GS 1380
DAY 4 [Feb 2]
Theophilus Benson
Last Class

helper Code (app.c)

create socket (line 74 - 101)
convert args into byte array (line 102 - 103)
sent array over socket to "other server" (line 106)
wait (line 108)
get data over socket (line 108)
convert data into return value (line 112)

RPC (remote procedure call)

1) make network connections (if keep track)
2) take fun arguments & convert into packets
3) packets into return values
4) failures: what are they? How to react!
proto (protobuf)

server address (ip/port)
func interface protocol type
message type

arg/return ⇔ msgs/packets

func foo (int, short)

extra info

app
RPC stub
RPC framework
Transport

login
RPC stub
RPC framework
Transport

RPC V. REST
GET
PUT/POST

① everyone knows
your func signature
② optimizations

1 JSON rules!!!

GET "Theo Login"

"login"/"username" tab

server ← app + args

failures
RPC framework
HTTP error codes

IDEA
server fails...
Request retry: set a timeout & if a reply isn't received within that time then retry the request.

Suppress duplicates: maintain a “log” of processed requests. Check every request against the log; if you notice a duplicate then drop & ignore this request.

Response replay: maintain a “log” of processed requests & their responses. Check every request against the log; if you notice a duplicate then reply with the cached response.

Response replay
identify duplicates & respond with cache response

V. Duplicate suppression
identify duplicates & ignore them

Both prevent the system from doing duplicate work
**RPC (Semantics)**

<table>
<thead>
<tr>
<th>Retry</th>
<th>Suppress duplicates</th>
<th>Replay (server side) response</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>at-least-once (1 or more)</th>
<th>at-most-once (0 or 1)</th>
<th>exactly once (impossible)</th>
</tr>
</thead>
</table>

- **RPCID**: Doesn't matter how many times you call this function, it only happens once.
- location in file

**Idempotency**: doesn't matter how many times you call this function, only happens once.

- **append**: Not idempotent
  - `append (file, "hello")`
  - `20x`

- **write**: Yay! Idempotent
  - `write (file, 20, "hello")`
  - `20x`
RPC

(1) RPC V REST
(2) RPC Config file (proto)
(3) failures (3 semantics)
(4) Idempotency (at-least-once)
(5) illustrated some of benefits of RPC (write less code)