CS 3210: Final Project

Remote Procedure Call

Ning Wang
David Benas
Zongwan Cao

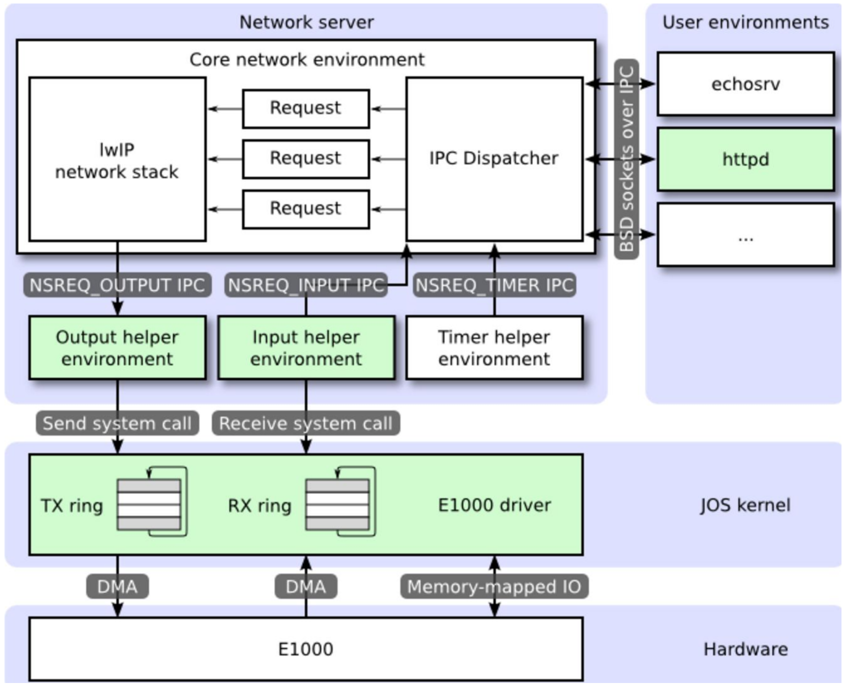
Motivation

- Motivation: socket interface is hard to use
  - Procedure call is a well understood mechanism
  - We extend JOS to support remote procedure call (RPC)
  - It provides a very useful paradigm for communication across network
  - It also makes it easier to build distributed systems
- Wait...currently JOS does not have network support
  - We add a network driver and protocol stack (lwIP) to JOS

Part I: Network Driver

- Implementation overview
    - Initializing transmit circular list
    - Implementing transmitting packets function
    - Initializing receive circular list
    - Implementing receiving packets function
    - Building a web server

Part I: Architecture

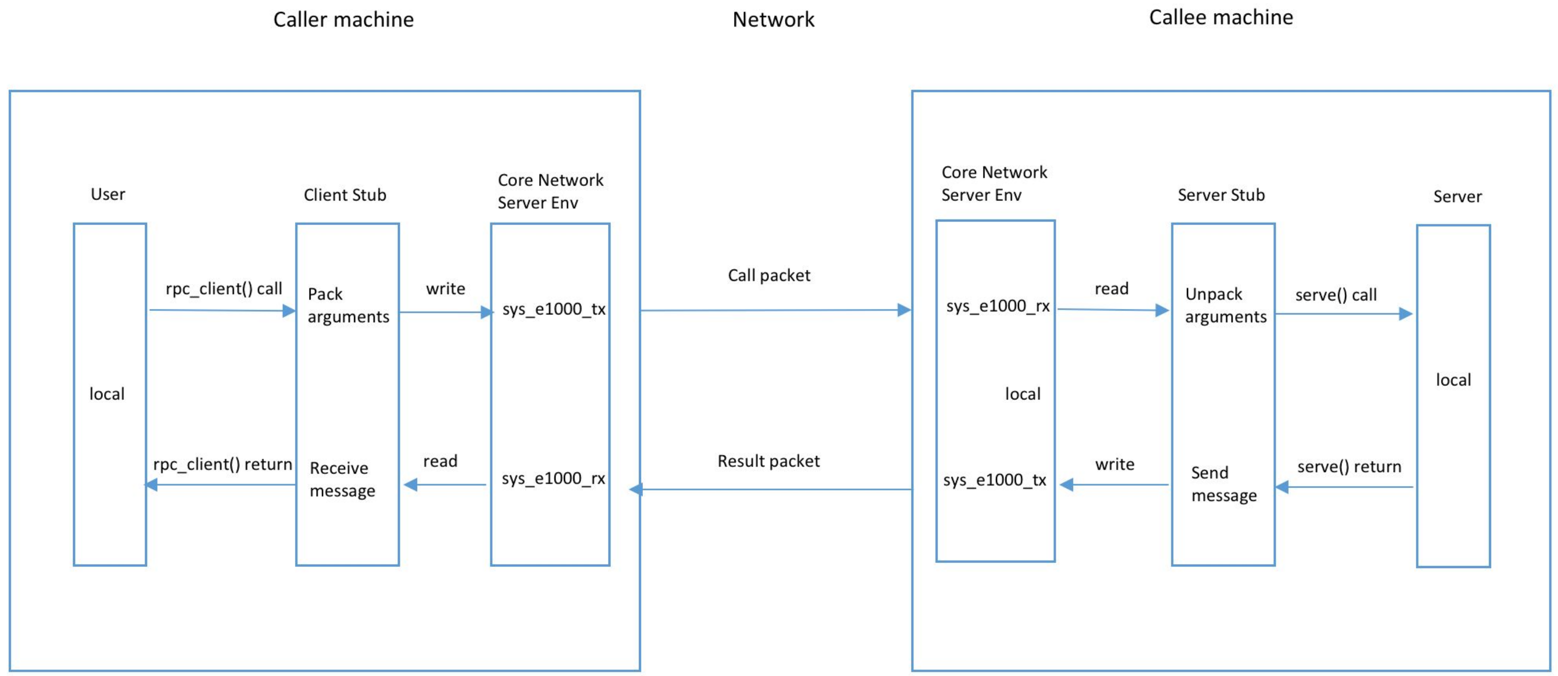


An architecture diagram from JOS lab 6

Part II: Remote Procedure Call

- Data types
  - Currently support Char, Int and String as arguments
  - At most 6 arguments can be used
- Implementation
  - RPC support is written as a user-level library
  - RPC client passes arguments by local call to the stub code
  - RPC server binds a function to be called remotely
  - All the arguments are packed into a single network packet
  - The stub code uses socket interface to transmit RPC packet and results

Part II Architecture



An architecture diagram of RPC

Demo Time

- Part I demo:
    - Trust us, it works
- Part II demo:
    - A distributed Key-Value Store using RPC library

Things to improve

- Ability to use non-hardcoded MAC addresses
  - Load from EEPROM
- Multithreaded RPC server
  - Need to change the server stub code to support multithreaded RPCs
  - Also need to synchronize user requests in conflict
- Dynamically generate stub code
  - Need to write a compiler to generate C stub code based on user-defined interfaces
- Reduce copy overhead
  - Need to add kernel support to directly copy arguments into kernel network buffer

Q & A