Remote Procedure Calls

Implementing Remote Procedure Calls

Andrew D. Birrell, Bruce Jay Nelson

Lightweight Remote Procedure Call

Brian Bershad, Tom Anderson, Ed Lazowska, Hank Levy

Presented by Mark Reitblatt

Implementing Remote Procedure Calls

- ACM System Software Award
- Andrew Birrell
  - Xerox PARC, DEC, MSR (current)
- Bruce Jay Nelson
  - CMU PhD, Xerox PARC, Cisco

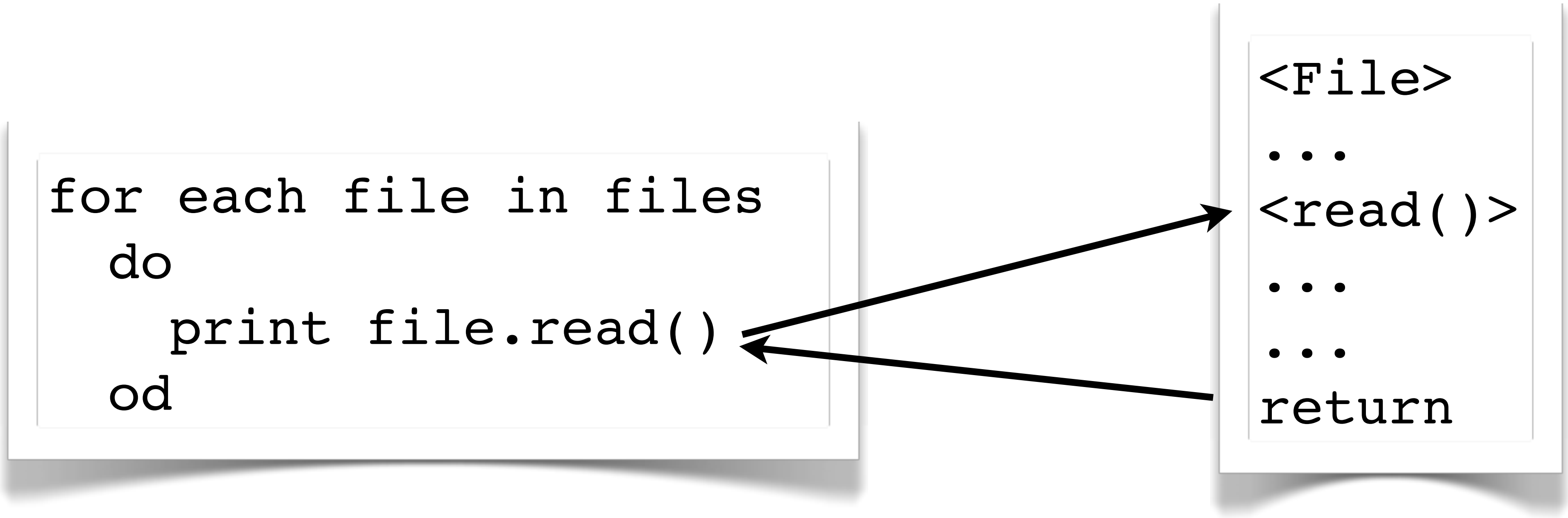
RPC

- How do we program distributed systems?
- What's the fundamental abstraction?
  - Message passing
  - Shared memory
  - Fork-Join

RPC Goals

- Clean, simple programming model
  - Transparent distribution
  - Late binding
    - Decide distribution at runtime
- Efficiency
- Generality

Procedures

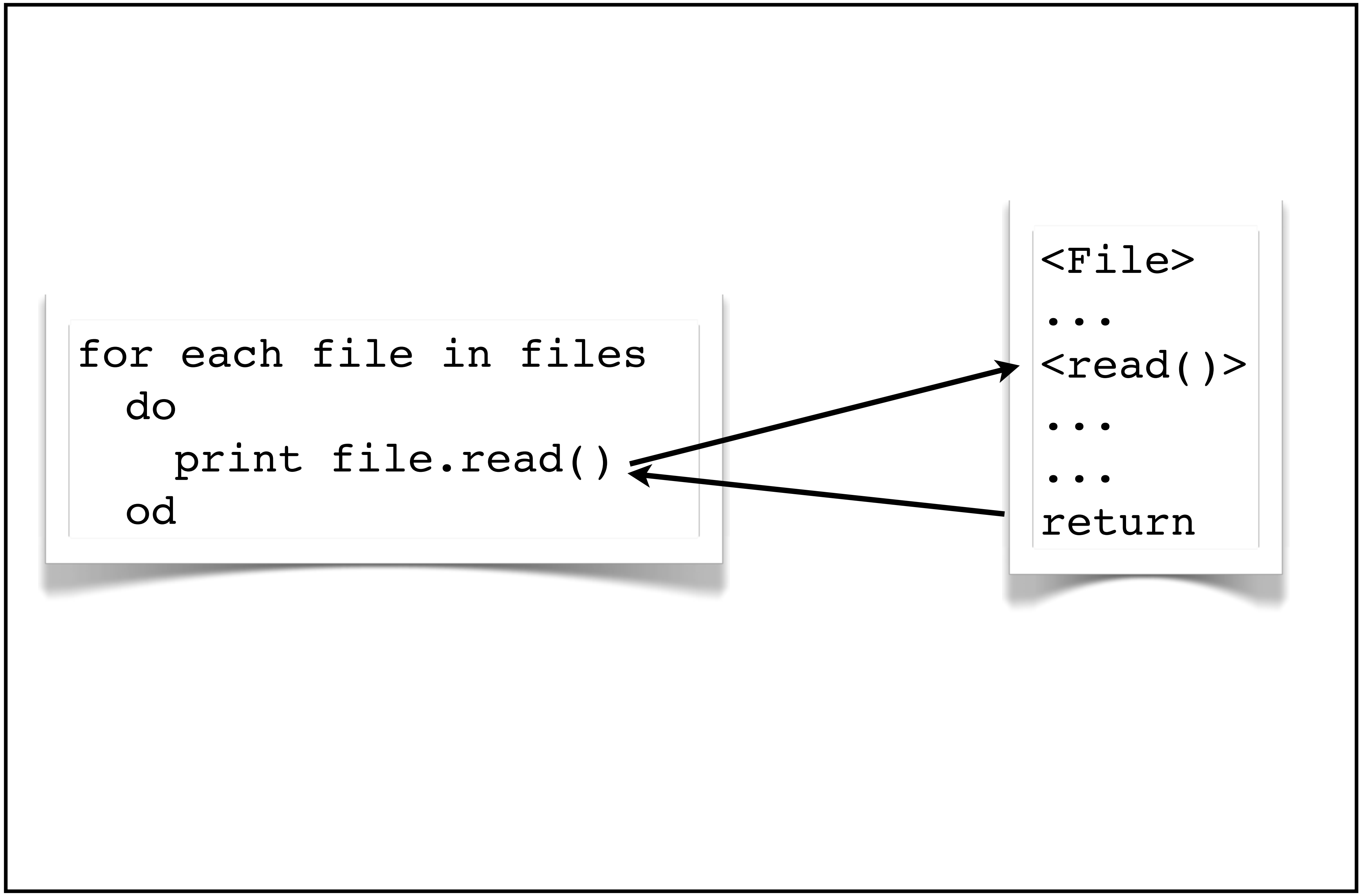


```
for each file in files
  do
    print file.read()
  od
```

```
<File>
...
<read()>
...
...
return
```

- Simple
- Well understood
- Easy-to-use
- Common

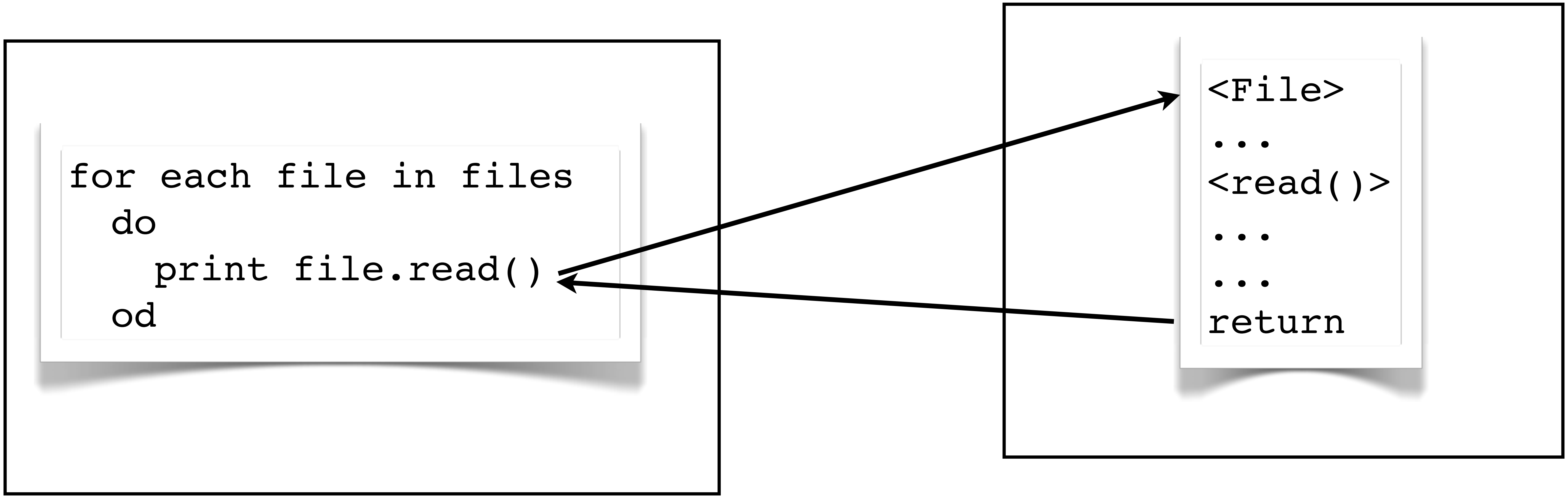
Distributed Application



```
for each file in files
  do
    print file.read()
  od
```

```
<File>
...
<read()>
...
...
return
```

Distributed Application

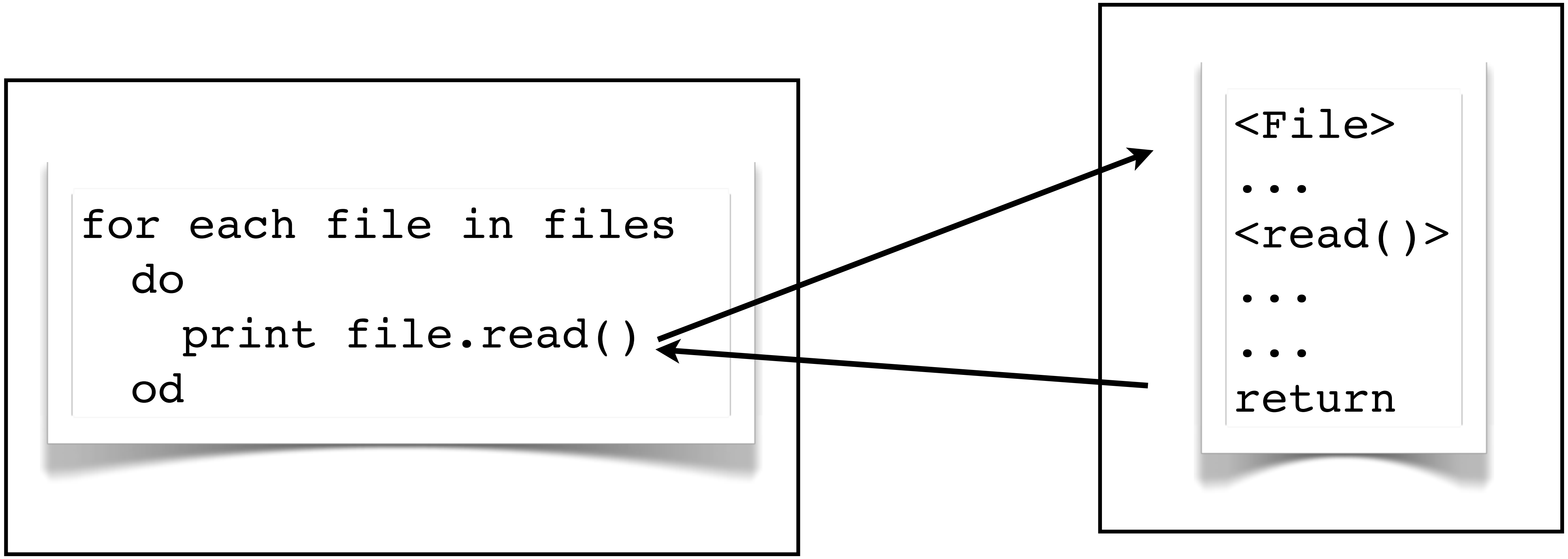


```
for each file in files
  do
    print file.read()
  od
```

```
<File>
...
<read()>
...
...
return
```

Just distribute the procedure to another machine

Distributed Application



```
for each file in files
  do
    print file.read()
  od
```

```
<File>
...
<read()>
...
...
return
```

But there's no cross-machine "`jmp`"

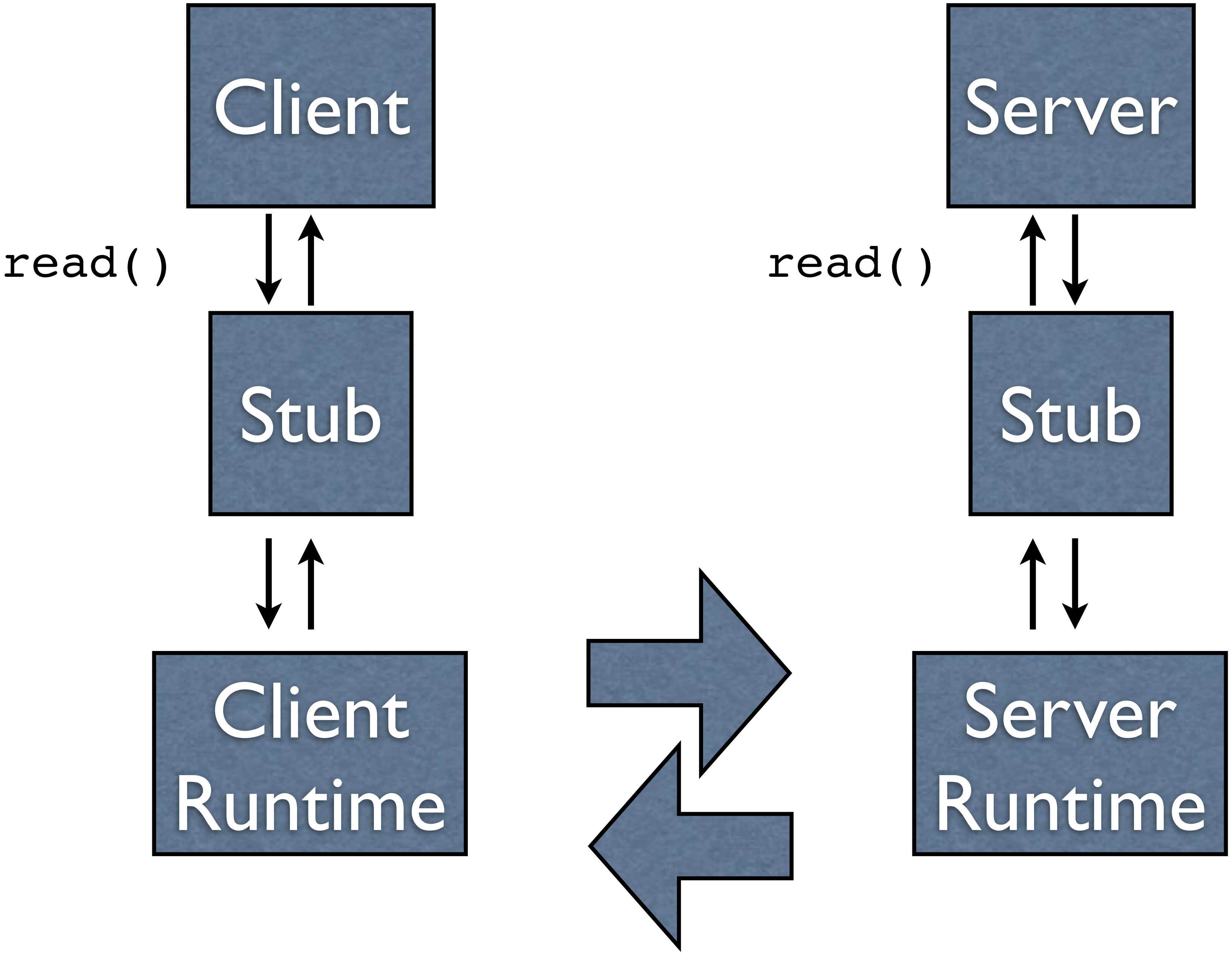
Need to deal with the network now

RPC

- Mechanism exposes stub procedure entry point
- Stub transforms call into packets
- RPC Runtime finds server, sends call
- Reverse process on server

```
for each file in files
  do
    print file.read()
  od
```

```
<File>
...
<read()>
...
...
return
```



Client

read()

Stub

Client Runtime

Server

read()

Stub

Server Runtime

Stubs

- Intercept procedure call
- Serialize arguments
- Automatically generated
  - Advantage of high-level language

Runtime

- Handles transmission to/from server
- Shared across different stubs
- End-to-end principle?

Simple?

- Remote calls aren't local calls
  - Pass-by-reference?
  - Network related exceptions
  - Out-of-memory exception?
  - Call-backs?

Binding

- How to match signatures to instances
  - Signature: FileAccess.Alpine
  - Instance: Ebbets.Alpine
- Where is the instance?
  - Ebbets.Alpine -> 3#22#
- Is this still transparent?

Lightweight Remote Procedure Call

- Brian Bershad
  - UW PhD, CMU Prof, SPIN, returned to UW
- Tom Anderson
  - UW PhD, Berkeley Prof, returned to UW
- Ed Lazowska
  - UW Prof
- Hank Levy
  - UW Prof

LRPC

- Enter µ-kernels:
  - Split into separate address spaces (protection domains)
  - RPC common interface
    - “Remote” across domains on single machine
    - Deals with disjoint address spaces

LRPC (Cont.)

- Cross-domain RPC expensive
  - Forces functionality into single domains for performance
  - Defeats purpose of μ-kernel

Make the common case fast

- Cross-domain RPC
  - 100-30x more common
  - Tends to be simple
    - Few, small parameters

Why is the common case slow

- Stub overhead
- Message buffer overhead
  - Double buffer
- Message transfer
- Context switch

LRPC Solution

- Essentially a bunch of hacks that "work"
  - Remap argument stack into server domain
  - Client provides thread for execution
  - Domain caching
  - Single arg copying into A stack

Performance

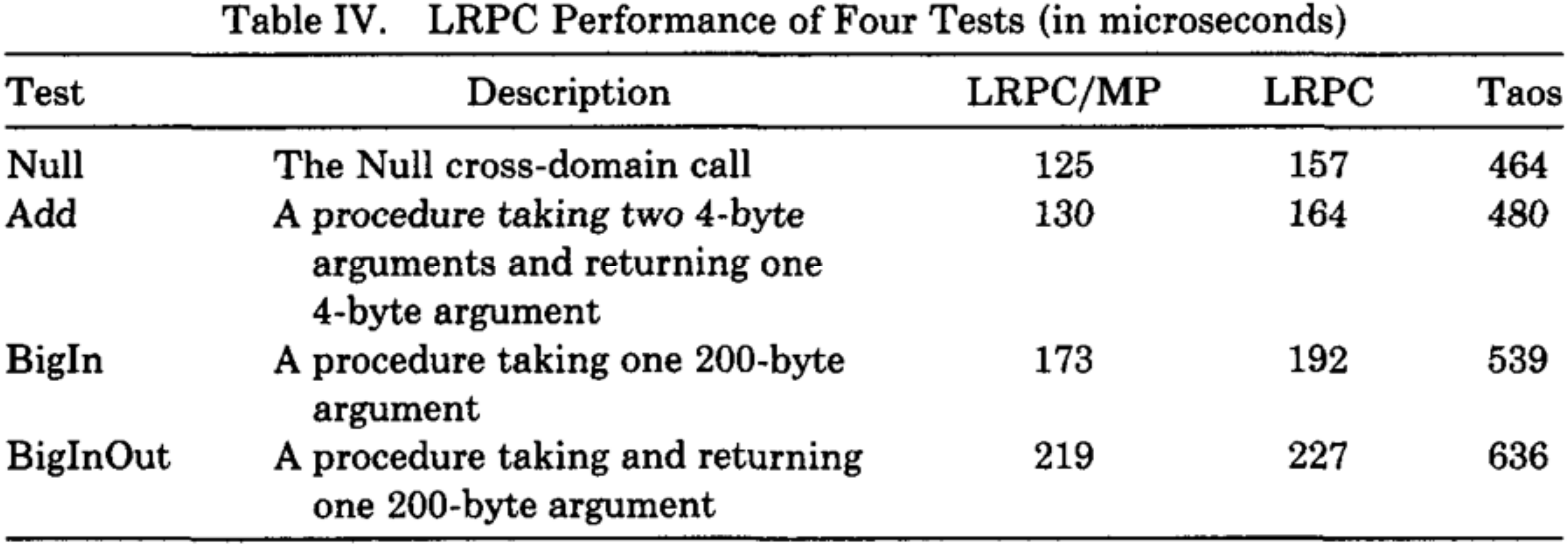


Table IV. LRPC Performance of Four Tests (in microseconds)

Test	Description	LRPC/MP	LRPC	Taos
Null	The Null cross-domain call	125	157	464
Add	A procedure taking two 4-byte arguments and returning one 4-byte argument	130	164	480
BigIn	A procedure taking one 200-byte argument	173	192	539
BigInOut	A procedure taking and returning one 200-byte argument	219	227	636

Performance Cont.

Fig. 2. Call throughput on a multiprocessor.