Some Findings from Assignment 1

Wide variety of specifications:

- From 1 to 10 pages
- More protocol spec vs. more implementation spec
- More or less complete (at a first glance)

Encodings

- JSON (text-based)
  - Base64-encoded data
- Unknown
- 3 x Box notation (binary)
  - Single bit to distinguish between data and control packet
  - Packet type field
- Text-based for control, binary for packets?
- RFC 822-style with 8-bit encoding for the data part
- Binary 64-bit words + scrambling to avoid deterministic bit errors
- HTTP-style + RFC 822

Protocol Operation (1)

- Connection setup
  - Explicit establishment via some handshake mechanism
    - Two-way, three-way, four-way, cookies against DoS
- File transmission
  - Various forms of checksums (e.g., MD5, HMACs)
  - Sequence numbers
  - Data + ACK (cumulative, selective ACKs)
  - Data + NACK + a final ACK
  - Dynamic RTO calculation
- Flow control
  - Explicit window size indication
  - Fixed window (negotiated at session setup)
  - Window size derived from delay x bandwidth product

Protocol Operation (2)

- Completion
  - Explicit end signaling + confirmation
  - Implicit server-side detection leads to confirmation
  - Plain shutdown (and hope)?
- Parallel upload
  - Transparent to the protocol

Some Observations on Possible Constraints

- Limited sizes (filename < 255 characters)
- Manual mapping: media type -> binary constant
  - Need to keep up to date
- Sometimes many options
  - Is there a common baseline?
- Did you think about sequence number wrap around?

Protocol Design

Assignment 2:

1. Solution analysis
2. Stress tests
3. adaptive fip

Reminder: Group Info Needed

- Send one email per group in exactly the following format (one line per group member)
  "Last name:First name:ID:email address"
  Mustermann:Erika:12345Z:erikam@example.com
- Just about two groups (out of nine!) got this right!

1. Solution Analysis

- Take a look at someone else's protocol from assignment 1
- Write down your observations (high level perspective)
  - Is the design spec sufficient to create interoperable implementations?
    - Where is it not? What is missing?
  - Is the protocol spec robust?
  - Do you find errors? (concepts rather than details)
  - What else do you observe?
- Practical matters:
  - We will pair two groups (in one case: three groups)
  - We will send out the design documents to the respective groups
  - You may update each other later on (but CC our course assistants)

2. Attack your implementations

- Analyze your peer group's and your own protocol specifications
  - Which are angles that an attacker could use?
    - To kill the server
    - To launch a DoS attack against a competing journalist
    - …
- Go for it!
  - Challenge your own implementation
  - Challenge your peer group's implementation
- Important: both 1) and 2) are to learn
  - Grading of another solution will not depend on what you say about it
  - You may perform analysis and testing jointly
  - But we want independent submissions

Attacking fip… (1)

a) 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

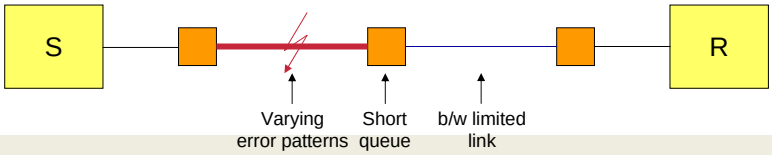
Attacking fip… (2)

b) Exploit knowledge about your / the other protocol
Construct malicious packets to subvert protocol operation
- Of the server
- Of the communication relationship between the server and another client

- What do you learn?
  - 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. Adaptive FIP

- Optimize your file transfer protocol
  - Imagine a cost function like C = Cd + Cv
    Cd = EUR 0.01 / ms delay (reception time – transmission timestamp)
    Cv = EUR 0.01 / byte sent
  - Minimize the cost for a file transfer
    - Calculate the cost for delay on the receiver side
      - Running both sender and receiver on the same machine provides clock sync
      - Return the cost in the last confirmation
    - Calculate the cost for the volume on the sender side and compute the total cost
  - How would your algorithm change if the cost function changes?
- Target environment
  - Error-prone link concatenated with low speed link, high latency



3. Adaptive FIP (2)

- The error pattern will vary in the mid-term
  - Adapt!
- Enhance your protocol for the optimization
  - Retransmissions incur cost in terms of delay (but minimize overhead)
    - Doing only retransmissions will be too costly
  - Proactive repair (e.g., XOR-based) FEC incurs cost in terms of overhead
    - Sending every packet ten times (or 200% FEC) will also be too costly
  - Provide feedback about the observed errors from the receiver to the sender
    - Examples: loss rate, loss patterns
  - Make the sender adapt for the optimization
    - Reduce FEC, increase FEC
- Remember: FEC requires packets of equal length
  - You may need to do some padding
- Implement and test!

Testing: udppipe

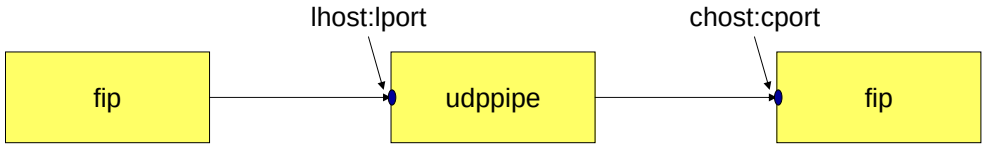
`udppipe -l [lhost:]lport –c [chost:]cport –b <bitrate> [-d <delay>]`

-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)



Only the forward path will be impacted by udppipe modifications.