Advanced Topics in Operating Systems, CS262

Prof. Eric A. Brewer

Lecture Notes, 10/7/98

RPC in the x-Kernel

I. RPC in the x-Kernel

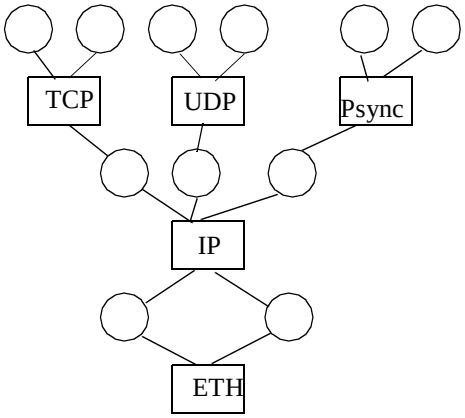
Goal: Designing RPC protocols in a flexible fashion, both to gain performance and to allow code reuse among multiple protocols.

Paper also describes some of the techniques used in the x-Kernel for efficient RPC implementation.

Two design techniques described:

- Virtual protocols: late binding of RPC protocol to low-level message delivery protocol.
- Layered protocols: decompose monolithic RPC protocol into a collection of more primitive protocol building blocks.

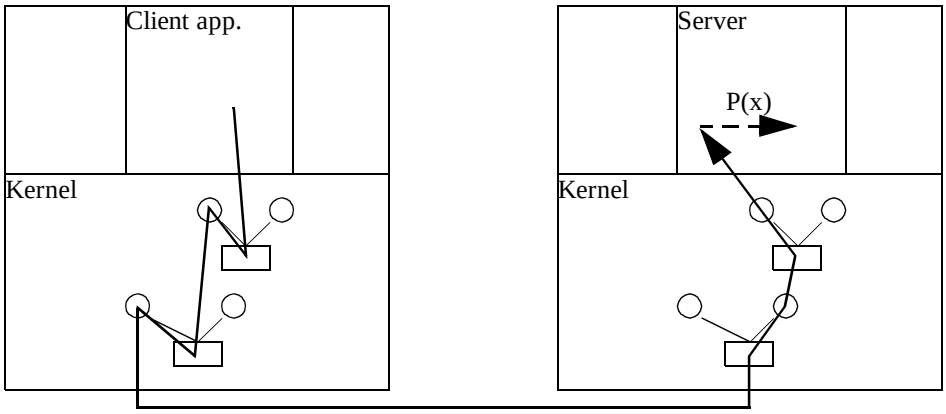
Example x-Kernel configuration:



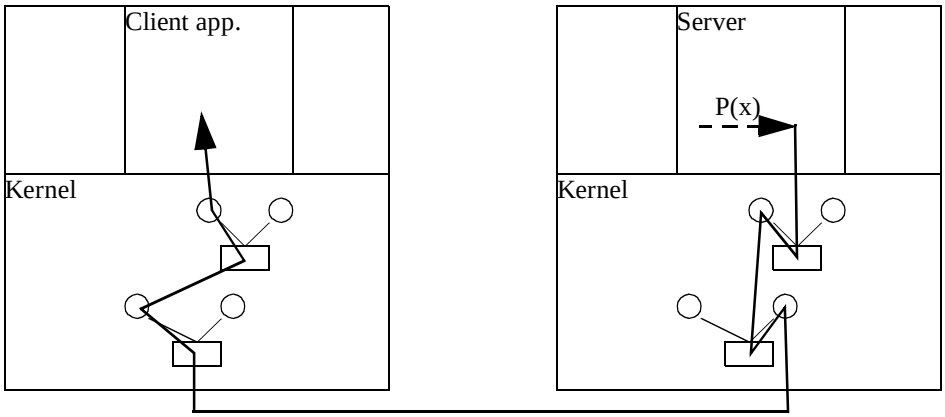
Points to observe:

- {TCP,UDP,Psync} session object per client.
- A client initiating e.g. a new TCP connection would invoke Open().
  A server wanting to e.g. accept incoming Psync RPCs would invoke OpenEnable().
- IP session objects are shared by multiple {TCP,UDP,Psync} clients.
- Two ETH session objects => multiple Ethernet interfaces.

RPC call P(x):



Return:



Notes:

- Client thread enters kernel and does protocol processing for RPC call message on client host.
- Kernel thread is used on the server host to process an incoming message; thread does all the protocol processing and then enters the server's address space, does the procedure call, returns to kernel and does protocol processing to send reply back to client host.
- Kernel thread on client host processes incoming reply message.
- NOTE: If the RPC is synchronous then at some point in the protocol stack the message must be "handed off" to the blocked sender thread.
- "Single-threaded" protocol processing only succeeds IF all resources are available AND RPC call and return each fit in a single network packet.

Virtual protocols:

- Basic idea: special-case the implementation of a protocol according to each instance.
- Requires late binding of protocol stack and uniform protocol interfaces.

- When removing an intermediate protocol layer must be able to map concepts from the higher layer down to the relevant equivalents in a lower layer (e.g. IP address -> Ethernet address).
- Technique doesn't work if inter-protocol dependencies exist that require the intermediate protocol layer to be present (e.g. TCP depends on the presence of the IP header).

Layered protocols:

- Goal: make it easier to design and implement new protocols.
- Basic idea: try to define a basic set of protocol building blocks that can be mixed and matched.
- Useful primarily if you're in the business of designing new communication protocols.

3 key features of the paper:

- Another "specialize the implementation to the specific instance of use" paper. This time the focus is on eliminating unnecessary intermediate network communication protocols when possible.
- Advocates building communication protocols out of smaller "building block" protocols that are layered on top of each other in the hope of being able to reuse these building block protocols when designing and implementing other communication protocols.
- Describes various efficient implementation techniques, such as using a single thread to process an RPC for the common case.

Some flaws:

- Not clear how general their virtual protocol design is. The paper admits having troubles with a number of "common" cases (e.g. TCP over VIP).
- Not clear how important layered protocols are to anyone who isn't a protocol design researcher.

A lesson: If you can special-case your implementation without compromising your interface then it usually pays to do so.