Remote Procedure Call

- Why have RPC?

- Is it a good model?

- How is it different from local procedure calls?

- Is it hard to implement?!?

What does it look like?!

Client:

```
  z = fn(x, y)
```

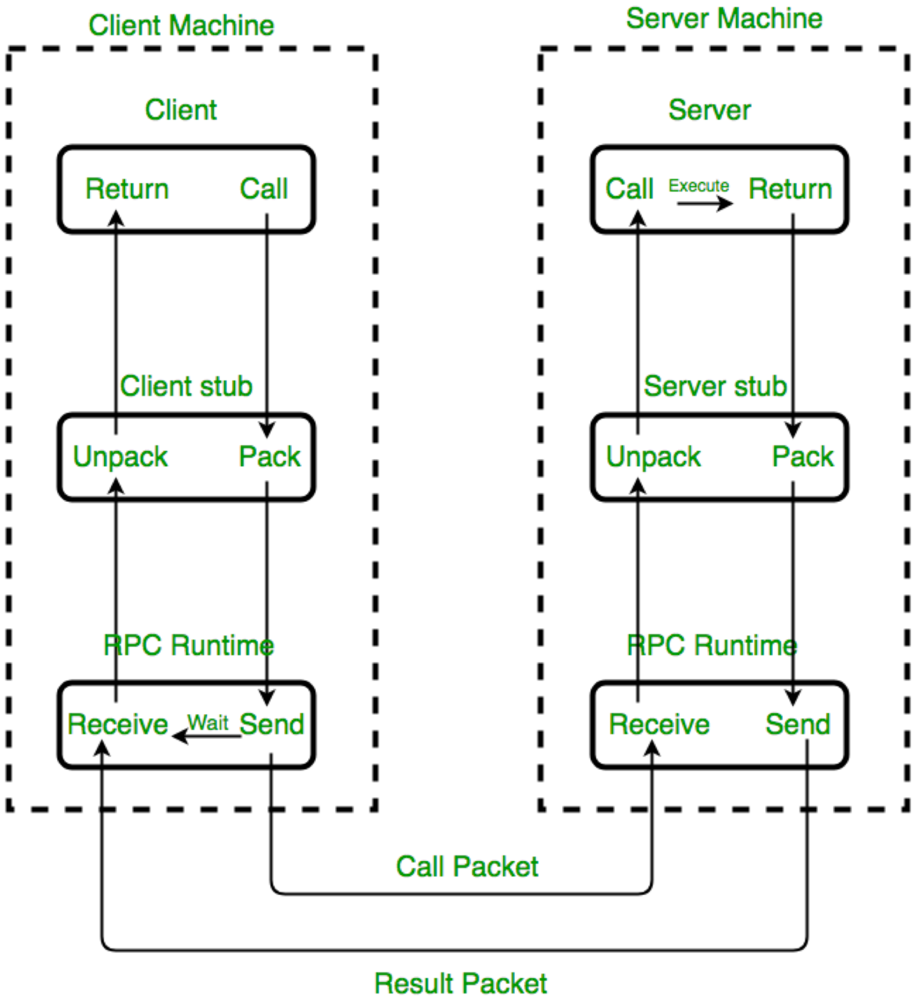
Server:

```
  fn(x, y) {
    compute
    return z
 }
```

RPC aims for this level of transparency

Motivation: even novice programmers can use function call!

RPC architecture: Stubs!



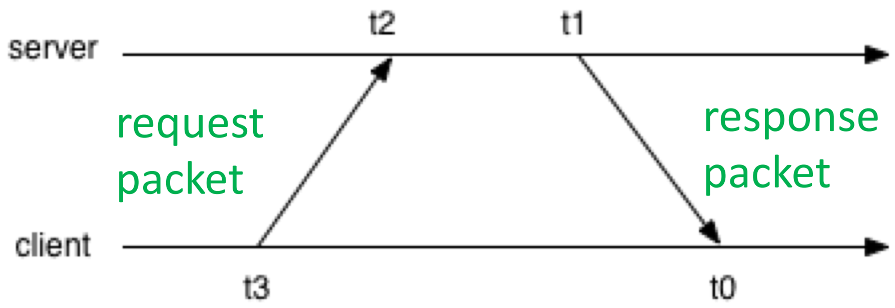
Implementation of RPC mechanism

LET’S PLAY!

- Simple UDP-based time synchronization client
- Test your client by synchronizing against a time server
- By running your synchronization client over a 12-hour-long period, you'll be able to calculate three characteristics:
  - average round-trip time between your client and the server,
  - average packet loss rate between your client and the server, and
  - average drift rate of your client's clock relative to the server's clock

- https://subscription.packtpub.com/book/networking_and_servers/9781786463999/1/ch01lvl1sec22/writing-an-sntp-client

Goes a little something like...



Using these timestamps, the client can calculate the estimated round-trip time (RTT) and estimated clock offset (Θ) as follows:

RTT = (t2 - t3) + (t0 - t1)
Θ = ((t2 - t3) - (t0 - t1)) / 2

Here, Θ is the estimated offset of the server's clock from the client's clock at the midpoint of the interaction.
In other words, the client should add Θ to its local clock in order to be in sync with the server's clock.

Get a start on this!

- Every **10 seconds**, your client should initiate an interaction with the time server.
- So, this means you will be calculating a series of (RTT, Θ) estimates, at most once every 10 seconds.
- Note that it's possible that your client's request packet gets dropped by the Internet, or that the server's response packet gets dropped.
    - If so, that interaction fails, and you won't generate estimates for that interaction. (You'll need to implement some kind of timeout to realize that an interaction has failed.)

Thoughts…

- Because of noise in the network, the (RTT, Θ) estimates will be a little jittery. Your client needs to perform some smoothing over recent estimates in order to figure out what the current clock offset to the server should be.
- Simple way to do the smoothing
  - keep track of the eight most recent successful interactions' estimates
  - for each new successful interaction, pick from those recent eight the estimates with the lowest RTT
  - use the associated Θ as the "smoothed Θ"
- Produce a log file that contains the following:
  - for each successful interaction, record the (RTT, Θ) estimate from that interaction
  - for each successful interaction, record the client local time according to the client hardware, the corrected client local time factoring in the "smoothed Θ", and the "smoothed Θ" itself.
  - note each unsuccessful interaction.

TO DO!

1.Test your time synchronization client. Base it on the code provided by the link!

2.Your client machine probably has the NTP time synchronization software running on it. If possible, disable this, so that your hardware's clock will start to drift!!!

3. Run your client for 12 hours, and collect the resulting log.

4. Analyze the log to calculate the following three numbers:

- the average round-trip time between your client and our server
- the packet loss rate between your client and our server, calculated as the percentage of interactions that failed
- the average rate at which your client's clock is drifting relative to the server's clock, measured in terms of microseconds per second. So, for example, a result of "-75 microseconds per second" means that for each second that ticks on the server's clock, your client's clock only ticks 0.999925 seconds.

Short write up for the lab next week!

Produce a short write up that includes the following

- how did you calculate your client's average clock drift rate?

- what timeout did you pick to detect a failed interaction, and what happens if the server's response packet arrives after that timeout?

- for each successful interaction, calculate the clients' average clock drift rate during the period since the previous successful interaction. Plot a histogram of these "instantaneous" clock drift rate values, and show that plot in your writeup.

  Hypothesize why the histogram is shaped as it is!!!