
    - Router Crashes
    - Bridge Crashes
    - Repeater Crashes
  - Link Crashes
- Classification Two:
  - Temporary Crashes / Non-Fatal Crashes
  - Permanent / Fatal Crashes
  - Intermittent / Unanticipated Repeated Crashes

# Crash Recovery Strategies in TCP / IP Networks and Internetworks



- In any connectionless packet delivery system, including the IP based systems, there is always a possibility, however small, that a Transport Protocol Data Unit (TPDU) may be lost on its way to the destination Host's Transport Layer.
- This simply means that, a TL mechanism must exist in such situations that could continually monitor the transfer / receipt status, identify any missing TPDU and take corrective measures for getting it, if protocol so enforces / permits.

# Crash Recovery Strategies in TCP / IP Networks and Internetworks ...



- These are the very mechanisms that may handle Subnet Crash Recovery, where so possible.
- However, the real challenge lies in Recovering from the Host Crashes. This is primarily because of the fact that certain amount of data loss (apart from the connection-loss, in case of Connection-oriented Transport Services) in the crashed host is bound to create problems in the recovery process. A little thought over the complexity forced by the commonly used Sliding Window Protocol in such a situation would make the magnitude of the problem clear.

# Client Crash Recovery Strategies



- Consider a case in which a Client Host which is downloading a large file from a Server that is remotely located crashes temporarily and then comes back.
- Clearly, three things would happen:
  - Any User / Control Data currently in the local buffer which is yet to be written to the disk / storage medium shall be lost.
  - Any Control Signal, say an ACK, generated but yet not transmitted shall be lost.
  - The data under processing at the time of crash, at the local host shall be lost.

# Client Crash Recovery Strategies ...



- Depending upon the protocol in use these situations may lead to different inferences, and hence may require different recovery policies.
- Let us assume that our protocol requires that the downloading Client sends an ACK to the sender only after successfully writing to the disk (in case of a simple Stop-and-Wait protocol). In such a case, if the data is written to the disk but before the ACK could be sent, the host crashes, it would mean the loss of the ACK as well as the Connection. Let us see what could happen when this host comes back to operation!

# Client Crash Recovery Strategies ...



- When, in this case, the Client Host comes back, it has already lost the transport connection and any data in its memory.
- Clearly, unless there exists a semi-permanent status-of-progress record that is updated after every successful operation, the host may be unable to track exactly what it was doing and the associated details.
- The first question is that who should maintain such a record and where?
  - One possibility is to query the Server about the status, but this is a non-starter given the sheer resource requirements servers shall be forced to have if they are to retain all such status data even if for a short duration of say five minutes.
  - Moreover, how do the Client know which Server to contact, if there is no record at its disposal locally.

# Client Crash Recovery Strategies ...



- These points suggest that it may be preferable to expect each Client Host to maintain these records and utilize them in the event of a crash.
- Assuming that our Client Host in question does have such a record, the next question is what steps should it follow for Recovery and Resumption of the file-download.
  - One possibility is that soon after the Client Host comes back, it seeks to learn about its network status and thereafter, retrieves its stored records which have an 'Incomplete so far' type of tag / flag.
  - The operation, previously interrupted due to the crash, could be resumed only after a fresh connection is requested to the designated Server and is obtained.
  - Once such a connection is setup, the Client may inform the Server that it needs the file-download to resume from a specific point onward (as per the local record) instead of a fresh full-retransmission.

# Server Crash Recovery Strategies



- Just like the cases we considered for Client Host Crash and a possible Recovery, many possible situations may arise, though very rarely, in which a Server Host crashes during a operation, say involving download or upload of a large file.
- One possibility could be to require the Server (after it quickly reboots) sending a Broadcast Message to all other visible Hosts querying about the respective transfer-status (like if they were interacting with it when it crashed and if yes, what was the pre-crash status?).

# Server Crash Recovery Strategies ...



- Clearly, this scheme assumes that the Crash was of very short duration and that various Host Clients have yet not 'closed' their open connections.
- Depending upon whether 'First Acknowledge then Write to the Output Stream' or 'First Write then Acknowledge' this scheme could invite Loss of a TPDU or generate a Duplicate TPDU respectively.
- Clearly, this suggests that both the Client and the Server need to maintain their respective records for a possible recovery; but still there may be situations which may be difficult be handled at the TL itself.
- This gives an important message loud and clear! The higher layer cannot be ignorant of a crash if an acceptable recovery is to be done.

# Recommended Readings:



- S. Keshav: An Engineering Approach to Computer Networking, AWL, 1997.
- A. S. Tanenbaum: Computer Networks, Fourth Edition, PHI, 2003.
- U. D. Black: Computer Networks, Second Edition, PHI, 1993.
- D. Bertsekas and R. Gallager: Computer Networks, Second Edition, PHI, 1992.
- G. R. McClain (Ed.): Handbook of Networking and Connectivity, AP Professional (Academic Press), 1994.

# Recommended Readings ...



- D. Comer: Internetworking with TCP / IP , Vol..-1, PHI, 1995.
- D. Comer & D. L. Stevens: Internetworking with TCP /IP, Vol.. 2-3, PHI,1994, 1993.
- W. Buchanan: Advanced Data Communication and Networks, Chapman & Hall, London, 1997.
- Uyles D. Black: TCP / IP & Related Protocols, Second Edition, McGraw-Hill, N. Y., 1995.