Today’s Reminders / Announcements

• Course home pages are no longer homeless
  – Home Address: http://www.cs.cmu.edu/~15-712/

• No Class Monday – MLK Jr Day

• I plan to have the semester schedule in place by Monday
  – Late Sunday Night, I will post the Readings for next week
“Implementing Remote Procedure Calls”
Andrew Birrell & Bruce Nelson 1984

• Bruce Jay Nelson (Xerox PARC)
  – Dissertation “Remote Procedure Call” (@ CMU!)
  – Chief Science Officer @ Cisco; d.1999

• Andrew Birrell (Xerox PARC)
  – Dryad (2007)
“Implementing Remote Procedure Calls”
Andrew Birrell & Bruce Nelson 1984

- Awarded ACM Software System Award in 1994

SigOps HoF citation (2007):
“This is THE paper on RPC, which has become the standard for remote communication in distributed systems and the Internet. The paper does an excellent job laying out the basic model for RPC and the implementation options.”
Goal: Communication Across a Network Between Programs Written in a High-level Language

• Use Message Passing?
  – Would encounter same design problems
  – Procedure calls already in high-level languages
  – Aside: HPC community uses MPI (messages)

• Use Remote Fork?
  – Would encounter same design problems
  – Also, what do you return, entire contents of memory??
Goal: Communication Across a Network Between Programs Written in a High-level Language

• Use Distributed Shared Memory?
  – Need to represent remote addresses in PL (may need to extend address width)
  – Likely too costly, done at page granularity

Aside:
  – Long history of research into general DSM (mixed success)
  – Successful specialized DSM: key-value stores, parameter servers
Remote Procedure Calls

• Make Distributed Computation easy
  – Procedure calls are a well-known & well-understood mechanism for transfer of program control and data
  – Clean & simple semantics; Simple enough to be efficient; General
  – Design goal: RPC semantics should be as close as possible to single-machine procedure call semantics (e.g., no time-outs)

• Make semantics of RPC package as powerful as possible, w/o loss of simplicity or efficiency

  “We wanted to make RPC communication highly efficient (within, say, a factor of five beyond the necessary transmission times of the network).”

• Provide secure communication with RPC
Remote Procedure Calls

• Not a new idea: e.g., discussed in 1976 paper

• Major issues facing the designer of an RPC facility:
  – Precise semantics of a call in the presence of failures
  – Semantics of address-containing arguments in the absence of a shared address space
  – Integration of remote calls into existing programming systems
  – Binding (how caller determines location/identity of callee)
  – Suitable protocols for transfer of data & control between caller and callee
  – How to provide data integrity and security in an open communication network
Components of the System

On single-machine, call still works

Fig. 1. The components of the system, and their interactions for a simple call.
Packets Transmitted in Simple Call

- When Arguments & Return result each fit in a single packet

- Caller periodically probes, and callee acks
  - Less work for server vs. pushing “I’m alive” messages
  - Caller can get “called failed” exception (unlike a local call)
Discussion: Summary Question #1

• **State the 3 most important things the paper says.** These could be some combination of their motivations, observations, interesting parts of the design, or clever parts of their implementation.
A Complicated Call
Stubs

• User-stub and server-stub are automatically generated, using **Mesa interface modules** (basis for separate compilation)
  – Specification Interface: List of procedure names, together with the types of their arguments and results

• Lupine stub generator checks that user avoids specifying args/results that are incompatible with the lack of a shared address space
• **Naming**
  
  – Use Mesa interface module name + instance
Discussion of Binding

• Use Grapevine distributed database
  – Maps type to set of instances
  – Maps instance to network address

• Importing an interface has no effect on exporting machine’s data structures (scalability)

• Unique identifier means bindings broken on server crash

• Can only call explicitly exported procedures

• Grapevine enforces access control, facilitates authentication

• Options include importer specifies type and gets nearest
Packet-level Transport Protocol

• Up to factor of 10 faster than using general protocols
  – Not doing large data transfers
  – Goals: Minimize latency to get result & Server load under many users (minimize state info & handshaking costs)

• Guarantee: If call returns, procedure invoked exactly once

• Do not abort on server code deadlock or infinite loop. Why not?

• When connection is idle, only a single machine-wide counter

• Rate of calls on ExportInterface in a single machine limited to an average rate of less than one per second
Minimizing Process Swap Cost

• Maintain at each machine idle server processes ready to handle incoming packets (avoid process creates)

• For simple RPCs, only 4 process swaps

• Also, bypass SW layers for intranet RPCs
  – Modularity vs. performance trade-off
  – Today: RDMA
Discussion: Summary Question #2

• **Describe the paper's single most glaring deficiency.** Every paper has some fault. Perhaps an experiment was poorly designed or the main idea had a narrow scope or applicability.
“At present it is hard to justify some of our insistence on good performance because we lack examples demonstrating the importance of such performance.”
• Describe what conclusion you draw from the paper as to how to build systems in the future. Most of the assigned papers are significant to the systems community and have had some lasting impact on the area.
• Multicasting or broadcasting can be better than RPCs

Aside: Today?
– Remote Method invocation
– Naming via key-value store containing hostname (or IP address) and port number
– RPC over UDP/IP (unreliable, but more efficient); IP handles multi-packet arguments
– Sun’s RPC (ONC RPC) system, with stub compiler rpcgen, is widely used, e.g., in Linux; provides XDR, a common data representation in messages
A Stub Generator for Multilanguage RPC in Heterogeneous Environments

P. B. Gibbons

77
A Stub Generator for Multilanguage RPC in Heterogeneous Environments

January 1987, pp. 77-87, vol. 13
DOI Bookmark: 10.1109/TSE.1987.232837

Keywords
Stub Generator, Argument Marshalling, Distributed Systems, Heterogeneous Environments, Interface Specification Language, Interlanguage Procedure Call, Remote Procedure Call

Authors
P.B. Gibbons, Department of Electrical Engineering and Computer Science, Computer Science Division, University of California

Abstract
A stub generator for marshalling the arguments and results of remote procedure calls in heterogeneous environments is presented. The stub generator is itself language and machine independent, and derives all its knowledge of source languages and machine types from a set of language and machine specifications. These specifications can be paired in any combination to accommodate interlanguage calls between differing machines.
Wednesday’s Paper

TBD 😊