HO CHI MINH CITY UNIVERSITY OF SCIENCE
FACULTY OF ELECTRONICS AND TELECOMMUNICATIONS
DEPARTMENT OF TELECOMMUNICATIONS AND NETWORKS

COURSE

COMPUTER NETWORKS

Chapter 06

UDP TRANSPORT

Reference: Peter L Dordal, "An Introduction to Computer Networks," Jul 26, 2019

Lecturer: Nguyen Viet Ha, Ph.D. Email: nvha@fetel.hcmus.edu.vn

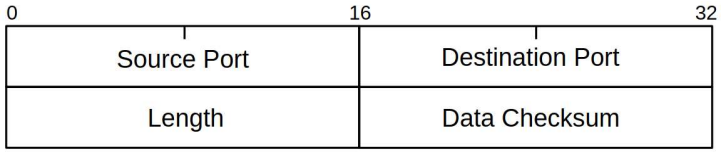
1 User Datagram Protocol

1. User Datagram Protocol – UDP

❖UDP provides simple datagram delivery to remote sockets (<host,port> pairs).

➢UDP as "almost a null protocol".

❖The two features it adds beyond the IP layer are port numbers and a checksum. UDP Header:



➢Port number:

○An application can now connect to an individual server ***process***, rather than simply to a host.

1. User Datagram Protocol – UDP

❖UDP is **unreliable**, in that there is no UDP-layer attempt at *timeouts*, *acknowledgment* and *retransmission*.

❖UDP is **unconnected** (or **stateless**)

➢Deliver packets without *negotiation*.

❖The **UDP checksum** covers the UDP header, the UDP data and also the source and destination IP addresses.

❖Using in "local" transport or real-time transport.

1. User Datagram Protocol – UDP

❖Sometimes UDP is used simply because it allows new or experimental protocols to run entirely as user-space applications; no kernel updates are required.

❖**QUIC (Quick UDP Internet Connections)**:

- ➢From Google
- ➢Support the HTTP protocol.
- ➢Advantage:
  - ○Supporting multiplexed streams in a single connection.
  - ○ A lost packet blocks its own stream until it is retransmitted, but the other streams can continue without waiting.

1. User Datagram Protocol – UDP

❖Sometimes UDP is used simply because it allows new or experimental protocols to run entirely as user-space applications; no kernel updates are required.

❖**QUIC**:

- ➢From Google
- ➢Support the HTTP protocol.
- ➢Supporting multiplexed streams in a single connection.
  - ○A lost packet blocks its own stream until it is retransmitted, but the other streams can continue without waiting.

1. User Datagram Protocol – UDP

❖**QUIC**:

- ➢Disadvantage:
  - ○Nonstandard programming interface, but note that Google can achieve widespread web utilization of QUIC simply by distributing the client side in its Chrome browser.
  - ○Breaks the "social contract" that everyone should use TCP.

1. User Datagram Protocol – UDP

❖**QUIC**:

- ➢QUIC eliminates the initial RTT needed for setting up a connection.
  - ○Allowing data delivery with the very first packet.
  - ○Requires a recent previous connection.
- ➢QUIC provides support for advanced congestion control.
  - ○Currently using TCP CUBIC (15.15 TCP CUBIC).
- ➢One downside of QUIC is its nonstandard programming interface, but note that Google can achieve widespread web utilization of QUIC simply by distributing the client side in its Chrome browser.

1. User Datagram Protocol – UDP

❖**DCCP (Datagram Congestion Control Protocol)**:

- Transport protocol build atop UDP.
- Preserving UDP's fundamental tolerance to packet loss.
- Adds a number of TCP-like features to UDP:
  - Connection setup and teardown.
  - TCP-like congestion management.
- DCCP data packets are numbered.
  - Delivered to the application in order.
- DCCP also adds acknowledgments to UDP, but in a specialized form primarily for congestion control.
  - No retransmission.

1. User Datagram Protocol – UDP

❖**DCCP (Datagram Congestion Control Protocol)**:

- DCCP is specifically intended to run in in the operating-system kernel, rather than in user space.
  - This is because the ECN congestion-feedback mechanism (14.8.3 Explicit Congestion Notification (ECN)) requires setting flag bits in the IP header, and most kernels do not allow user-space applications to do this.

1. User Datagram Protocol – UDP

❖**UDP Simplex-Talk**:

- Early standard examples for socket programming.
  - The client side reads lines of text from the user's terminal and sends them over the network to the server.
  - The server then displays them on its terminal.
- The server does not acknowledge anything.
- The server must select a port number, which with the server's IP address will form the socket address to which clients connect.

1. User Datagram Protocol – UDP

❖**UDP Simplex-Talk**:

- On the **server side**, simplex-talk must do the following:
  - Ask for a designated port number.
  - Create a **socket**, the sending/receiving endpoint.
  - **Bind** the socket to the socket address, if this is not done at the point of socket creation.
  - Receive packets sent to the socket.
  - For each packet received, print its sender and its content.
- The **client side** has a similar list:
  - **Look up** the server's IP address, using DNS.
  - Create an "anonymous" socket; we don't care what the client's port number is.
  - Read a line from the terminal, and send it to the socket address *<server_IP,port>*.

2 Trivial File Transport Protocol

2. Trivial File Transport Protocol (TFTP)

- A protocol based on UDP.
- Supports file transfers in both directions.
- No support a mechanism for authentication.
  - Any requestable files are available to anyone.
- Because TFTP is UDP-based, and clients can be implemented very compactly.
- Well-suited to:
  - The downloading of startup files to very compact systems, including diskless systems.

2. Trivial File Transport Protocol (TFTP)

- It uses stop-and-wait.
  - Uses a fixed timeout interval.
- Offers limited security.
  - TFTP is typically confined to internal use within a LAN.

2. Trivial File Transport Protocol (TFTP)

- Has five packet types:
  - ***Read ReQuest (RRQ)*** containing the filename and a text/binary indication.
  - ***Write ReQuest (WRQ)***.
  - ***Data***, containing a 16-bit block number and up to 512 bytes of data; begins at 1.
    - All blocks of data contain 512 bytes except the final block.
    - Data < 512 is identified as the final block.
      - If the file size was divisible by 512, the final block will contain 0 bytes of data.
    - *Denote: Data[N]:* Packet with the N^{th} block of data.

2. Trivial File Transport Protocol (TFTP)

- Has five packet types:
  - ***ACK***, containing a 16-bit block number of the block being acknowledged
    - *Denote:* ACK[N] acknowledges Data[N].
  - ***Error***, for certain designated errors. All errors other than "Unknown Transfer ID" are cause for sender termination.

2. Trivial File Transport Protocol (TFTP)

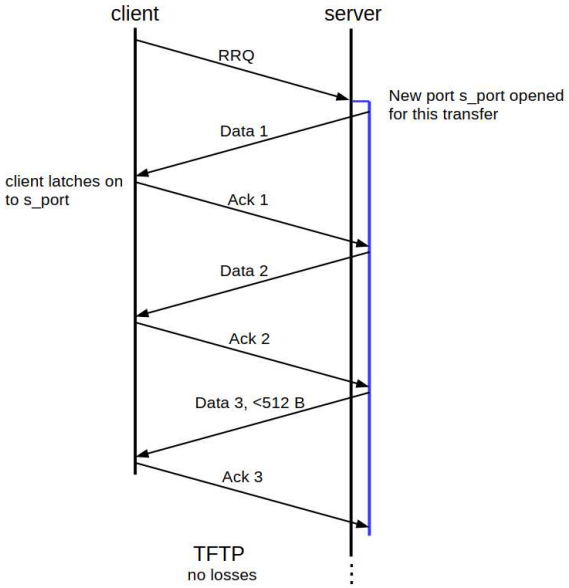
- Because TFTP uses UDP (as opposed to TCP) it must take care of packetization itself, and thus must choose a block size small enough to avoid fragmentation.
- The TFTP server listens on UDP port **69** for arriving RRQ packets.
- For each RRQ requesting a valid file, TFTP server implementations almost always create a separate process (or thread) to handle the transfer.
  - That child process will then obtain an entirely new UDP port, which will be used for all further interaction with the client, at least for this particular transfer.

2. Trivial File Transport Protocol (TFTP)

- TFTP file requests typically proceed as follows:
  1. The client sends a RRQ to server port *69*.
  2. The server creates a child process, which obtains a new port, *s_port*, from the operating system.
  3. The server child process sends *Data[1]* from *s_port*.
  4. The client receives *Data[1]*, and thus learns the value of s_port. The client will verify that each future *Data[N]* arrives from this same port.
  5. The client sends *ACK[1]* (and all future ACKs) to the server's *s_port*.
  6. The server child process sends *Data[2]*, etc, each time waiting for the client *ACK[N]* before sending *Data[N+1]*.
  7. The transfer process stops when the server sends its final block, of size less than 512 bytes, and the client sends the corresponding ACK.

2. Trivial File Transport Protocol (TFTP)

- TFTP file requests typically proceed as follows:



3 Fundamental Transport Issues

❖Old Duplicate Packets

❖Lost Final ACK

❖Duplicated Connection Request

3. Fundamental Transport Issues

- **Reboots**

QA

Lecturer: Nguyen Viet Ha, Ph.D.
Ho Chi Minh City University of Science
Faculty of Electronics and Communications
Department of Telecommunication and Networks
Email: nvha@fetel.hcmus.edu.vn