Protocol Design

Assignment 2: uft.2

Stress test your uft… (1)

1. Write a small program that can generate arbitrary UDP packets
   - Use it to generate data and control packets to send to your uft client and/or server
     - Packets should be somewhat close to real ones, yet random
     - Some suggestions: right total size but arbitrary contents, inconsistent field values (e.g., mismatch of packet length and length field), undefined values for selected fields, strange file names, …
   - Observe and document what happens
   - Suggest reasonable fixes
     - To your protocol specification
     - To your implementation
   - Implement selected ones that can be done with reasonable effort

Stress test your uft… (2)

2. Examine your protocol design with respect to two senders (accidentally or maliciously) sending a file of the same name to the same receiver address at the same time
   - Document your observations
   - What type of protocol refinements (if any) would be needed to fix this?
   - If you like, try this out with your small random packet generator
     - Sniff an existing session (e.g., one transmitting a large file at low rate)
     - Add a second sender sending a different file and check the result
3. Temporarily(!) try to turn your client into a DoS tool
   - Modify the requests to the server to direct the file transmission to an unsuspecting third party target
   - Document whether or not this is possible and why/why not.
   - How would you fix this?

Part 2: UDP file transfer v2: uft.2

- Extend your uft to support an adaptive congestion control scheme
  - Ack-clocking, TFRC-based, or something else deemed useful
  - Motivate and document your choice
- Your scheme should scale
  - At least from 10 kbit/s to 1 Mbit/s data rate
  - How about different delays?
- You obviously need to modify the semantics of “-b”
  - Suggest something useful for this parameter

Testing: udppipe

`udppipe -l [lhost:]lport –c [chost:]cport –b <bitrate> [-d <delay>]`

-l: transport address to receive UDP packets on from first uft peer; in the opposite direction, packets are sent to the address they were received from

-c: transport address to send UDP packets to (the other uft peer needs to transmit its responses to the address taken from recvfrom ())

-b: bit rate specified in kbit/s

-d: delay specified in milliseconds (default: none)