Client - Server Paradigm

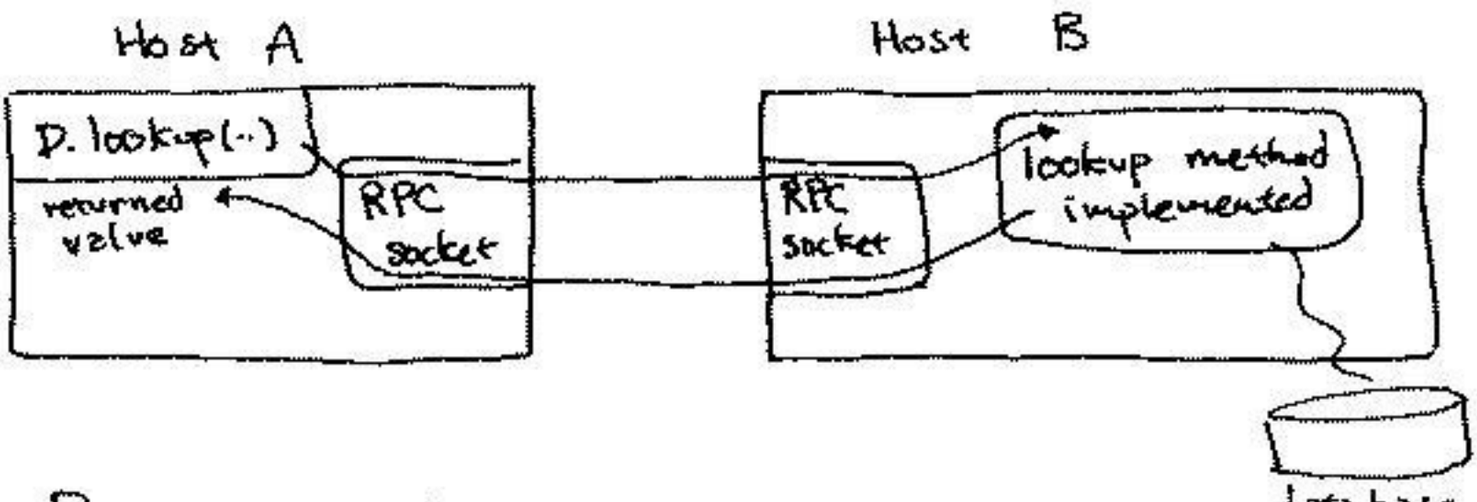
we have seen that TCP and UDP follow this pattern

Remote Procedure Call

an attempt to hide the complexities of client-server networking behind method invocation

Example: `b = D.lookup ( address, city )`

is it a "local" function?
is it really a network-called method?
do we care?



PROBLEM: how can we implement RPC as "automatically" as possible? (Make life easy for programmers using and developing.)

Technical difficulties:

- port numbers, IP addresses
- efficiencies
- dealing with failures (reliability)
- security, authority, heterogeneity

Stateless vs Stateful

Stateless =

RPC call's result is completely specified & determined by its parameters

Examples:

sqrt ( )

dictionary lookup (word)

database_search (key)

Stateful =

RPC call's result is determined by calling parameters and history of previous calls

Examples:

game_move ( )
// "position" of game is remembered by server

iterators

set sum so far

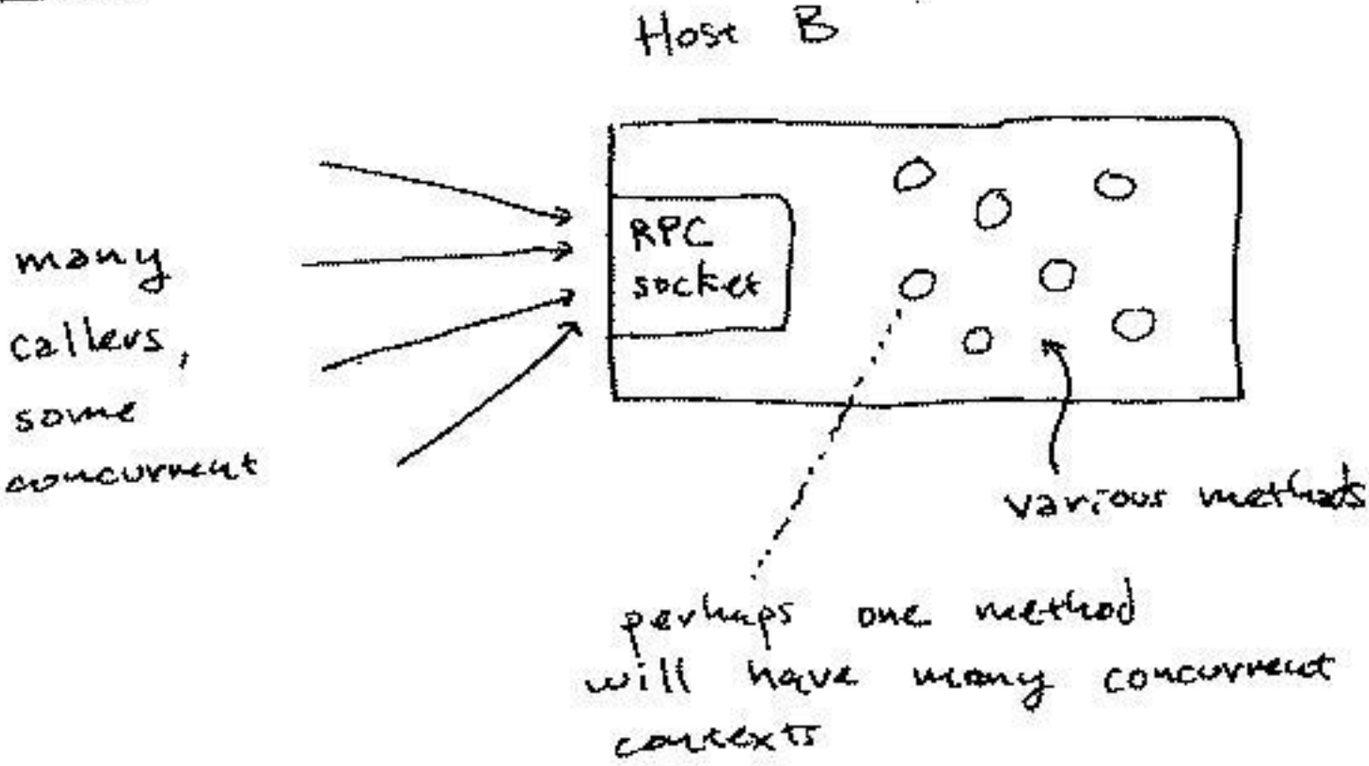
secure methods that require an earlier "logon" with password

all could be stateless if caller would supply complete context (and server would return new context)

↑ context could be very large space in memory -- impractical to send in messages.

Of the two modes, Stateful is far more complicated in implementation.

Consider



- How to multiplex?

  2^{16} ports aren't enough

- How to remember context?

  require client to supply an object reference -- a "handle" -- for the context on Host B

ex: on Host A:

b = D.lookup(addres, city)

Object D has internal variable equal to handle for Host B context.

Other Intricacies:

- Need some way to establish initial context
- Need to "clean up" when context no longer needed (not easy, because clients can fail silently)
- Need a programming environment for developing & testing stateful RPC

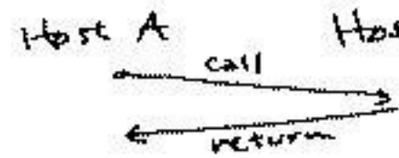
The Latency Problem

when method calls can be remote, they can cause long delays (blocked while waiting for call to return)

▷ Synchronous vs asynchronous RPC

synchronous is the normal way

```
b = D.lookup(...)
```



asynchronous uses "call back"

```
D.lookup(...)
    ⋮
update_from_lookup():
    b = recv(...)
```